VisEscape: A Benchmark for Evaluating Exploration-driven Decision-making in Virtual Escape Rooms

Anonymous ACL submission

Abstract

Escape rooms present a unique cognitive challenge that demands exploration-driven planning: with the sole instruction to *Escape the room*, players must actively search their environment, collecting information, and finding solutions through repeated trial and error. Motivated by this, we introduce **VisEscape**, a benchmark of 20 virtual escape rooms specifically designed to evaluate AI models under these challenging conditions, where success depends not only on solving isolated puzzles but also on iteratively constructing and refining spatial-temporal knowledge of a dynamically changing environment. On VisEscape, we observe that even state-of-the-art multi-modal models generally fail to escape the rooms, showing considerable variation in their progress and problem-solving approaches. We find that integrating memory management and reasoning contributes to efficient exploration and enables successive hypothesis formulation and testing, thereby leading to significant improvements in dynamic and exploration-driven environments [1].

1 Introduction

Realistic goals often necessitate exploration: when cooking with available ingredients, one must search through cabinets to identify ingredients and determine how to combine them into a meal. Likewise, a trip to a foreign city cannot be planned without searching for major sights and train schedules. In such cases, effective planning depends on acquiring information through interaction with the environment rather than having it provided a priori.

Existing works on embodied agents have addressed the importance of exploration, but it has primarily been studied in the context of decomposable and explicit tasks like "Place a cleaned egg in a microwave" (Shridhar et al., 2020; Puig et al., 2018; Li et al., 2021; Kim et al., 2024). However, the capability of agents to autonomously search for and utilize relevant information in environments where tasks and solutions are not predefined has not been sufficiently investigated.

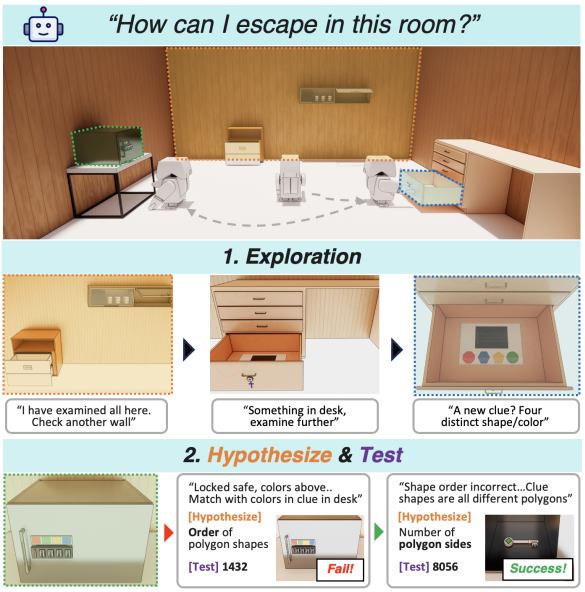


Figure 1: Depiction of the exploration-driven problem-solving in VisEscape. Agents must (1) actively explore to uncover relevant information and (2) subsequently formulate and test hypotheses through interaction to solve puzzles to a successful escape.

To bridge this gap, we argue that practical evaluation of decision-making agents should assess their ability to conduct self-directed exploration to identify underlying objectives and subsequently take reasoned actions. In particular, this entails formulating hypotheses for problem-solving based on information and clues gathered through exploration.

To evaluate agents under these conditions, we propose VisEscape, a benchmark of 20 virtual escape rooms for assessing exploration-centric problem-solving capacities of multimodal agents. Given the implicit goal of "escape the room",

[1] We will publicly release the assets and code

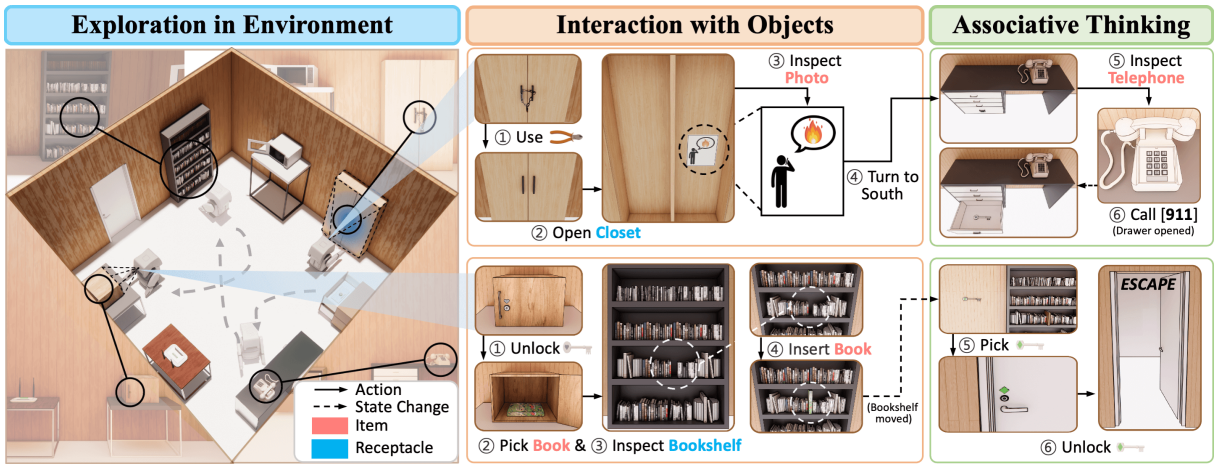


Figure 2: An illustration of an excerpt of a trajectory from VisEscape. To escape the room successfully, agents must explore multiple directions and diverse views, and interact with various objects. Additionally, they need to infer associations between two or more scenes in different locations to solve creative puzzles.

agents explore their surroundings, integrate relevant information, discover how to use objects, and solve complex puzzles to escape. Moreover, they must retain previously observed information and interpret indirect clues, and reason beyond immediate observations while dynamically adapting to environmental changes.

The evaluation of various MLLMs on VisEscape showed that even advanced models like Claude-3.5-Sonnet and GPT-4o exhibited low escape rates of less than 10% without any hints. To test how memory and reasoning, which are core cognitive abilities for both agents and humans (), navigate the cognitive challenges posed by VisEscape, we apply dedicated memory management and reasoning modules to models.

We find that this integration enables models that were previously wholly unsuccessful in escaping to achieve successful escapes; achieving x2.4 enhanced effectiveness and x3.8 improved efficiency. Through analysis, we observe that memory management reduces repetitive actions, enabling efficient exploration, and that memory and reasoning exhibit a synergistic effect.

2 Key Challenges of VisEscape

Figure 2 illustrates an example of the problem-solving process required in the VisEscape. The game agent should execute a sequence of tasks comprising diverse interactions.

1. Self-directed Exploration. As in real escape room games, the game agent in VisEscape receives no instruction except for the instruction 'escape the room'. Therefore, without a clear roadmap, the agent must actively explore and figure out how to proceed through trial and error. This setup requires the agent to build and adjust its mental model of the environment, recognizing key objects, spatial relationships, and potential interactions. Success depends on the agent's ability to hypothesize, test, and adapt its understanding. The task mirrors real-world reasoning challenges, where goals must be inferred, and solutions emerge through engagement.

2. Long-Horizontal successive task. In the absence of explicit guidance, the agent must achieve an average of **8.4 checkpoints** (critical tasks that act as bottlenecks in achieving goal completion), which requires an average of **27.2 steps** to successfully escape, even for the oracle trajectory. The considerable length of the task sequence and the dependencies between checkpoints require the agent to effectively utilize information from its past trajectory. Success hinges on the agent's ability to integrate prior discoveries into its ongoing decision-making process.

3 Details on Dataset Composition and Construction

3.1 Dataset Composition

VisEscape consists of 20 distinct rooms. Each room comprises four walls corresponding to the four cardinal directions, containing multiple objects classified into two types: *Receptacles* and *Items*, following the object categorization criteria used in (Shridhar et al., 2020). *Receptacles* are objects fixed within a room; they can be secured with locks or locking mechanisms and are capable of concealing or containing items. *Items* can be collected into the inventory, and serve two functions: facilitating interactions with other recepta-

Figure 3: An illustration of each component in VisEscape.

	Scene	Receptacle	Item	Quiz	Interaction
# Total	1084	118	102	27	311
# Types	662	32	23	27	37
Avg./room	54.2	5.9	5.1	1.4	15.6

Table 1: Statistics for scenes, receptacles, items, quizzes, and interactions in VisEscape.

cles or providing clues to unlock locks. VisEscape has 32 different receptacles and 23 different items, with each room containing a distinctive visual quiz. Each room contains an average of 5.9 receptacles and 5.1 items, along with one or two visual quizzes. Statistics and examples are presented in Table 1 and Figure 3.

Visual Quizzes. Escape rooms challenge players to connect diverse pieces of information gathered through exploration, often by linking elements that initially appear unrelated. For example, players may need to associate similar symbols found in different locations or identify how objects of the same color are interconnected. Recognizing that escape rooms are well-suited for assessing the capability of AI models to draw inferences from environmental cues and formulate hypotheses for problem-solving, we incorporate visual quizzes. We integrate one to two number or letter code locks paired with corresponding hand-crafted visual quizzes into each room.

Game Graph. To track the dynamically changing state of the room and provide corresponding observations, we use a Finite State Machine (FSM). FSM consists of two components: **states**, which correspond to the visual observations provided to the agent, and **transitions**, which define state changes based on player actions and specific conditions being met.

Actions. Agent can perform two types of actions: movement and object interaction. For movement, the agent can *inspect [object]* to examine it closely or *step back* to gain a broader view. Additionally, it can *turn to [direction]* to view walls that are not currently visible. For object interaction, the agent can use acquired items (e.g., Unlock with key) or perform actions associated with visible objects (e.g., Turn on laptop). When the agent observes a lock requiring a quiz answer, an additional action to input answer becomes available.

At each gamestep, the agent is given actions for movement and other possible actions based on its current state and observation. When the agent performs an action that satisfies an object's predefined conditions or corresponds to the lock's answer, it triggers state transitions according to game graph. See Figure 15 for examples of object interactions.

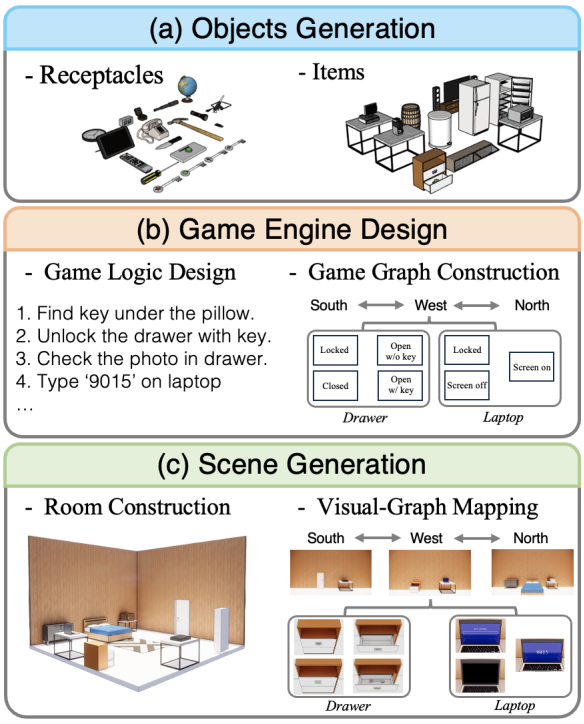


Figure 4: Process of VisEscape construction.

3.2 VisEscape Construction

Figure 4 shows the creation process for rooms in VisEscape. **(a)** Objects are created using 3D modeling software, and those required for arrangement in each room were sampled. **(b)** Using the sampled objects, game logic that defines key checkpoints to complete the game is designed through AI-human collaboration. Game graph is then constructed as an FSM to track the overall game state and enable state transitions triggered by agent's actions. **(c)** Objects are arranged within the room according to its setup, and visual observations are mapped to corresponding game states. See Appendix B for detailed explanations of each process and prompts used for game logic design.

	Frontier MLLMs					*Open-source MLLMs*			
Model	**SR↑(%)**	**GC↑(%)**	**SPL↑(%)**	**Step↓**	**Model**	**SR↑(%)**	**GC↑(%)**	**SPL↑(%)**	**Step↓**
Claude 3.5 Sonnet	**6.3**	11.11	**3.35**	**286.7**	*Qwen2.5VL-32B*	1.7	5.71	0.26	298.1
GPT-4o	3.3	16.00	0.60	294.5	*InternVL2.5-38B*	0.0	5.75	-	-
GPT-4o-mini	0.0	5.37	-	-	*InternVL2-40B*	0.0	0.0	-	-
Gemini-1.5-Pro	0.0	**18.51**	-	-	*LLaVA-v1.6-34B*	0.0	0.0	-	-
Gemini-2.0-Flash	0.0	9.54	-	-	*LLaVA-OneVision-7B*	0.0	0.0	-	-
Human	82.5	95.80	51.30	52.8					

Table 2: Performance comparison of various MLLMs and human performance on the VisEscape benchmark. The best results for each metric are **bolded**. Refer Appendix F for detailed information on the human study. For models that failed to escape entirely, the standard deviation is marked with "-".

4 Examining MLLMs on VisEscape

We evaluate the performance of various MLLMs, ranging from closed-source frontier models to open-source models on the VisEscape benchmark.

4.1 Experiment Setup

At each step, the model receives the current observation, history of the last 20 observations and actions, held items, and available actions, from which it selects one executable action. For each room, we conducted three different trials and averaged the resulting metrics. Each experiment was terminated early if no significant progress occurred within 100 consecutive steps and was automatically concluded after 300 steps.

4.2 Metrics

We adopted widely used metrics for evaluating embodied agents. These include: **1. Success Rate (SR)**, which is the ratio of escape success in an episode; **2. Goal Completion (GC)**, the ratio of checkpoints completed at the end of an episode; **3. Step**, the total actions taken by an agent; and **4. Success weighted by Path Length (SPL)** (Anderson et al., 2018), which penalizes success rate by total number of steps taken.

4.3 Results

Table 2 presents the experimental results. For open-source models, none of the open-source models achieved a single success, except for *Qwen2.5VL-32B*. Notably, even sophisticated models such as *Claude 3.5 Sonnet*, *GPT-4o*, and *Gemini-1.5-Pro* demonstrated success rates below 10%. These results underscore the highly challenging nature of the exploration-centric problem-solving environment provided by VisEscape for AI models, concurrently highlighting significant avenues for improvement.

For models to perform well in VisEscape, they must effectively execute two primary capabilities that VisEscape requires: exploration and iterative hypothesis formulation and test. Rather than devising a specialized framework for the unique 'escape room' task, we aim to analyze how modules considered crucial in agent modeling contribute to the exploration-centric problem-solving tasks provided by this benchmark. To this end, we augment VisEscaper with Memory management and Reasoning modules.

5 VisEscaper: Exploring Agent in VisEscape

5.1 Memory Management Module

In VisEscape, agent can only observe a limited portion of the room. For effective navigation and exploration, the agent should maintain spatial memory that the agent constructs to form a coherent mental map of the space. Additionally, the agent must remember the actions it has attempted to learn from its trials and avoid redundant efforts.

Therefore, we designed and applied a Memory Management module for the game agent (Figure 5-(1)). The memory module consists of **structured spatial memory**, **exploration memory**, and **salient action memory**, which are constructed and updated periodically. It compressively and structurally manages memory from observation and action histories accumulated over tens to hundreds of steps, while also updating periodically. This allows the agent to make clear decisions from a condensed history, effectively mitigating its processing overload by addressing both cognitive demands and the model's inherent context window constraints.

For the experiment, memory construction begins after the first 10 steps. Subsequently, memory management is performed every 10 steps. To avoid exceeding the context length and performance drop (Zhao et al., 2024) with multiple image inputs, we used image captions generated by the same model to replace visual observations with tex-

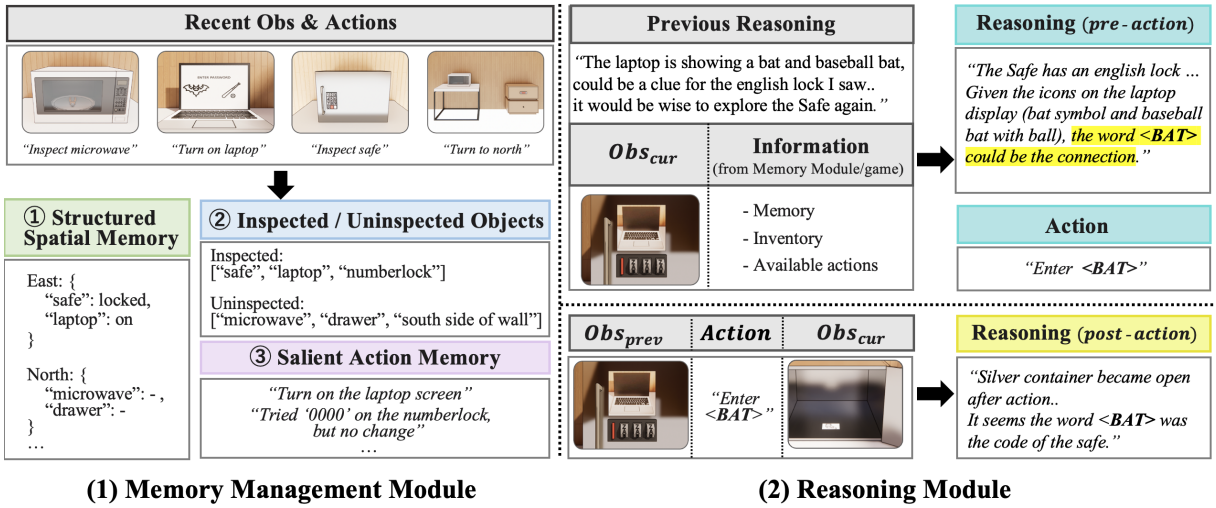


Figure 5: An overview of memory management module and reasoning module, along with examples of inputs (gray boxes) and outputs (colored boxes) for each module.

tual descriptions. Refer to Appendix D.2 for more details.

5.2 Reasoning Module

Exploration within escape rooms necessitates a cycle of hypothesis formulation and test. Specifically, the agent progresses by: 1) formulating hypotheses based on its memory and available information before action-decision, and 2) analyzing the outcomes to determine the validity of these hypotheses. Thus, we designed pre-action and post-action reasoning for hypothesis formulation and verification.

Pre-action reasoning, inspired by (Yao et al., 2023), requires the agent to 'think' before action selection. This approach not only facilitates more effective action decision but also enables the decomposition and analysis of the model's reasoning process; we find that the model formulates hypotheses in various ways. We will discuss this in §6.2.

Post-action reasoning involves the agent assessing the consequences of its actions, informed by the executed actions and the observations made immediately before and after those actions. This allows for hypothesis evaluation, and the results feed into the subsequent pre-action reasoning. The agent might then formulate different hypotheses or attempt new explorations, leading to more successful subsequent attempts.

5.3 Results

Table 3 shows scores of VisEscaper and the difference compared to BaseAgent. Frontier models improved across all metrics. For open-source models, goal completion generally increased, though

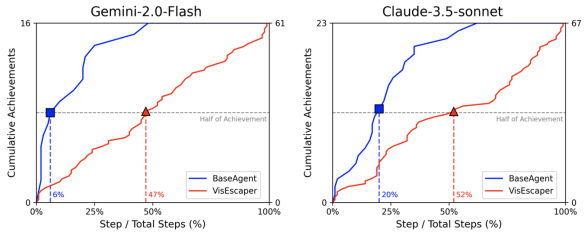


Figure 6: Cumulative goal completion over trajectory time steps for BaseAgent and VisEscaper in each room. For comparison, each agent's progression is shown on the same scale. Additionally, the point where 50% of goal completion is achieved is marked.

eventually not enough for room escape. However, the performance of *Qwen2.5VL-32B*, driven by the adoption of memory management and reasoning, increased to a level comparable to *Claude 3.5 Sonnet*. Notably, although higher success rates and goal completion typically require more steps, all models with VisEscaper nevertheless demonstrated a reduction in trajectory length, leading to an improvement in SPL. These results underscore that agents become more effective (achieving more) and more efficient (within fewer steps) when they are enabled with memory management and reasoning.

Furthermore, the design of VisEscaper enables the agent to achieve steady progress without becoming disoriented within the expanding information space. Figure 6 illustrates the cumulative goal completion for BaseAgent and VisEscaper over the course of a trajectory. Progress by BaseAgent is concentrated early in the trajectory (half of its achievements are realized within the initial 6% of steps for *Gemini-2.0-Flash* and 20% for *Claude 3.5 Sonnet*). In contrast, VisEscaper demonstrates

Model	BaseAgent → VisEscaper (*diff.*)			
	SR↑(%)	GC↑(%)	SPL↑(%)	Step↓
Frontier MLLMs				
Claude 3.5 Sonnet	6.7 → 21.7 (↑15.0)	11.11 → 32.37 (↑21.26)	3.35 → **10.92** (↑7.57)	286.7 → **249.5** (↓37.2)
GPT-4o	3.3 → 13.3 (↑10.0)	16.00 → 29.02 (↑13.02)	0.60 → 3.91 (↑3.31)	294.5 → 276.2 (↓18.3)
GPT-4o-mini	0.0 → 6.7 (↑6.7)	5.37 → 21.31 (↑15.94)	0.00 → 1.05 (↑1.05)	300.0 → 291.2 (↓8.8)
Gemini-1.5-Pro	0.0 → **25.0** (↑25.0)	18.51 → **42.16** (↑23.65)	0.00 → 6.59 (↑6.59)	300.0 → 258.1 (↓41.9)
Gemini-2.0-Flash	0.0 → 23.3 (↑23.3)	9.54 → 34.40 (↑24.86)	0.00 → 6.32 (↑6.32)	300.0 → 255.0 (↓45.0)
Open-source MLLMs				
Qwen2.5VL-32B	1.7 → 3.3 (↑1.6)	5.71 → 11.46 (↑5.75)	0.26 → 0.42 (↑0.16)	298.1 → 295.8 (↓2.3)
InternVL2.5-38B	0.0 → 0.0 (-)	5.75 → 6.20 (↑0.45)	0.00 → 0.00 (-)	300.0 → 300.0 (-)
InternVL2-40B	0.0 → 0.0 (-)	2.88 → 4.91 (↑2.03)	0.00 → 0.00 (-)	300.0 → 300.0 (-)
LLaVA-v1.6-34B	0.0 → 0.0 (-)	0.17 → 0.54 (↑0.37)	0.00 → 0.00 (-)	300.0 → 300.0 (-)
LLaVA-OneVision-7B	0.0 → 0.0 (-)	1.08 → 1.25 (↑0.17)	0.00 → 0.00 (-)	300.0 → 300.0 (-)

Table 3: Comparison of performance before and after applying the memory management and reasoning modules. For ease of reference, the agent operating without these two modules (Table 2) is referred to as 'BaseAgent'. Standard deviation for steps is in Table 13.

consistent progression.

Upon analyzing model trajectories, we observed failures predominantly occur in visual quizzes and are more pronounced in open-source models, which achieved smaller performance gains than frontier models even with memory and reasoning modules. We identified two primary reasons for this gap.

First, reasoning modules significantly boost visual quiz accuracy for Frontier models, an effect less evident in open-source models (Appendix C.2). Second, compared to Frontier models, open-source models tend to explore evidence less thoroughly and more frequently resort to brute-force attempts-resulting in meaningless and repetitive actions, ultimately causing them to get stuck in the game (Appendix C.1).

Model	exp_{hint}				
	SR(%)	GC(%)	HCR↓(%)	SPL (%)	Step
Frontier MLLMs					
Claude 3.5 Sonnet	90.0	96.33	**18.87**	**30.46**	**103.1**
GPT-4o	**95.0**	**97.39**	20.74	26.83	112.9
GPT-4o-mini	46.7	67.40	32.16	8.93	228.2
Gemini-1.5-Pro	71.7	82.53	20.18	18.22	163.3
Gemini-2.0-Flash	66.7	75.78	25.11	15.56	179.2
Open-source MLLMs					
Qwen2.5VL-32B	61.7	73.13	30.51	12.07	197.0
InternVL2.5-38B	66.7	79.23	41.69	12.23	210.5
InternVL2-40B	8.3	48.18	61.61	0.96	291.1
LLaVA-v1.6-34B	15.0	34.79	64.97	1.47	289.8
LLaVA-OneVision-7B	8.3	41.02	62.89	0.80	294.9

Table 4: Results when hints are allowed for models. **HCR** means **Hint-assisted Completion Rate**: the ratio of checkpoints completed by the agent with the assistance of hints, where lower is better.

5.4 Results on Hint-guided Experiment

However, despite overall improvements, if the model becomes deadlocked in certain challenging checkpoints and thereby blocks subsequent progress, comprehensive experimental analysis can become difficult. Thus, we also experimented with hint-guided setting; if the model is stuck for 30 steps or more, a guideline message for the next checkpoint is provided to the model. More details on hint-guided experiment are in Appendix A.4.

A comparison of performance between scores from VisEscaper of Table 3 and Table 4 shows that after hints are allowed, all models demonstrated a significant increase in success rate and goal completion. Notably, *Claude 3.5 Sonnet* and *GPT-4o* escaped rooms using fewer than half the steps compared to before receiving hints, leading to the greatest improvement, especially in SPL, which captures performance and efficiency together. Also, smaller models still improve substantially with hints, but rely on hints more frequently, as shown by higher HCR.

6 Deep Analysis on Each Module

6.1 Memory Management

Repetitive Actions. Ideally, an escape room player with good memory is less likely to repeat actions or re-explore areas. Consequently, to investigate how memory management influences an agent's action decision within a trajectory, we examined the action repetition in a trajectory from diverse models. We only included actions related to object interaction and answer submission, since *inspect, turn, step back* are part of the exploration

Model	*None.*	*Me.*	*Re.*	*Me&Re*
Claude 3.5 Sonnet	29.1	**24.3**	8.5	**5.7**
GPT-4o	27.8	**26.1**	20.3	**7.9**
InternVL2.5-38B	27.5	30.2	34.8	**21.6**
LLaVA-v1.6-34B	53.7	**50.8**	34.1	**24.6**
Diff. (Avg)	-1.6		**-9.5**	

Table 5: Ratio of repetitive actions. ***None***-BaseAgent, ***Me.***-BaseAgent +memory, ***Re.***-BaseAgent +reasoning, and ***Re.&Me.***-VisEscaper. *Diff (Avg).* denotes the averaged difference with or without memory.

process.

According to Table 5, memory management enabled efficient action decision by reducing the overall rate of repetition actions. However, the effect was greater when memory management was performed with reasoning (-9.5%) compared to when it was performed without reasoning (-1.6%). This suggests that the synergy between memory management and reasoning capabilities allows the agent to more effectively utilize past experiences to avoid redundant actions and make more optimal decisions. Analysis of difference between models are suggested in Appendix C.5.

6.2 Reasoning

Adopting reasoning process before action decision enables analyzing how agents in VisEscape perform reasoning and how they differ across models. By dividing their reasoning sentence into five components: Observation, Recall, Planning, Hypothesizing and Guess, we analyzed how they utilize these components and combine these components to perform reasoning. For categorization, we broke down the reasoning process and performed a machine-based evaluation for each piece with GPT-4o (Achiam et al., 2023). Specifically, we analyzed two models (*Claude 3.5 Sonnet* and *LLaVA-v1.6-34B*). The proportion of each component is shown in Figure 7, and its composition is in Table 6. The top 3 most frequent component combinations were reported. See Appendix D.1 for the criteria used to divide each module and human assessment for ensuring reliability.

Through decomposition, we find following results: **(Fig. 7) Memory Recall.** Across both models, they frequently referenced memory (26.9% and 29.2%). This highlights that reasoning can be influenced by the establishment of good memory. **(Tab. 6) Reasoning Composition.** *Claude 3.5 Sonnet* exhibits longer reasoning sequences, systematically combining various components. Notably, their reasoning often proceeds after revisiting current observations and past memories. In contrast, *LLaVA-v1.6-34B* demonstrates simpler and shorter reasoning sequences. **(Fig. 7 &Tab. 6) Hypothesis Formulation.** *Claude 3.5 Sonnet* frequently attempts to formulate reasoned hypotheses, often preceded by analyzing the current observation and consulting its memory before formulating hypothesis, while *LLaVA-v1.6-34B* rarely does so.

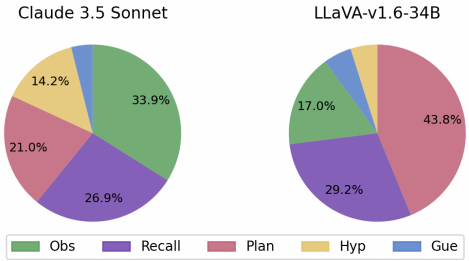


Figure 7: Proportion of each reasoning component.

7 Replacing VLM with LLM

Socratic models, as represented by (Shin et al., 2024), is an approach that uses image captioners to convert visual inputs into descriptive text inputs and then uses LLMs to perform multimodal inference. Given that MLLMs excel at visual perception but often struggle with complex visual reasoning (Li et al., 2024b; Gan et al., 2022; Liu et al., 2023b), we adopt LLMs to address this gap.

We compared the results from two experimental setups as §4.1: one where the MLLM directly processed the visual inputs, and another where a LLM processed image captions generated by an MLLM. For controlled experiments, we used only the backbone LLM of the corresponding MLLM-refer to Table 8. Across the models investigated, excluding *InternVL2.5-38B*, improvements were observed in most metrics when socratic method was adopted instead of MLLMs.

Notably, for *InternVL2.5-38B*, which underperformed with *Qwen-32B-Instruct*, switching to *Deepseek-R1-Distill-Qwen-32B*-sharing the same architecture but with reasoning capabilities enhanced by distillation from Deepseek-R1 (Guo et al., 2025)—markedly improved all evaluation metrics. These results suggest that: 1. Many MLLMs struggle with complex reasoning in multimodal contexts—a critical capability for VisEscape, and 2. Integrating strong reasoning capabilities of LLMs with visual perception of VLMs can lead to enhanced multimodal reasoning capability.

Model	Composition	Ratio	Length	Example
Claude 3.5 Sonnet	(*Obs*, *Recall*, *Plan*)	18.1	127.0	[*Obs*] I'm facing Door with Keycard lock. [*Recall*] I haven't specifically tried Entrancecard. [*Plan*] I should inspect the lock.
	(*Obs*, *Recall*, *Hypo*, *Plan*)	15.8	141.5	[*Obs*] We're facing a door lock. [*Recall*] Seems to connect with the kiosk's display. [*Hypo*] This suggests we need to find four words. [*Plan*] Return to the kiosk to check.
	(*Obs*, *Recall*, *Plan*, *Hypo*)	14.8	127.5	[*Obs*] We're currently facing locked box. [*Recall*] Already tried some number combinations. [*Plan*] Re-checking east wall poster might be worthwhile. [*Hypo*] We may find all primes numbers in poster.
LLaVA-v1.6-34B	(*Recall*, *Plan*)	38.5	55.3	[*Recall*] I've already inspected wardrobe &door. [*Plan*] I should inspect sensor lock on the door.
	(*Plan*)	21.3	39.4	[*Plan*] I will inspect coffee machine to see if there are any clues or items.
	(*Obs*, *Plan*)	11.7	54.4	[*Obs*] Locker is currently locked. [*Plan*] I should try to unlock it to see if there are any clues.

Table 6: The top 3 compositions of reasoning components most frequently utilized by *Claude 3.5 Sonnet* and *LLaVA-v1.6-34B*. Token lengths are calculated using the 'token-count' library, set ting model as gpt-3.5-turbo.

		exp_{base}				exp_{hint}				
Model(MLLM / LLM)	Input Modality	SR↑(%)	GC↑(%)	SPL↑(%)	Step↓	SR↑(%)	GC↑(%)	HCR↓(%)	SPL↑(%)	Step↓
InternVL2.5-38B	Image	0.0	6.20	-	-	66.7	79.23	41.69	12.23	210.5
InternVL2.5-38B + Base LLM	Caption	1.7	9.57	0.67	295.9	35.0	53.76	38.14	7.08	245.3
InternVL2.5-38B + Base LLM (R1)	Caption	**10.3**	**19.58**	**7.81**	**261.6**	**80.0**	**87.06**	**25.53**	**19.37**	**166.8**
InternVL2-40B	Image	0.0	4.91	-	-	8.3	48.18	61.61	0.96	291.1
InternVL2-40B + Base LLM	Caption	0.0	3.88	-	-	**32.0**	**68.00**	**48.22**	**4.39**	**250.6**
LLaVA-v1.6-34B	Image	0.0	0.54	-	-	15.0	34.79	64.97	1.47	289.8
LLaVA-v1.6-34B + Base LLM	Caption	0.0	**4.14**	-	-	**38.3**	**61.80**	**50.08**	**4.59**	**257.7**
LLaVA-OneVision-7B	Image	0.0	1.25	-	-	8.3	41.02	62.89	0.80	294.9
LLaVA-OneVision-7B + Base LLM	Caption	0.0	**1.62**	-	-	**25.0**	**58.93**	70.70	**2.40**	**287.3**

Table 7: Comparative performance of results from MLLMs and Socratic method. Experiment is conducted by models with memory management and reasoning module. exp_{base} and exp_{hint} denote experiments conducted without hints and hint-guided experiments, respectively.

8 Related Works

8.1 Interactive Environments for Evaluating LLM/VLM

Text-based games have long been used to benchmark LLMs, offering interactive settings to assess reasoning and behavior (Côté et al., 2019; Hausknecht et al., 2020; Qian et al., 2024). More recent benchmarks extend this to testing scientific knowledge (Wang et al., 2022) and ethical reasoning (Pan et al., 2023). Multimodal environments like Minecraft and embodied agent simulators have further enabled evaluation on both LLM and VLM agents in visually grounded settings (Wang et al.; Liu et al., 2024; Dong et al., 2024; Guss et al., 2019; Kolve et al., 2017; Puig et al., 2018; Shridhar et al.; Manolis Savva* et al., 2019). GUI-based benchmarks (Deng et al., 2024, 2023; Xu et al., 2021) now assess how these models perceive and act on web interfaces, testing their ability to recognize visual elements, reason about functions (Shahbandeh et al., 2024), and perform tasks (Shi et al., 2017; Furuta et al., 2023; Yao et al., 2022; Zhou et al.).

8.2 LLM/VLM as an Action Decision Maker

The effective deployment of LLM/VLM agents in real-world applications requires adaptability to dynamic and unpredictable environments (Mnih et al., 2013; Nagabandi et al., 2019; Zhang et al., 2018). Agents must reason adaptively and learn continuously (Zhou et al., 2024). This is support by three core capabilities: experience accumulation (Feng et al., 2024) leveraging past interactions, strategic exploration (Zhu et al., 2023) balancing risk and learning, and knowledge abstraction (Zhao et al., 2023; Paglieri et al., 2024) forming high-level representations. Our agent incorporates two modules-Memory and Reasoning (Yao et al., 2023; Ke et al., 2024; Zheng et al., 2025)-to support these core capabilities and enable effective performance in dynamic settings.

9 Conclusion

In this study, we introduce VisEscape, a novel benchmark of escape room scenarios designed to rigorously test multimodal agents' abilities to explore, reason, and make decisions in interactive, dynamic environments. We show that existing multimodal models struggle under these challenging conditions. By integrating memory management and reasoning, we observed that agents could achieve more efficient exploration and reasoning, and noted the crucial synergy between memory and reasoning. Our findings underscore the importance of memory and adaptive reasoning for building more capable, autonomous AI systems in complex and real-world tasks.

10 Limitations

While VisEscape presents a valuable benchmark for testing the exploration-driven problem-solving abilities of multimodal agents, the specific task of room escapes may not fully generalize to real-world environments where agents encounter more complex situations and more unfamiliar objects. Additionally, since the action space of VisEscape is discrete rather than continuous, there is a need to convert the conclusions drawn from multiple modules into feasible continuous action sequences for real-world execution.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and 1 others. 2018. On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757.

Anthropic. The claude 3 model family: Opus, sonnet, haiku.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. arXiv preprint arXiv:2502.13923.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, and 1 others. 2024. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. arXiv preprint arXiv:2412.05271.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, and 1 others. 2019. Textworld: A learning environment for text-based games. In Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7, pages 41–75. Springer.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. Advances in Neural Information Processing Systems, 36:28091–28114.

Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024. On the multi-turn instruction following for conversational web agents. arXiv preprint arXiv:2402.15057.

Yubo Dong, Xukun Zhu, Zhengzhe Pan, Linchao Zhu, and Yi Yang. 2024. Villageragent: A graph-based multi-agent framework for coordinating complex task dependencies in minecraft. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL).

Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. 2024. AGILE: A novel reinforcement learning framework of LLM agents. In Advances in Neural Information Processing Systems (NeurIPS).

Hiroki Furuta, Yutaka Matsuo, Aleksandra Faust, and Izzeddin Gur. 2023. Exposing limitations of language model agents in sequential-task compositions on the web. arXiv preprint arXiv:2311.18751.

Zhe Gan, Linjie Li, Chunyuan Li, Lijuan Wang, Zicheng Liu, and Jianfeng Gao. 2022. Vision-language pre-training: Basics, recent advances, and future trends. arXiv preprint arXiv:2210.09263.

Google DeepMind. 2024. Google gemini ai update: December 2024. Accessed: 2025-05-20.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948.

William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden R Codel, Manuela Veloso, and Ruslan Salakhutdinov. 2019. Minerl: A large-scale dataset of minecraft demonstrations. arXiv preprint arXiv:1907.13440.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 7903–7910.

Kaiyu He, Mian Zhang, Shuo Yan, Peilin Wu, and Zhiyu Zoey Chen. 2024. Idea: Enhancing the rule learning ability of large language model agent through induction, deduction, and abduction. Preprint, arXiv:2408.10455.

Fucai Ke, Zhixi Cai, Simindokht Jahangard, Weiqing Wang, Pari Delir Haghighi, and Hamid Rezatofighi. 2024. Hydra: A hyper agent for dynamic compositional visual reasoning. In European Conference on Computer Vision, pages 132–149. Springer.

Taewoong Kim, Cheolhong Min, Byeonghwi Kim, Jinyeon Kim, Wonje Jeung, and Jonghyun Choi. 2024. Realfred: An embodied instruction following benchmark in photo-realistic environments. In

European Conference on Computer Vision, pages 346–364. Springer.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. arXiv.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and 1 others. 2024a. Llava-onevision: Easy visual task transfer. arXiv preprint arXiv:2408.03326.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, and 1 others. 2021. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. arXiv preprint arXiv:2108.03272.

Zhiyuan Li, Dongnan Liu, Chaoyi Zhang, Heng Wang, Tengfei Xue, and Weidong Cai. 2024b. Enhancing advanced visual reasoning ability of large language models. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 1915–1929, Miami, Florida, USA. Association for Computational Linguistics.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. Advances in neural information processing systems, 36:34892–34916.

Shunyu Liu, Yaoru Li, Kongcheng Zhang, Zhenyu Cui, Wenkai Fang, Yuxuan Zheng, Tongya Zheng, and Mingli Song. 2024. Odyssey: Empowering agents with open-world skills. arXiv preprint arXiv:2407.15325.

Yiting Liu, Liang Li, Beichen Zhang, Shan Huang, Zheng-Jun Zha, and Qingming Huang. 2023b. MaTCR: Modality-aligned thought chain reasoning for multimodal task-oriented dialogue generation. In Proceedings of the 31st ACM International Conference on Multimedia, pages 5776–5785. Association for Computing Machinery.

Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. 2019. Habitat: A Platform for Embodied AI Research. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. 2019. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In International Conference on Learning Representations (ICLR).

Davide Paglieri, Bartłomiej Cupiał, Sam Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. 2024. Benchmarking agentic llm and vlm reasoning on games. arXiv preprint arXiv:2411.13543.

Alexander Pan, Jun Shern Chan, Andy Zou, Nathaniel Li, Steven Basart, Thomas Woodside, Hanlin Zhang, Scott Emmons, and Dan Hendrycks. 2023. Do the rewards justify the means? measuring trade-offs between rewards and ethical behavior in the machiavelli benchmark. In International conference on machine learning, pages 26837–26867. PMLR.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8494–8502.

Cheng Qian, Peixuan Han, Qinyu Luo, Bingxiang He, Xiusi Chen, Yuji Zhang, Hongyi Du, Jiarui Yao, Xiaocheng Yang, Denghui Zhang, and 1 others. 2024. Escapebench: Pushing language models to think outside the box. arXiv preprint arXiv:2412.13549.

Mobina Shahbandeh, Parsa Alian, Noor Nashid, and Ali Mesbah. 2024. Naviqate: Functionality-guided web application navigation. arXiv preprint arXiv:2409.10741.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In International Conference on Machine Learning, pages 3135–3144. PMLR.

Suyeon Shin, Junghyun Kim, Gi-Cheon Kang, Byoung-Tak Zhang, and 1 others. 2024. Socratic planner: Inquiry-based zero-shot planning for embodied instruction following. arXiv preprint arXiv:2404.15190.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In International Conference on Learning Representations.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. Transactions on Machine Learning Research.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11279–11298.

Ziyue Wang, Yurui Dong, Fuwen Luo, Minyuan Ruan, Zhili Cheng, Chi Chen, Peng Li, and Yang Liu. 2025. How do multimodal large language models handle complex multimodal reasoning? placing them in an extensible escape game. Preprint, arXiv:2503.10042.

Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica S Lam. 2021. Grounding open-domain instructions to automate web support tasks. arXiv preprint arXiv:2103.16057.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024a. Qwen2 technical report. Preprint, arXiv:2407.10671.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024b. Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. Advances in Neural Information Processing Systems, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR).

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, and 1 others. 2024. Yi: Open foundation models by 01. ai. arXiv preprint arXiv:2403.04652.

Lijun Zhang, Shuang Lu, and Zhi-Hua Zhou. 2018. Adaptive online learning in dynamic environments. In Advances in Neural Information Processing Systems (NeurIPS), pages 1323–1333.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2023. ExpeL: LLM agents are experiential learners. In Proceedings of the AAAI Conference on Artificial Intelligence.

Bingchen Zhao, Yongshuo Zong, Letian Zhang, and Timothy Hospedales. 2024. Benchmarking multi-image understanding in vision and language models: Perception, knowledge, reasoning, and multi-hop reasoning. arXiv preprint arXiv:2406.12742.

Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2025. Lifelong learning of large language model based agents: A roadmap. arXiv preprint arXiv:2501.07278.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. Webarena: A realistic web environment for building autonomous agents. In The Twelfth International Conference on Learning Representations.

Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. 2024. Symbolic learning enables self-evolving agents. arXiv preprint arXiv:2406.18532.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. arXiv preprint arXiv:2305.17144.

Appendix Overview

- In Appendix A, we provide details about our experimental setup and the prompt template used.
- In Appendix B, we provide additional information about our data construction process.
- In Appendix C, we provide results and analysis for additional experiments that were not covered in the main paper.
- In Appendix D, we provide additional details not extensively covered in the main paper.
- In Appendix E, we briefly introduce and analyze works that address the topic of escape rooms.
- In Appendix F, we present details about the human study conducted.

A Experimental details

A.1 Model List

In the list below, we list the versions of the LLMs used in our experiments. For GPT, Gemini, and Claude models, we use the officially released APIs, and for the others, we use the model versions available on Huggingface.

- Claude-3.5-Sonnet (Anthropic): `claude-3-5-sonnet-20241022`
- GPT-4o (Achiam et al., 2023): `gpt-4o-2024-08-06`
- GPT-4o-mini (Achiam et al., 2023): `gpt-4o-mini`
- gemini-1.5-Pro (Team et al., 2024): `Gemini-1.5-Pro`
- gemini-2.0-Flash (Google DeepMind, 2024): `Gemini-2.0-Flash`
- Qwen2.5-VL-32B-Instruct (Bai et al., 2025): `Qwen/Qwen2.5-VL-32B-Instruct`
- LLaVA-v1.6-34B (Liu et al., 2023a): `llava-hf/llava-v1.6-34b-hf`
- LLaVA-OneVision-Qwen2-7B (Li et al., 2024a): `llava-hf/llava-onevision-qwen2-7b-ov-hf`
- InternVL2-40B (Chen et al., 2024): `OpenGVLab/InternVL2-40B`
- InternVL2.5-38B (Chen et al., 2024): `OpenGVLab/InternVL2_5-38B`
- Qwen2-7B-Instruct (Yang et al., 2024a): `Qwen/Qwen2-7B-Instruct`
- Qwen2.5-32B-Instruct (Yang et al., 2024b): `Qwen/Qwen2.5-32B-Instruct`
- Yi-34B (Young et al., 2024): `NousResearch/Nous-Hermes-2-Yi-34B`
- DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025): `deepseek-ai/DeepSeek-R1-Distill-Qwen-32B`

A.2 MLLMs and Corresponding Backbone LLMs

As shown in Table 7, we used only the LLM that served as backbone language model of corresponding MLLM for controlled experiments. Base LLMs corresponding to each MLLMs used in Table 7 are shown in Table 8. [2]

Model Type	Model Name
VLM	InternVL2.5-38B
Base LLM	Qwen2.5-32B-Instruct
Base LLM (R1)	DeepSeek-R1-Distill-Qwen-32B
VLM	InternVL2-40B
Base LLM	Yi-34B
VLM	LLaVA-v1.6-34B
Base LLM	Yi-34B
VLM	LLaVA-OneVision-7B
Base LLM	Qwen2-7B-Instruct

Table 8: MLLMs and their corresponding base LLMs.

[2] The documents specifying the LLMs selected as language backbone models are as follows:

- InternVL2.5-38B: https://huggingface.co/OpenGVLab/InternVL2_5-38B
- InternVL2-40B: https://huggingface.co/OpenGVLab/InternVL2-40B
- LLaVA-v1.6-34B: https://huggingface.co/llava-hf/llava-v1.6-34b-hf
- LLaVA-OneVision-7B: https://huggingface.co/llava-hf/llava-onevision-qwen2-7b-ov-hf

A.3 Computational Resources and Environments

For closed-source models, we used the OpenAI API for running GPT-4o and GPT-4o-mini, and the Anthropic API for running Claude-3.5-Sonnet.

For open-source models, we downloaded models from Huggingface and served models using the vLLM server. We categorized the models as follows: small models (under 10B parameters), large models (30 to 40B parameters). For small models, we allocated a single NVIDIA RTX 4090 GPU per model. For large models, we allocated two or four NVIDIA A6000 GPUs per model.

All results reported in this paper are aggregated from three different runs conducted in 20 rooms. The time required to complete each room ranged from as little as 10 minutes (for models under 10B parameters, which often terminated early due to the maximum turn limit) to as long as 90 minutes (for Deepseek-R1-Distill-Qwen-32B, which required considerable time for reasoning). For closed-source models, GPT-4o and Claude-3.5-Sonnet cost approximately $2–3 per room for three separate experiments, totaling about $50 for a complete run.

A.4 Hints

To facilitate the evaluation of the agent's progression along its path, all game logic in VisEscape is designed to be sequential. Therefore, the next checkpoint the agent must achieve after its current one is predetermined, as is the hint to guide it there. Hint messages are provided only during Hint-guided experiments if the agent gets stuck for 30 or more steps (i.e., shows no change in checkpoint progression).

Here are two examples of hint messages.

- **Object interaction**: "Use key to unlock the wardrobe at the south wall."
- **Answer for lock**: "Solve the numberlock in the safe at the east wall. The answer is 8056."

A.5 Prompt templates

Table 16, 17, 18, 19, 20 shows experiment prompts.

B Dataset Construction Process

B.1 Asset Creation Using Trimble SketchUp

We use Trimble SketchUp to create object assets (receptacles and items) for the room escape dataset. Receptacles are designed with predefined states, such as open/closed, locked/unlocked, and pushed/pulled, enabling agent interactions. To accommodate scene dynamics, we model these predefined states and generate corresponding receptacles to reflect those states. The visualized examples of receptacles and items are shown in Figure 8.

B.2 Scene Creation Using Autodesk Revit

We employ Autodesk Revit to construct room escape scenes by integrating the assets created in Trimble SketchUp. Autodesk Revit enables the precise placement of receptacles and objects within the scene to simulate realistic escape room environments. Each scene includes multiple viewpoints, which are categorized into three types:

- **Wall View**: A view showing an entire wall on one side of east, west, north, or south. This view updates dynamically according to the states of receptacles and items belonging to each wall.
- **Receptacle View**: A close-up view of a receptacle accessed through the "inspect" action from the Wall View. It changes to reflect the states of receptacles.
- **Item View**: A close-up view of an item accessed through the "inspect" action from the Receptacle View. It provides a close-up perspective of critical elements such as locks, puzzles, and other objects directly involved in problem solving.

B.3 Enhancing Photo-realism Using Chaos Enscape

To enhance the realism of the dataset, we use Chaos Enscape, a rendering tool that processes scene data generated in Autodesk Revit. Chaos Enscape improves the visual fidelity of the scene, providing photorealistic object representations. This increased realism facilitates effective agent reasoning and interaction within the environment, enabling accurate simulation and analysis of escape room scenarios. Refer to Figure 9 to check the before and after application of Chaos Enscape. Visualization of other rooms is in Figure 17.

B.4 Game Logic Design

We used OpenAI's web interface GPT models to design efficient and automated game logic. The authors then manually reviewed the generated game logic design to ensure error-free gameplay. The prompt used for game logic design is in Table 21.

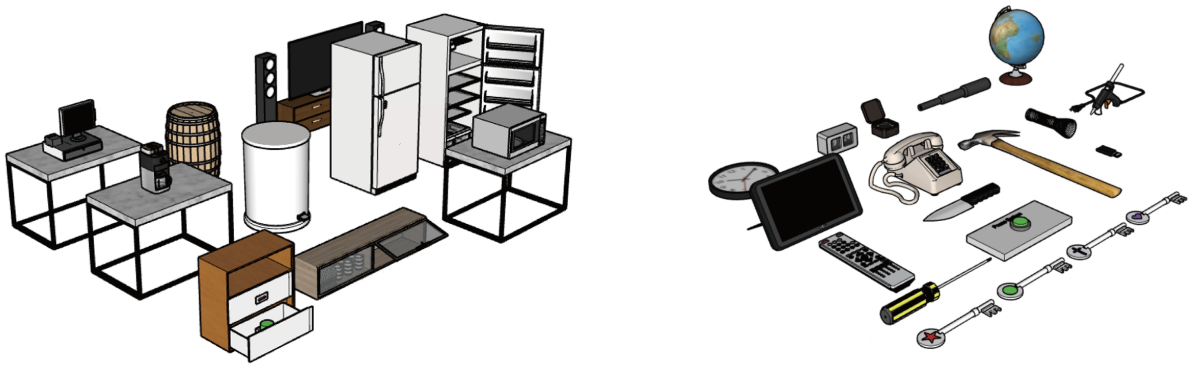


Figure 8: Objects created using Trimble Sketchup for VisEscape. On the left are objects of the *Receptacle* type, and on the right are objects of the *Item* type.

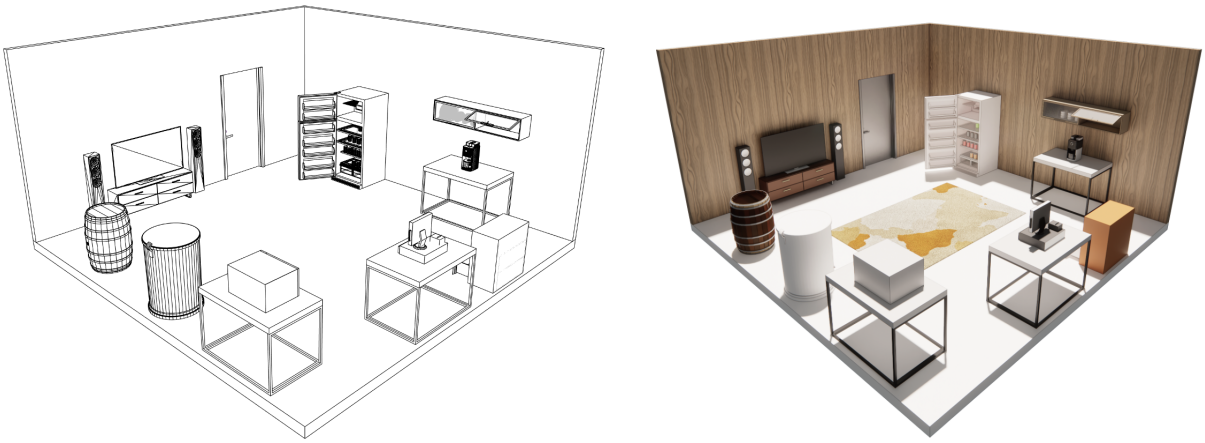


Figure 9: Comparison of applying visual rendering using Chaos Enscape. The left shows the view in Autodesk Revit before using Chaos Enscape, and the right shows the view in Chaos Enscape.

C Additional Experiments and Analysis

C.1 Tendency of Brute-Force Approaches

We found that when attempting to solve visual quizzes, smaller models frequently resorted to brute-force strategies—randomly trying different passwords or sequences without grounding their actions in any meaningful clues. For example, for problems requiring agents to arrange a given set of numbers in the correct order, some agents repeatedly attempted various possible combinations. To verify whether this behavior constituted brute-force attempts, we directly annotated whether the reasoning process and the corresponding attempted-answer pairs indicated a brute-force approach or not. Specifically, we annotated actions according to the following rules: (1) If the reasoning was grounded in meaningful clues or observations, we did not annotate it as brute-force. (2) If the attempted answers involved repeatedly trying different arrangements or combinations without valid justification, then we annotated them as brute-force.

Table 9 demonstrates that while Claude-3.5-Sonnet and GPT-4o typically grounded their puzzle-solving attempts in meaningful visual clues, GPT-4o-mini, InternVL2.5-38B, and LLaVA-v1.6-34B frequently resorted to brute-force approaches. Additionally, models that made a greater effort to identify reasoning justifications for their answers exhibited higher success rates in actual experiments. This behavior partially explains why these lower-performing models consistently achieved lower SPL and step efficiency scores—they made more attempts and failed more frequently on quizzes.

C.2 Reasoning and Visual quizzes

In VisEscape, all visual quizzes require associative thinking abilities. Analyzing game trajectories of diverse models, we find that most failures were on visual quizzes. So, we investigated whether these were problems they inherently couldn't solve, or if they failed because they couldn't properly associate

Model	Brute-force attempt ratio(%)	Solved quiz ratio(%)
Claude-3.5-Sonnet	0.0	21.2
GPT-4o	0.9	25.9
GPT-4o-mini	10.7	6.3
InternVL2.5-38B	10.5	6.6
LLaVA-v1.6-34B	70.0	0.0

Table 9: Results for puzzle-solving. **Brute-force attempt ratio** denotes the ratio of brute-force attempts among all answers submitted for numeric or alphabetic lock puzzles. **Solved quiz ratio** denotes the ratio of correctly solved locks to the total number of locks in VisEscape.

the key clues within the escape room environment. For this, we reformed them as multi-image Visual Question Answering (VQA) problems. By providing the model with both the clue image and the target lock image, we inform them that the two images are related to each other, and then have them solve the problem. Used prompts are in Table 22.

Furthermore, we compare pass@1 and pass@10 to determine if models could correctly solve the task via iterative reasoning, even when their initial attempt failed. To support this repeated hypothesizing and testing, the model's previously generated reasoning sentence for an incorrect answer, along with the incorrect answer itself, was fed back as input for the subsequent attempt.

We selected one visual quiz from each room and then evaluated the performance of each model. Table 10 shows the number of correct answers by each model on these 20 visual quizzes. Results show that open-source models generally demonstrate lower performance (Pass@1 without reasoning) on these visual quizzes compared to frontier models. Also, Reasoning tends to improve the number of correctly solved quizzes across all models.

While frontier models like GPT-4o and Claude-3.5-Sonnet show substantial performance gains with reasoning, the extent of this improvement varies, with some open-source models also showing notable gains, particularly when more attempts are allowed (e.g., InternVL2.5-38B shows a +5 improvement in Pass@10 with reasoning).

Also, iterative reasoning, as indicated by the general increase in scores from Pass@1 to Pass@10 under reasoning conditions, leads to a higher number of correct answers. This suggests that an iterative process of hypothesis formulation and testing enables the models to reach the correct solutions.

Model	Pass@1		Pass@10	
	No Reason	Reason	No Reason	Reason
GPT-4o	9	13 (+4)	9	14 (+5)
Claude 3.5 Sonnet	6	10 (+4)	11	14 (+3)
Gemini-1.5-Pro	7	8 (+1)	9	10 (+1)
GPT-4o-mini	6	7 (+1)	7	8 (+1)
Qwen2.5VL-32B	3	5 (+2)	4	6 (+2)
InternVL2.5-38B	2	2 (+0)	3	8 (+5)
InternVL2-40B	2	2 (+0)	2	5 (+3)

Table 10: Comparison of pass@1 and pass@10 scores for models: performance with and without reasoning.

C.3 Effect of the Exploration Memory

We hypothesize that among the three components of memory management module, particularly the Exploration memory component, which tracks already explored versus unexplored scenes or objects, encourages agents to seek out unseen scenes or objects. To verify this, we conducted an additional ablation experiment in which we specifically removed the Exploration memory, while keeping the other two components (Structured spatial memory and Salient action memory) intact within the Memory module. For both settings, we measured the proportion of scenes observed by the agent from the set of essential observations (as determined by the oracle trajectory) required to successfully complete the game. Figure 10 shows that InternVL2.5-38B exhibited only a slight difference in exploration of unique scenes, whereas LLaVA-v1.6-34B shows a substantial decrease.

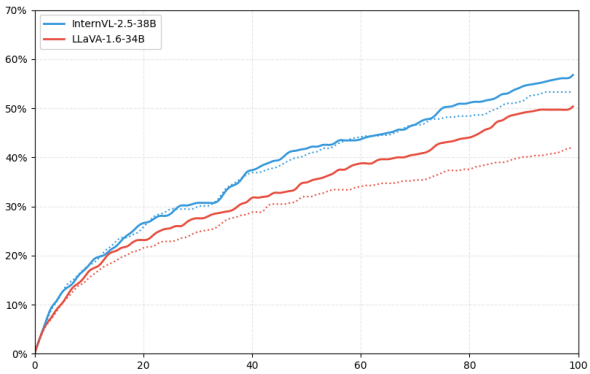


Figure 10: Proportion of essential scenes observed by the agent at least once within the initial 100 steps. Solid and dotted lines denote results with and without exploration memory, respectively. Removing exploration memory significantly reduces the number of observed essential scenes in both models.

C.4 Comparison on Progress

Figure 11 illustrates the progression of goal completion across all rooms during the first 100 steps.

Notably, while Claude-3.5-Sonnet performs best overall, it exhibits a distinctly accelerated trajectory after approximately 30-40 steps. This acceleration indicates that strong models better manage accumulated room-state changes, effectively leveraging previous observations to make strategic and efficient decisions later in the trajectory.

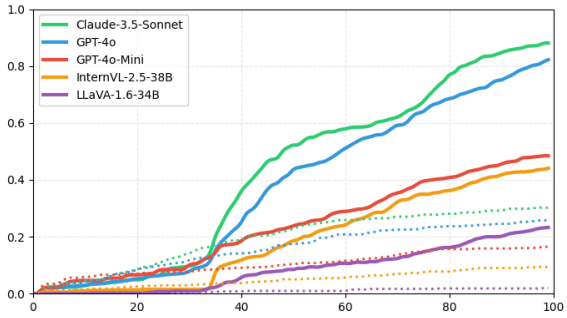


Figure 11: Average GC (Goal Completion) progression across all rooms and experiments during the first 100 steps. The X-axis represents the number of steps, and the Y-axis represents the goal completion at each step. Solid lines represent results from hint-guided experiment, while dotted lines represent those without any hints.

Additionally, models exhibiting rapid progress (Claude-3.5-Sonnet and GPT-4o) used fewer hints compared to other models (low HCR in Table 4). This indicates that rapid progress by high-performing models does not stem from heavy reliance on hints; instead, they use hints selectively only to resolve specific obstacles and, once past these obstacles, accelerate progress by effectively leveraging information accumulated up to that point.

C.5 Repetition

To analyze the performance difference between frontier models and open-source models, we analyzed repetitive actions (repeating the same action in identical situations) exhibited by each model within their trajectories. We observe that lower-performing models frequently repeat actions they have already taken throughout their trajectories, getting stuck in repetitive cycles. Figure 12 shows that these redundant attempts are more prevalent in lower-performing models.

C.6 Statistics for Image Captions

Since the Memory module uses image captions to construct and manage long-term memory, it is crucial to evaluate how accurately the MLLM captures

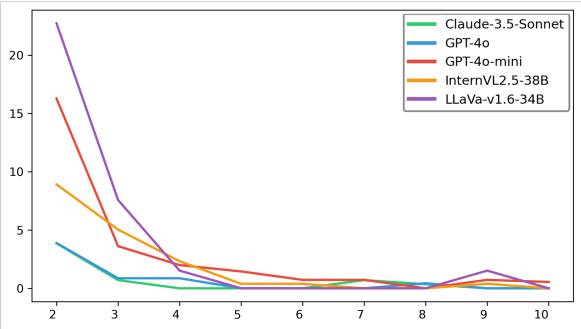


Figure 12: The ratio of actions attempted multiple times (two or more repetitions) within a single trajectory, distributed across the frequency of attempt counts (ranging from 2 to 10+ repetitions). The X-axis represents the number of repeated attempts, and the Y-axis shows the ratio of these attempts out of all actions.

essential information about the current state of the scene. To evaluate captions, we define the ground-truth (GT) state as the complete set of all receptacles and items present in an image obtained from the internal game engine. We then parse the generated captions and examine whether they contain the information consistent with the GT state. Table 11 shows that captions generated by InternVL2.5-38B provide the most accurate information in the most concise way, even when compared to GPT-4o.

Model	Accuracy(%)	Avg. Length
GPT-4o	73.9	47.03
GPT-4o-mini	72.8	64.46
LLaVA-v1.6-34B	63.8	91.19
InternVL2.5-38B	75.1	38.50

Table 11: Evaluation on captions generated by each VLMs. Accuracy indicates whether the caption accurately describes the current game state. Avg. Length denotes the average length of all captions generated by each model for every unique scene.

C.7 Full Results of LLM Experiment

Table 12 includes results for GPT-4o and GPT-4o-mini, which were omitted in Table 7. When GPT-4o-mini used captions instead of images as input, there was a slight improvement across all metrics, but it did not show a significant difference compared to other open-source models. Notably, GPT-4o's performance decreased in exp_{base} but slightly improved in exp_{hint} setting.

C.8 Reasoning Decompsition

The proportions of reasoning components and their composition for other models are shown in Table 14

Model(VLM / LLM)	Input Modality	exp_{base}				exp_{hint}				
		SR(%)	GC(%)	SPL(%)	Step	SR(%)	GC(%)	HCR(%)	SPL(%)	Step
GPT-4o	Image	13.3	29.02	3.91	276.2	95.0	97.39	20.74	26.83	112.9
GPT-4o	Caption	10.0	23.54	3.42	280.7	**98.3**	**98.00**	**17.39**	**33.60**	**91.8**
GPT-4o-mini	Image	6.7	21.31	1.05	291.2	46.7	67.40	32.16	8.93	228.2
GPT-4o-mini	Caption	**11.7**	**24.21**	**3.25**	**280.3**	**53.3**	**73.94**	33.93	**9.76**	**225.6**
InternVL2.5-38B	Image	0.0	6.20	0.0	-	66.7	79.23	41.69	12.23	210.5
InternVL2.5-38B + Base LLM	Caption	1.7	9.57	0.67	295.9	35.0	53.76	38.14	7.08	245.3
InternVL2.5-38B + Base LLM (R1)	Caption	**10.3**	**19.58**	**7.81**	**261.6**	**80.0**	**87.06**	**25.53**	**19.37**	**166.8**
InternVL2-40B	Image	0.0	4.91	0.0	-	8.3	48.18	61.61	0.96	291.1
InternVL2-40B + Base LLM	Caption	0.0	3.88	0.0	-	**32.0**	**68.00**	**48.22**	**4.39**	**250.6**
LLaVA-v1.6-34B	Image	0.0	0.54	0.0	-	15.0	34.79	64.97	1.47	289.8
LLaVA-v1.6-34B + Base LLM	Caption	0.0	**4.14**	0.0	-	**38.3**	**61.80**	**50.08**	**4.59**	**257.7**
LLaVA-OneVision-7B	Image	0.0	1.25	0.0	-	8.3	41.02	62.89	0.80	294.9
LLaVA-OneVision-7B + Base LLM	Caption	0.0	**1.62**	0.0	-	**25.0**	**58.93**	70.70	**2.40**	**287.3**

Table 12: Full results are shown in Table 4. For GPT-4o and GPT-4o-mini, we used the same model but varied the input modality type, since they do not provide language models that receives text input only.

Model	exp_{base}		exp_{hint}	
	Step(*Mean.*)	Step(*Std.*)	Step(*Mean.*)	Step(*Std.*)
Claude 3.5 Sonnet	249.5	47.11	103.1	29.38
GPT-4o	276.2	67.54	112.9	38.59
GPT-4o-mini	291.2	54.82	228.2	45.98
Gemini-1.5-Pro	258.1	84.44	163.3	41.45
Gemini-2.0-Flash	255.0	48.12	179.2	41.97
Qwen2.5VL-32B	295.8	-	197	32.62
InternVL2.5-38B	300.0	12.5	210.5	52.26
InternVL2-40B	300.0	-	291.1	59.85
LLaVA-v1.6-34B	300.0	-	289.8	51.51
LLaVA-OneVision-7B	300.0	-	294.9	43.41

Table 13: Mean and standard deviation of gameplay steps for each model. The standard deviation was calculated using only successful escape trajectories. For models that failed to escape entirely, the standard deviation is marked with "-".

and Figure 13.

D Detailed Explanation

D.1 Reasoning Component

We have classified the reasoning processes that models undertake while solving problems in VisEscape as follows: Observation, Guess, Hypothesis Formation, Planning, and Recall. The classification criteria are as follows:

- **Observation:** Visually perceiving, describing, or analyzing the current scene.
- **Guess:** Making a guess or assumption based on priors or knowledge, without exact evidence acquired in the environment.
- **Hypothesis Formation:** Forming a hypothesis or idea to test in the environment, based on any evidence acquired in the environment.
- **Planning:** Establishing or formulating action plans within the environment.
- **Recall:** Retrieving information from history - remembering the state/location of objects, actions previously performed, and other relevant details.

We extract the pre-action reasoning and action performed by each model at every game step from the experimental logs, which were run with an applied memory management and a reasoning module. And they are annotated using GPT-4o (Achiam et al., 2023). We set temperature as 0.0 for reproduction. To ensure the reliability of our machine-based evaluation method, we sampled 100 instances for each of the categories classified by GPT-4o, totaling 500 samples. These sentences were then manually annotated by humans following the same procedure. Figure 14 shows a heatmap between human and GPT-4o annotations, suggesting that while the model's classifications largely align with human annotations, the model frequently misclassified the *Guess* type as *Hypothesis Formation*.

D.2 Memory Management Module

- **Structured spatial memory**: This memory hierarchically stores information about room walls, the receptacles on those walls, the states of these receptacles, along with their visual information and characteristics. This serves as a directional aid for navigating toward specific receptacles or items.
- **Exploration memory**: This memory stores information about which objects and items have been inspected so far and which have not. This encourages the agent to explore unobserved information rather than getting stuck on the information acquired thus far.

Model	Composition	Ratio	Length
Claude 3.5 Sonnet	*(Obs, Recall, Plan)*	18.1	127.0
	(Obs, Recall, Hypo, Plan)	15.8	141.5
	(Obs, Recall, Plan, Hypo)	14.8	127.5
	(Obs, Recall, Obs, Hypo, Plan)	9.5	155.2
	(Obs, Recall, Hypo)	8.6	125.9
Gemini-1.5-Pro	*(Obs, Plan)*	28.8	28.1
	(Obs, Recall, Plan)	18.6	46.1
	(Plan)	17.4	6.5
	(Recall, Plan)	17.0	32.8
	(Recall, Obs, Plan)	6.0	43.0
GPT-4o	*(Recall, Plan)*	31.6	62.8
	(Recall, Hypo)	13.7	53.9
	(Recall, Hypo, Plan)	11.0	67.4
	(Obs, Plan)	8.8	54.0
	(Recall, Gue, Plan)	8.5	65.5
InternVL2.5-38B	*(Recall, Plan)*	23.3	76.7
	(Plan)	20.9	13.1
	(Recall, Obs, Plan)	11.5	90.7
	(Obs, Plan)	9.6	68.7
	(Obs, Recall, Plan)	8.4	93.6
LLaVA-v1.6-34B	*(Recall, Plan)*	38.5	55.3
	(Plan)	21.3	39.4
	(Obs, Plan)	11.7	54.4
	(Recall, Obs, Plan)	8.9	64.0
	(Obs, Recall, Plan)	6.5	67.0
Qwen2.5VL-32B	*(Obs, Recall, Hypo, Plan)*	16.3	219.0
	(Obs, Recall, Plan)	15.7	143.1
	(Recall, Obs, Hypo, Plan)	15.1	143.1
	(Plan)	12.7	6.4
	(Obs, Plan)	7.9	95.0

Table 14: Reasoning composition for other models.

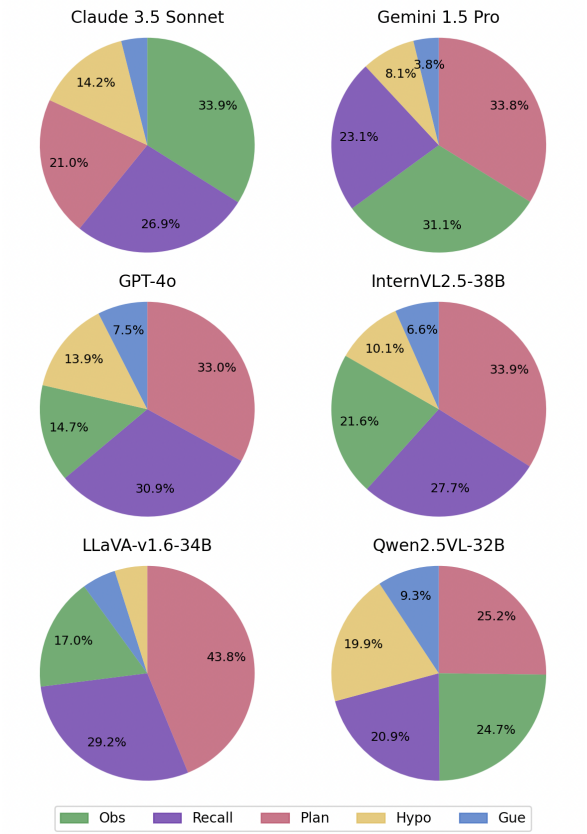


Figure 13: Proportion for each reasoning component for other models.

- **Salient action memory**: This memory selectively stores actions that are worth remembering, such as interactions with objects, attempting to enter passwords, etc. This allows the agent to make action decisions by referencing actions it has performed in the past.

D.3 Visualization of Interactions in VisEscape

Figure 15 is a visualization of interactions defined in VisEscape. The size of each word is proportional to its frequency of appearance within the game.

D.4 Example of Visual Quizzes

Figure 16 shows two examples of visual quizzes in VisEscape.

E Related works on Escape Rooms

EscapeBench (Qian et al., 2024) proposes a benchmark to evaluate creative reasoning in text-only environments, with a focus on tool use as a proxy for measuring creativity. IDEA (He et al., 2024) presents an agent that mimics the loop of human rule-learning by integrating multiple forms of reasoning to guide its actions, focusing on solving

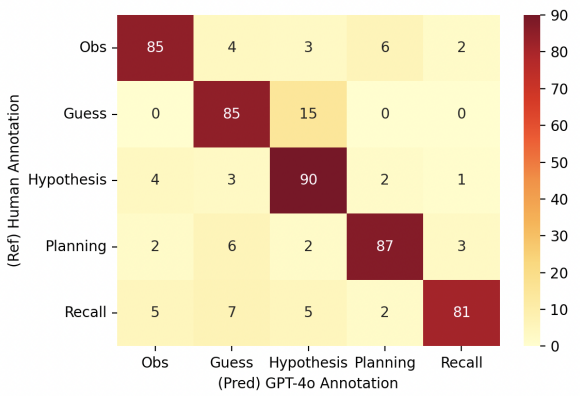


Figure 14: Heatmap of agreement between GPT-4o and human annotations for reasoning type classification. In 85.6% of the instances, the category annotations from GPT-4o and humans were identical.

Figure 15: Wordcloud of all types of actions defined in VisEscape. We excluded "open" and "close", which are available on most receptacles.

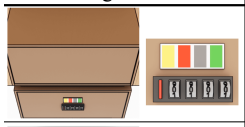
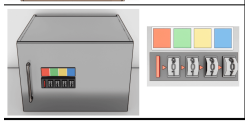
Target Lock	Clue	Explanation
		Count the **number of cans** of each color in the refrigerator. These numbers correspond to each color of the lock. Answer is **[3542]**.
		Count the **number of sides** of each polygon in the image in the desk. These numbers correspond to each color of the lock. Answer is **[8056]**.

Figure 16: Examples of visual quizzes that requires associative thinking in VisEscape. For all problems, we controlled the spatial positioning of the locks and clue images to ensure they remain distinctly separated. This design forces the agent not only to store each piece of information independently but also to identify relevant connections between them (such as "color" in the example above) and reason based on the established associations between these separate pieces of information.

text-based puzzles. MM-Escape (Wang et al., 2025) aims to extend the escape room paradigm by building an embodied environment for the evaluation of MLLM agents. Departing from embodied setups, our visually grounded benchmark blends tool use with puzzle-solving to create a more flexible yet challenging reasoning framework for evaluating MLLMs.

F Human Study

Room	SR(%)	GC(%)	Steps	Duration(s)
1	75.0	92.5	41.8	231.3
2	100.0	100.0	67.3	319.5
3	50.0	87.5	71.1	319.5
4	75.0	93.8	112.0	447.0
5	100.0	100.0	77.3	274.5
6	100.0	100.0	37.0	117.0
7	100.0	100.0	46.8	225.0
8	100.0	100.0	53.0	567.3
9	75.0	90.0	52.0	567.3
10	75.0	91.7	54.0	204.5
11	50.0	79.2	37.8	231.3
12	75.0	92.9	31.3	144.0
13	100.0	100.0	41.5	199.8
14	100.0	100.0	57.3	468.3
15	100.0	100.0	36.0	151.0
16	100.0	100.0	40.8	106.0
17	25.0	96.9	63.3	488.3
18	100.0	100.0	40.5	399.3
19	75.0	91.7	43.8	378.5
20	75.0	100.0	50.8	530.3
Average	82.5	95.8	52.8	318.5

Table 15: Human evaluation per-room success rates, goal completions, step counts, and time durations.

To compare the performance of AI models and human performance on escape room games, we conducted a human study. We recruited a total of 20 participants, including both individuals familiar and those unfamiliar with escape room games, and each was instructed to complete tasks in four different rooms. Figures 18 illustrate the Gradio interface used to provide the UI for playing VisEscape.

Table 15 presents the results of the human evaluation for 20 rooms, including the success rate, goal completion, number of steps, and time duration for experiments. Participants achieved high success rates in most rooms. We paid them based on the minimum hourly wage.

Variables:
direction, available_actions

Initial prompt:
You are an AI agent playing a room escape game. The room is surrounded by 4 walls, and you can explore other walls by "turn_to_[direction]". Each wall has objects that you can interact with, and you can inspect the object by "inspect [object]". Please read the given information and task description below carefully and respond accordingly.

Prompt:
{Initial Prompt}
<Current Observation>{direction} side of room - [IMAGE]</Current Observation>
Based on these information, choose next action to progress in the game. You can do one of the following actions: {available_actions}
Before you act, think first, and then act. Your thought should be in section [THINK], and your action should be in section [ACTION]. In [ACTION], respond ONLY with the chosen action, no other text.
[THINK]
[ACTION]

(a) Prompts for Reasoning module on initial step

Variables:
memory, salient_action_history, action_history, direction, current_scene_desc, previous_react, available_actions, hint_guideline_text

Puzzle text [OPTIONAL: if *ispuzzle* mode]:
<ANSWER> is an action to input the answer to open the lock you are facing. When you choose <ANSWER>, you should follow this format: "<ANSWER>your answer</ANSWER>".

Prompt (Reason):
{Initial Prompt}
<Memory>{memory}</Memory>
<Action Memory>{salient_action_history}</Action Memory>
<Recent actions(from oldest to latest)>{action_history}</Recent actions>
<Current Observation>{direction} side of room - {current_scene_desc}</Current Observation>
<Your Thought and Action before this turn>{previous_react}</Your Thought and Action before this turn>
{available_actions}
{Puzzle text}
Before you act, think first, and then act. If there is a hint message, you should choose action to accomplish the guideline in hint message. {hint_guideline_text}
Your thought should be in section [THINK], and your action should be in section [ACTION]. In [ACTION], respond ONLY with the chosen action or <ANSWER>your answer</ANSWER>, no other text.

Prompt (Retry):
{Initial Prompt}
<Memory>{memory}</Memory>
<Your Previous Action> {before_action}
<Available Actions> {available_actions}
You just performed the <Your Previous Action>, but that action is not currently available in <Available Actions>. Referring to your memory, choose an action that is necessary to perform <Your Previous Action>.
{hint_guideline_text}
You should choose one of the following actions:{available_actions}
Please respond following below format without any other text: [ACTION]

(b) Prompts for Reasoning module and Retry

Table 16: Prompts for the Reasoning module-before action.

Variables:
action_history

Initial prompt:
You are an AI agent playing a room escape game. The room is surrounded by 4 walls, and you can explore other walls by "turn_to_[direction]". Each wall has objects that you can interact with, and you can inspect the object by "inspect [object]". Please read the given information and task description below carefully and respond accordingly.

last_10_history (for log in action_history):
Observation: [{log['scene']}]\n
Action: [{log['action']}]-{log['analysis']}

spatial_json_format:
```
{"direction 1" : {
   "objects":["object1", "object2", ...]
},
...}
```

inspected_objects_json_format:
```
[{"object 1" : {
   "state":"",
   "characteristics":"",
   "additional info":""
, ...}]}
```

Prompt:
{Initial Prompt}
<Last 10 logs(from oldest to latest)> {last_10_history} </Last 10 logs>
Here is the definition of information:
<Last 10 logs>: Sequence of observation, action-effect, next observation, next action-effect for each turn.
Here is the task:
Construct your memory about the room, based on the last 10 logs. Follow below guidelines:
1. Identify which objects exist on each directional wall, and add the information to "[SPATIAL MEMORY]" section.
2. If you have inspected an object via "inspect [object]", you should add the information to "[INSPECTED OBJECTS]" section.
3. The objects that you have not inspected via "inspect [object]" should be added to "[UNINSPECTED OBJECTS]" section.
4. Add any information from observations which is not included in [SPATIAL MEMORY] and [INSPECTED OBJECTS] that you think is necessary for solving other problems to the "[Additional Memory]" section.
Please respond following below format without any other text:
[SPATIAL MEMORY] {spatial_json_format}
[INSPECTED OBJECTS] {inspected_objects_json_format}
[UNINSPECTED OBJECTS] []
[ADDITIONAL MEMORY] [1. additional memory1, 2. additional memory2, ...]

Table 17: Prompts for memory management module - for first run (memory construction).

Variables:
spatial_memory, action_history

Initial prompt:
You are an AI agent playing a room escape game. The room is surrounded by 4 walls, and you can explore other walls by "turn_to_[direction]". Each wall has objects that you can interact with, and you can inspect the object by "inspect [object]". Please read the given information and task description below carefully and respond accordingly.

last_10_history (for log in action_history):
Observation: [{log['scene']}]\n
Action: [{log['action']}]-{log['analysis']}

spatial_json_format:
```
{"direction 1" : {
   "objects":["object1", "object2", ...]
},
...}
```

inspected_objects_json_format:
```
[{"object 1" : {
   "state":"",
   "characteristics":"",
   "additional info":""
, ...}]}
```

Prompt:
{Initial Prompt}
<Current Memory> {spatial_memory} </Current Memory>
<Last 10 logs(from oldest to latest)> {last_10_history} </Last 10 logs>
Here is the definition of each information:
<Last 10 logs>: Sequence of observation, action-effect, next observation, next action-effect for each turn.
Here is the task:
Update your memory about the room, based on the last 10 logs. Follow below guidelines:
1. If you newly inspected an object via "inspect [object]" among objects in "[UNINSPECTED OBJECTS]" section,
  you should add the information to "[INSPECTED OBJECTS]" section.
2. Based on the information that you can obtain from last 10 logs, update <Current Memory>
  if there exists any information that you can obtain from <Last 10 logs> but not in <Current Memory>.
3. Add any information not included in spatial memory and inspected objects that you think is necessary
  for solving other problems to the "[ADDITIONAL MEMORY]" section.
Please respond following below format without any other text:
[SPATIAL MEMORY] {spatial_json_format}
[INSPECTED OBJECTS] {inspected_objects_json_format}
[UNINSPECTED OBJECTS] []
[ADDITIONAL MEMORY] [1. additional memory1, 2. additional memory2, ...]

Table 18: Prompts for Memory management module after memory construction

Variables:
previous_scene, previous_action, current_scene

Prompt:
{Initial Prompt}
<Previous Observation> : {previous_scene}
<Previous Action> : {previous_action}
<Current Observation> : {current_scene}
Here is the definition of each information:
<Previous Observation> is a description of a scene before your action, and <Current Observation> is a description of a scene after your action.
Here is the task:
Analyze the effect of your action by comparing <Previous Observation> and <Current Observation>.
The analysis should be concise and definitive, not descriptive.
Keep it under 10 words.
[ANALYSIS]

Table 19: Prompts for Reasoning module-after action.

Variables:
`item_name`

Prompt:
```
This image is a close-up view of an item '{item_name}'.
Describe this image. Your description should fulfill the following rules:
1. Description should include every visual information, but concise and clear.
2. Do not start description with phrases like 'The image depicts', 'The image shows', etc.
```

(a) Prompts for getting caption from image of item-view

Variables:
`object_type, items_str`

```
This image is a close-up view of an object '{object_type}'.
In {object_type}, the following objects are present: {items_str}
Describe this image. The names of visible objects should be expressed using the given object names
above, enclosed in "".
Your description should fulfill the following rules:
1. Description should include every visual information, but concise and clear.
2. Do not include any analysis of the scene or the room, just describe the image.
3. Do not start description with phrases like 'The image depicts', 'The image shows', etc.
```

(b) Prompts for getting caption from image of object-view

Variables:
`objects`

```
This image is a wall view of a room, with following objects: {objects}.
Describe this image. The names of visible objects should be expressed using the given object names
above, enclosed in "".
Your description should fulfill the following rules:
1. Description should include every visual information, but concise and clear.
2. Do not include any analysis of the scene or the room, just describe the image.
3. Do not start description with phrases like 'The image depicts', 'The image shows', etc.
```

(c) Prompts for getting caption from image of wall-view

Table 20: Prompts for captioning observations.

Prompt:

You are a game designer proposing ideas for an 'Escape Room Game'. I want to create an 'Escape Room Game' in a room structure enclosed by four walls. Inside the room, there are various objects, and it's an 'Escape Room Game' where actions initiated from a specific object trigger other objects, allowing puzzles to be solved one by one. There are two types of objects:

1. Receptacle:
- Definition) Objects that can contain or hold other objects, such as tables, drawers, or wardrobes.
- Each Receptacle is visually distinct and has unique States. For example, a door without a lock has two States: 'open' and 'closed', but a door with a lock has three States: 'locked', 'open', and 'closed'. Also, a drawer with two compartments can have four States depending on the open/closed status of each compartment: 'opened_opened', 'opened_closed', 'closed_opened', 'closed_closed'.
- Each Receptacle must be assigned to one of the four walls of the room.
- Receptacles can contain Items. However, considering realistic constraints, each Item must define the "Interactable State(s)" of the Receptacle where it can be interacted with (i.e., visible). For example, if you have a key and a wardrobe with States defined as [locked, closed, open], the key's interactable state for that wardrobe is [open]. Interactable States can be multiple. For instance, for a drawer with two compartments, if a key exists in the second compartment, the key's interactable states are [closed_open, open_open].

2. Item:
- Definition) Objects that can be triggers for interaction with specific Receptacles and other Items, such as keys, buttons, photos, etc. Items must be contained within a Receptacle. There are two types of Items:
- 2-a. Applicable Item: Items that can be picked up, stored in an inventory, and applied to other objects, such as keys or ID cards.
- 2-b. Quiz Item: Items that contain a puzzle, which, when correctly solved or triggered by a specific action, activates a change in another specific object within the game.

Furthermore, in addition to objects, the game requires internal logic for escaping the room. For example, a wardrobe is initially locked, but finding and applying a key from elsewhere unlocks it. A drawer is initially locked, but solving a quiz elsewhere opens it. The game logic can be very creative and diverse, but it must be defined according to the following constraints:

1. It must be limited to logic that changes the state of a Receptacle.
2. Actions related to game logic must be associated with an Item. For example, changing a drawer from a 'locked' State to a 'closed' State via an 'apply Key' action is game logic. However, changing a drawer from an 'opened' State to a 'closed' State via a 'close' action is not game logic.

If these two constraints are met, the in-game logic can be expressed atomically in the following format:
(receptacle_name.state_name_before_action, action_name(item_name),
receptacle_name.state_name_after_action, checkTrigger(interaction_name))

In summary, to design one escape room game, the following game components are needed:
- Approximately 5 Receptacles, all possible states defined for each object, and which wall they belong to.
- Approximately 5 Items, and which Receptacle they belong to.
- Approximately 5 game logic interactions, expressed in the atomic format described above.

Please design a game logic with following receptacles and items:
[Receptacle] - 2TierDrawer, Cupboard, Safe, Wardrobe, Desk, Door
[Receptacle] - Key1, key2, ID card, button, puzzle
Design an escape room game by creating interactions between objects and puzzles. Storytelling is not required, and the game does not need to be particularly creative or innovative.

Table 21: Prompts used for game logic design.

Figure 17: Example of rooms rendered using Chaos Enscape.

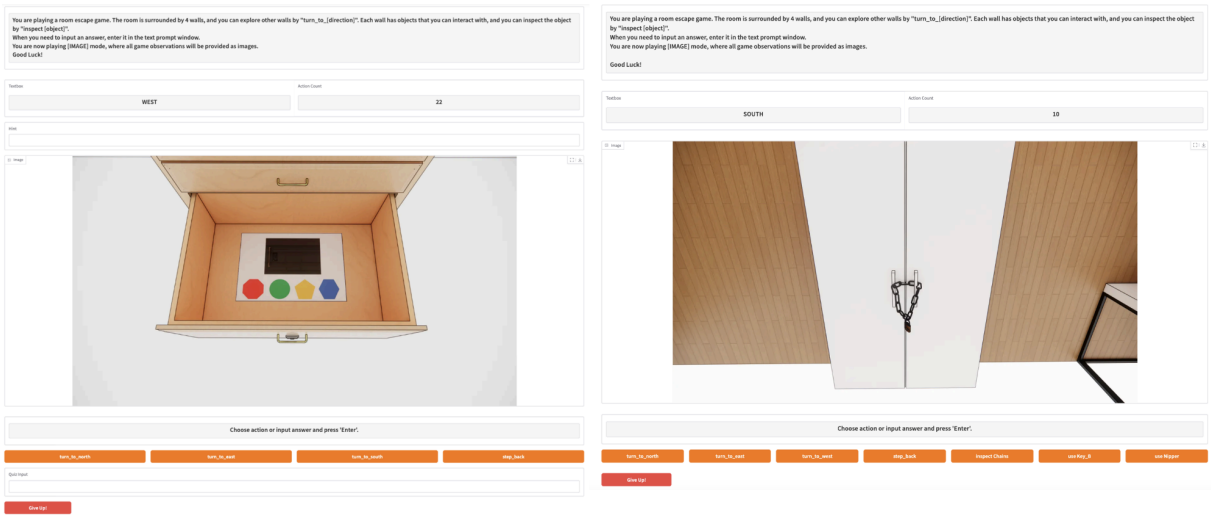


Figure 18: User Interface generated by Gradio for conducting experiments for human trajectory.

Variables:
image1, image2, [Optional]tried_answers

{History} (if *tried_answers* is given):
You have tried the following answers: {tried_answers}, but they are all wrong.

Prompt:
{image1} is a hint image for the password of the lock described in {image2}.
You should guess the password of the lock solely based on the hint image.
{History}
Respond ONLY with the your answer, no other text.

(a) Prompts for multi-image VQA without reasoning.

Variables:
image1, image2, [Optional]tried_answers

{History} (if *tried_answers* is given):
You have tried the following answers: {tried_answers}, but they are all wrong.

Prompt:
{image1} is a hint image for the password of the lock described in {image2}.
You should guess the password of the lock solely based on the hint image.
{History}
Before you guess, think first, and then guess the password. Your thought should be in section [Think], and your answer should be in section [Answer]. In [Answer], respond ONLY with the your answer, no other text.
[Think]
[Answer]

(b) Prompts for multi-image VQA with reasoning.

Table 22: Prompts for multi-image VQA.

Variables:
think, action

Prompt:
You are tasked with analyzing the logs of an AI agent in an escape room game.
I'll give you a think-action pair that the AI agent has performed.
Your task is as follows:

1. Divide the agent's reasoning chain into distinct units that fall into the categories explicitly given below:
 - Observation: visually perceiving, describing, or analyzing the current scene
 - Guess: making a guess or assumption based on priors or knowledge, without exact evidence acquired in the environment
 - Hypothesis Formation: forming a hypothesis or idea to test in the environment, based on any evidence acquired in the environment
 - Planning: Establishing or formulating action plans within the environment
 - Recall: Retrieving information from history - remembering the state/location of objects, actions previously performed, and other relevant details

A unit could be:
 - A single sentence
 - Multiple sentences together
 - Multiple units within a single sentence

2. If a unit does not fit into the above categories, create a new category called [None-<new category name>] and place it there.
3. List all units in the same order as they appear in the original text. Do not reorder or rearrange the units.

Input:
[Think]{think}
[Action]{action}

Response Format:
[Category of Unit 1]: Sentence or phrase of given [Think]
[Category of Unit 2]: Sentence or phrase of given [Think]
[Category of Unit 3]: Sentence or phrase of given [Think]
...
Response:

Table 23: Prompts for annotating components of reasoning.