# Towards Generalizable Visually Grounded Exploration of Household Devices

**Linhao Zheng[1*], Zeming Liu[2*], Wangke Chen[1], Li Zeng[1], Wanxiang Che[3], Heyan Huang[1], Yuhang Guo[1†]**
[1]School of Computer Science and Technology, Beijing Institute of Technology
[2]School of Computer Science and Engineering, Beihang University
[3]Research Center for Social Computing and Interactive Robotics, Harbin Institute of Technology
zhenglinhao@bit.edu.cn zmliu@buaa.edu.cn guoyuhang@bit.edu.cn

## Abstract

Recent advancements in Vision-Language Models (VLMs) have demonstrated impressive capabilities in static visual recognition and high-level semantic reasoning. However, current embodied exploration paradigms still heavily rely on imitation learning from human-annotated trajectories, which severely limits agents' generalization ability. The key bottleneck of realizing general autonomous embodied agents lies in *Generalizable Visually Grounded Exploration*—the ability to operate novel devices without manuals or specific training by actively grounding abstract world knowledge into fine-grained visual affordances. Yet, existing benchmarks fail to evaluate this capability: they generally rely on explicit documents and annotated trajectories, neglecting the dynamic *Hypothesis-Interaction-Refinement* process essential for functional device operation. To bridge this gap, we introduce **VGEBench**, a comprehensive benchmark designed to evaluate the generalizable visually grounded exploration capabilities of VLMs. Unlike static datasets, we construct a Logic-Driven State Machine framework. This framework simulates multi-turn interaction loops, compelling agents to achieve goals by active visual perception and feedback-driven correction. Experimental results demonstrate that existing VLMs face significant challenges in translating semantic knowledge into physical execution and maintaining long-horizon state tracking. [1]

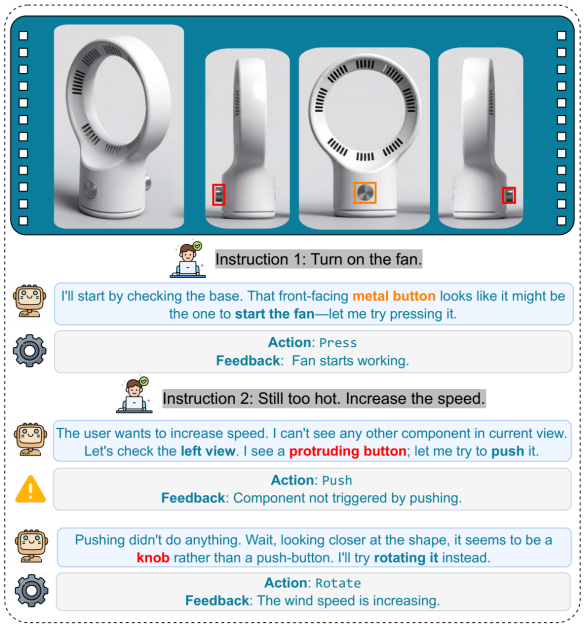



Figure 1: An example of **Generalizable Visually Grounded Exploration**. The example demonstrates a **Hypothesis-Interaction-Refinement** loop: after failing to increase the speed by pushing a component, the agent interprets the physical feedback, updates its hypothesis, re-evaluates the visual features to identify the component as a rotatable knob, and successfully corrects its action.

## 1 Introduction

The ultimate goal of Artificial General Intelligence (AGI) is to develop embodied agents capable of assisting humans by seamlessly interacting with the devices in their daily lives (Duan et al., 2022; Wang et al., 2023; Driess et al., 2023). From heating food in a microwave to adjusting a standing fan, these activities constitute the fundamental fabric of daily living. While humans can effortlessly manipulate a never-before-seen device without consulting a manual, this ability remains a formidable challenge for current AI systems. Humans achieve this proficiency not by memorizing the operation of every specific device instance, but through *generalizable exploration* grounded in general world knowledge. When encountering a novel device, a human forms a hypothesis based on affordance perception (e.g., "the ❄ symbol might represent temperature control") (Qian et al., 2024), interacts with it, and refines their understanding based on physical feed-

[*]Equal contribution
[†]Corresponding author: guoyuhang@bit.edu.cn
[1]Dataset and codes are publicly available at https://github.com/BITHLP/VGEBench



*Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 32511–32541
October 24-29, 2026 ©2026 Association for Computational Linguistics

back (e.g., "turning it left didn't work, so I must turn it right") (Bohg et al., 2017). This dynamic process of **Hypothesis-Interaction-Refinement** is the cornerstone of generalizable manipulation.

However, current research in embodied AI and Large Language Models (LLMs) has yet to fully capture this capability. On one hand, traditional robotic learning benchmarks typically rely on Imitation Learning or Reinforcement Learning (Tao et al., 2024; Li et al., 2024; Shridhar et al., 2021), requiring agents to be trained on specific environments, which limits their ability of generalization to unseen devices (Liang et al., 2022). On the other hand, LLM-based tool learning (Schick et al., 2023; Qin et al., 2023) has achieved success in software domains, but primarily operates on structured APIs with explicit documentation. These paradigms bypass the core challenge of the physical world: the need to bridge the gap between abstract common sense and instance-specific physical interfaces through active visual exploration and feedback reasoning.

To bridge this gap, we introduce **VGEBench**, a novel benchmark designed to evaluate the *Generalizable Visually Grounded Exploration* capabilities of Vision Language Models (VLMs) on household devices. Unlike previous benchmarks that focus on static visual recognition or rote execution, VGEBench simulates a high-fidelity, documentation-free environment where agents must figure out how to use 968 diverse devices across 26 categories solely through visual perception and interaction. Our setting poses unique challenges: (1) **Knowledge Grounding**: The agent must map abstract knowledge (e.g., "fans oscillate") to fine-grained, unlabelled visual affordances (e.g., a specific mechanical pin); (2)**Visual Grounding**: The agent must transcend static recognition to precisely localize interactive components, serving as the spatial prerequisite for valid physical interaction; (3) **Feedback-Driven Reasoning**: The agent must interpret diverse environmental feedback—ranging from state changes (lights turning on) to failures (pressing a knob)—to correct its plans in real-time.

Our main contributions are summarized as follows:

- We identify *Generalizable Visually Grounded Exploration* as a critical missing link for VLM agents in operating household devices, revealing a critical gap between abstract world knowledge and instance-level affordances.
- To bridge this gap, we present VGEBench, a novel benchmark which features a Logic-Driven State Machine framework that supports deterministic, multi-turn interactions, enabling the evaluation of long-horizon generalizable exploratory reasoning.
- We establish a multi-dimensional evaluation protocol and conduct extensive experiments on various VLMs, revealing their limitations in visual grounding, generalizable exploration, and long-horizon state tracking. Our results show that existing models struggle to translate general world knowledge into precise physical interaction in multi-turn interaction scenarios.

## 2 Related Work

**From API to Physical Interaction.** Large language models have demonstrated strong tool-use capability in language-centric settings, where tools are accessed through structured API interfaces (Schick et al., 2023; Qin et al., 2023). While advanced reasoning frameworks enhance decision-making through iterative thought traces (Yao et al., 2023; Shinn et al., 2023), these agents largely operate under a *manual-dependent* paradigm (Patil et al., 2025; Li et al., 2025), relying on explicit documentation or schema definitions provided in the prompt rather than discovering functionality through exploration (Cheng et al., 2025). Moving to operating systems, agents are required to navigate GUI environments (Xie et al., 2024; Rawles et al., 2024). While they support open-ended tasks, their perception is often simplified by accessing the accessibility tree (e.g., DOM/XML), and their interaction is confined to 2D screen spaces, lacking the *multi-view* active perception required in the 3D physical world.

**Fine-grained Visual Grounding.** Operating physical devices not only requires an understanding of the device's functionality but hinges on fine-grained visual grounding of interactive components (e.g., knobs, ports, or mode buttons). Traditional VQA-style benchmarks have advanced holistic vision-language understanding (Antol et al., 2015; Hudson and Manning, 2019; Marino et al., 2019) and physical object reasoning (Ma et al., 2025; Zhang et al., 2025), yet they largely evaluate *static* question answering, without explicitly modeling

| Benchmark | Exploration Capabilities | | | Functional Logic | | Vision | | | Domain |
|---|---|---|---|---|---|---|---|---|---|
| | GE | IFR | EC | DLM | MF | PE | FG | MV | |
| BFCL (Patil et al., 2025) | ✔ | ✔ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | API/Code |
| PhysToolBench (Zhang et al., 2025) | ✔ | ✗ | ✗ | ✗ | ✔ | ✔ | ✗ | ✗ | Physical Tools |
| OpenEQA (Majumdar et al., 2024) | ✔ | ✔✗ | ✗ | ✗ | ✔ | ✔ | ✗ | ✔ | Indoor Scenes |
| OSWorld (Xie et al., 2024) | ✔ | ✔✗ | ✔ | ✔ | ✔✗ | ✗ | ✔ | ✗ | Computer GUIs |
| AndroidWorld (Rawles et al., 2024) | ✔ | ✔ | ✔ | ✔ | ✔ | ✗ | ✔ | ✗ | Android Apps |
| HomeBench (Li et al., 2025) | ✗ | ✗ | ✗ | ✔ | ✗ | ✗ | ✗ | ✗ | Virtual Smart Home |
| RoboCasa (Nasiriany et al., 2024) | ✗ | ✗ | ✔ | ✗ | ✔ | ✔ | ✗ | ✔ | Kitchen Tasks |
| BEHAVIOR-1k (Li et al., 2024) | ✗ | ✔ | ✔ | ✗ | ✔ | ✔ | ✗ | ✔ | Household Tasks |
| **VGEBench (Ours)** | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | Household Devices |

Table 1: Comparison of VGEBench with existing related benchmarks. **Exploration: GE (Generalizable Exploration)**: training-free generalization without environment-specific RL/IL finetuning; **IFR (Interactive Feedback & Refinement)**: Supporting multi-turn hypothesis testing; **EC (Error Correction)**: Environment can support multi-turn trial-and-error. **Logic: DLM (Device Logic Modeling)**: Providing explicit device state and transition logic; **MF (Manual-Free)**: Relying on world knowledge and environmental feedback rather than provided documentation. **Vision: PE (Physical Entity)**: Targeting 3D functional appliances with realistic topology; **FG (Fine-grained Grounding)**: Localizing specific interactive parts (e.g., buttons) vs. coarse objects; **MV (Multi-View)**: Active perception across multiple viewpoints. ✔: Fully Supported; ✔✗ : Partially Supported; ✗: Not Supported.

actionable component localization under interaction constraints (You et al., 2023). Although there has been progress on linking language to specific regions or instances through referring expression comprehension (Yu et al., 2016; Liu et al., 2024) and grounded multimodal interfaces (Peng et al., 2023; Chen et al., 2023), there remains a practical granularity mismatch in current VLMs: models can be robust in holistic recognition but brittle when correct actions depend on small, functionally meaningful components that are easy to confuse across instances and viewpoints (Geng et al., 2022).

**Embodied Exploration.** Beyond knowing where to act, operating household devices requires anticipating what will happen when acting, with actions ultimately validated through observable state changes and feedback. Recently, benchmarks like OpenEQA (Majumdar et al., 2024) and PEAP (Lan et al., 2026) have focused on assessing embodied perception. However, they primarily evaluate passive observation or static functional reasoning rather than active, state-changing manipulation. In physical environments, traditional benchmarks (e.g., ALFWorld (Shridhar et al., 2021), BEHAVIOR (Li et al., 2024)) have established high-fidelity simulation environments for common household tasks. However, they typically rely on training agents within **specific simulators**—often via **Imitation Learning** on extensive **human action trajectories** (Savva et al., 2019). This paradigm inherently limits their generalizability to novel instances outside the training distribution. The central challenge remains **generalizable exploration**: enabling agents to map abstract world knowledge to instance-specific affordances without prior training or manuals. Unlike previous works that focus on either text-heavy reasoning or rote skill execution, VGEBench targets the intersection of visual grounding, active logic probing, and feedback-driven refinement.

## 3 VGEBench

To accurately evaluate the generalizable visually grounded exploration capabilities of VLMs in household environments, we constructed VGEBench, a large-scale benchmark of diverse household devices with fine-grained component annotations and executable interaction logic. As shown in Fig. 2, the data construction pipeline consists of three main stages: raw view collection, state machine annotation, and instruction generation.

### 3.1 Raw View Collection

To accurately evaluate fine-grained manipulation capabilities (e.g., rotating a specific knob or toggling a small switch), high-fidelity visual observations are essential. Traditional 3D model datasets (e.g., ShapeNetCore (Chang et al., 2015)) often lack the texture resolution or distinct functional details required for such precise interactions. Motivated by this gap, we sourced high-quality 3D models that support legible component identification

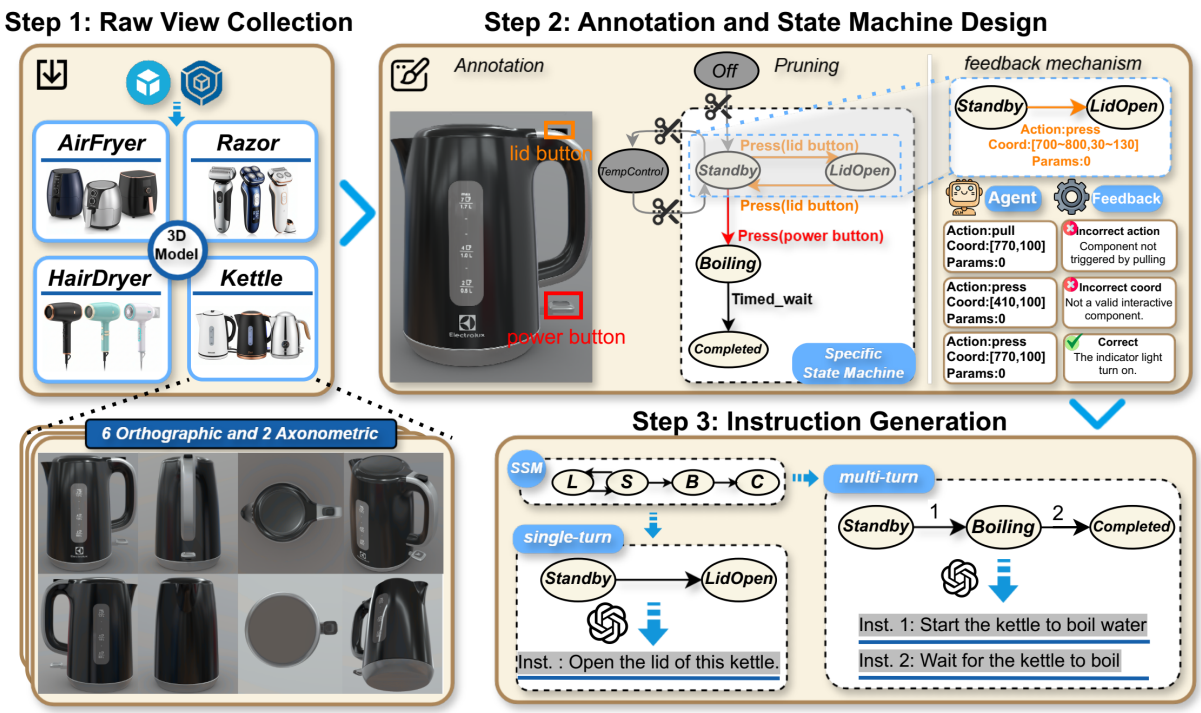



Figure 2: **The Data Construction Pipeline of VGEBench:** This pipeline consists of three main phases: raw view collection, Annotation and state machine design, and instruction generation. In the **Raw View Collection** phase, we acquire high-fidelity visual observations, capturing views for each 3D object to ensure comprehensive coverage. The process then proceeds to **Annotation and State Machine Design**, where a Universal Category State Machine (UCSM) is instantiated into a Specific State Machine (SSM) based on manual annotations of visible interactive components. Building on this SSM, our environment simulator can generate state-dependent feedbacks. Finally, **Instruction Generation** synthesizes diverse tasks by sampling valid state transition paths from the SSM and refining them into natural language instructions via LLMs.

from Sketchfab[2] and 3D Warehouse[3]. Utilizing the platforms' built-in 360-degree model preview features, human annotators captured 8 multi-view images per object (6 orthographic and 2 axonometric). We also conducted a rigorous manual verification for every candidate model to ensure the presence of clearly identifiable interactive components. We finally collected 968 high-fidelity 3D models across 26 common categories (e.g., CoffeeMachine, DigitalAlarmClock, WashingMachine), totaling 7,888 collected views. A detailed distribution is provided in Tab. 7.

### 3.2 Annotation and State Machine Design

We propose a logic-driven annotation protocol to ground visual perception into executable plans. First, human annotators labeled the bounding boxes of all *interactive components* (e.g., buttons, knobs) visible across the collected views. To govern the interaction logic, we designed a **Universal Category State Machine (UCSM)** for each category, which serves as a superset containing all potential functionalities within that category. Subsequently, we derived a **Specific State Machine (SSM)** for each individual object by pruning the UCSM based on the visual evidence of the object's components. For instance, if a specific weighing scale visually lacks a charging port, the corresponding charging states and transitions are removed from its SSM. This process yields a rigorous logic graph for every device, where transitions are bound to atomic actions and parameters. Component names are used only for internal state tracking and are not exposed to the model, ensuring the agent relies on visual features rather than textual shortcuts.

Notably, the generalization evaluated by VGEBench refers not to appearance-level novelty in static image recognition, but to *functional and interactive generalization*: whether an agent can select actions and parameters based on visual components and revise incorrect hypotheses through environment feedback. Since the 3D assets are sourced

[2] https://sketchfab.com/
[3] https://3dwarehouse.sketchup.com/

from public websites, we cannot guarantee that closed-source VLMs have not encountered related assets or renderings during pretraining. However, the manually constructed Logic-Driven State Machines define instance-specific component-action bindings and state-transition rules that are not provided by the static visual assets. Thus, even if a model has seen images of similar devices, it cannot directly obtain the instance-specific interaction logic evaluated by our simulator from those images alone.

### 3.3 Instruction Generation

Based on the validated SSMs, we generated instruction-following tasks divided into single-turn and multi-turn scenarios. We performed pathfinding on the SSMs starting from a manually aligned initial state to generate state transition sequences that avoid loops. To simulate realistic user commands, we converted these symbolic sequences into natural language instructions using GPT-5-mini. In this process, the LLM is used only to verbalize a symbolic state-transition sequence whose initial state, target state, and valid transition path are determined by the validated SSM. For example, a symbolic transition `Off` $\rightarrow$ `Active` is refined into natural commands like "Turn on the flashlight." In total, we constructed 4,948 single-turn and 10,005 multi-turn instructions. The detailed statistics of VGEBench and prompts used for instruction refinement can be found in the Appendix A and Tab. 20.

To examine whether the evaluation is sensitive to the choice of instruction generator, we regenerate instructions using additional generators and evaluate the resulting variants. The results show that VGEBench is robust to the choice of generation model. Detailed results are provided in Appendix E.3.

### 3.4 Environment Feedback and Task Evaluation

Inspired by ScienceWorld (Wang et al., 2022), we designed a closed-loop interactive system where the instantiated Specific State Machine (SSM) serves as the ground truth environment simulator. Unlike static evaluation datasets, our system evaluates the agent's ability to navigate through a state space by executing a sequence of visual actions. The interaction mechanism consists of two core components: the Feedback Generator and the Completion Judge.

**Deterministic Feedback Generation** At each time step $t$, the agent receives the current observation $O_t = (I_{view}, V_{list}, F_{t-1}, Inst_t)$, and executes an action $a_t = (\text{type}, \text{coord}, \text{params})$. Here, $I_{\text{view}}$ represents the image of the current view, $V_{\text{list}}$ denotes the set of accessible views, $F_{t-1}$ is the feedback from the environment regarding the previous action, and $\text{Inst}_t$ denotes the current user instruction. The environment simulator validates this action against the current state $S_t$ defined in the SSM. To simulate realistic physical constraints and provide instructive feedback, the system employs a hierarchical validation protocol:

**Validity Check:** The system first verifies if the action is permissible in the current view (e.g., physical interactions are prohibited in axonometric views like *front_top*, requiring a `Switch_view` action first) and whether the coordinates fall within the image bounds. Invalid actions trigger immediate "System Notification" errors without altering the state.

**Geometric & Semantic Matching:** If the action format is valid, the system queries the transitions defined in $S_t$. An action is considered successful only if:

**(1) coord:** The coordinates fall within the bounding box of a reactive component defined in the SSM for the current view.

**(2) action:** The action type matches the component's defined atomic action.

**(3) params:** The parameters satisfy the transition conditions (e.g., rotation direction).

**Environment Feedback:** To facilitate effective interaction and error recovery, the environment generates outcome-dependent feedback for each agent action. The feedback differentiates among qualitatively different interaction outcomes, enabling the agent to infer whether an action was successful, misapplied, or irrelevant.

**Task Completion Criteria** A task is defined not merely by reaching a final state, but by satisfying a sequence of sub-goals. Each task instance is associated with an ordered list of goal states $G = [g_1, g_2, \ldots, g_n]$. The agent can only observe the user instruction corresponding to the next goal after completing the current goal. To limit resource consumption and prevent infinite exploration loops, we further control the interaction budget of the VLM using a Global Interaction Budget ($B_{\text{global}}$) and a Local Interaction Budget ($B_{\text{local}}$). Once the agent reaches either interaction limit, the interac-

tion loop is immediately terminated, even if the task has not been fully completed. Importantly, task completion is state-based rather than trajectory-based. Any valid sequence of interactions is accepted as long as it reaches the required goal states in the specified order within the interaction budget. Accordingly, when multiple components or action sequences can validly lead to the same goal according to the SSM, all such executions are accepted. The theoretical shortest path is used for efficiency-related evaluation rather than as the only valid execution trajectory. The concrete feedback formulations and the interaction budget mechanism are deferred to Appendix D.

### 3.5 Quality Control and Validation

We implemented strict quality control throughout the dataset construction pipeline. At each stage, we randomly sampled 10% of the annotated items for review, with each batch containing at least 1,000 items to ensure that our sampling strategy provides a statistically reliable estimate. An annotator's annotations were accepted only if the error rate remained below 5%; otherwise, their annotations were rejected and reassigned. During Instruction Generation stage, inspired by MidMed(Shi et al., 2023), we further filtered out unreasonable state transition sequences to ensure that the resulting instructions remain natural, coherent, and executable.

After the dataset was constructed, following MT-bench (Zheng et al., 2023), we implemented a rigorous quality validation protocol on a subset randomly sampled to comprise 1% of the entire dataset. We defined three metrics to evaluate data quality on a 3-level scale $(0, 1, 2)$, which respectively assess the quality of vision, logic, and instruction alignment. Both human experts and two advanced VLMs (*Gemini-3-Flash*, *GPT-5-mini*) are employed for evaluation. As shown in Tab. 17, our dataset achieves consistently high scores across all dimensions, with *Gemini-3-flash/GPT-5-mini/Human Expert* reaching average scores of **1.978/1.947/1.943**. The strong human-model agreement further confirms the dataset's clarity and robustness. The complete quality-control workflow, data quality table and the scoring criteria are provided in Appendix G.

### 3.6 Data statistics

Tab. 2 presents the data statistics of VGEBench. We first curated a diverse collection of 968 high-precision 3D household device models spanning 26 categories. To ensure rich interactability, we collected 7,888 views of these 3D models, recognized 3,712 unique interactive components in 234 classes (e.g., *power buttons* or *charging ports*), and annotated 7,264 BBoxes for these components.

| Section | Item | Count |
|---|---|---|
| **Assets** | # Categories | 26 |
| | # Devices | 968 |
| **Vision** | # Total views | 7,888 |
| | # Component types | 234 |
| | # Unique components | 3,712 |
| | # Annotated BBoxes | 7,264 |
| **Logic** | # UCSM | 26 |
| | *- States / Transitions* | 302 / 1,206 |
| | # SSM | 968 |
| | *- States / Transitions* | 6,352 / 15,861 |
| **Tasks** | # Total episodes | 14,953 |
| | *- Single / Multi-turn* | 4,948 / 10,005 |

Table 2: Data statistics of VGEBench. **UCSM**: Universal Category State Machine; **SSM**: Specific State Machine.

To support realistic behavior Logic, we designed 26 Universal Category State Machines (UCSMs) as high-level template for each device category, comprising 302 states and 1,206 transitions. These were then instantiated into 968 Specific State Machines (SSMs) for individual devices to a total number of 6,352 states and 15,861 transitions. Based on this foundation, we generated **14,953** Task episodes. Notably, 10,005 of these are multi-turn interactions, which imposes significant challenges for long-horizon planning and exploration.

## 4 Experiment

### 4.1 Experimental Setups

**Models.** We conduct a comprehensive experiment on **VGEBench** across three categories of VLMs: (a) Closed-source VLMs: GPT-5-mini (OpenAI, 2025), Gemini-3-Flash (Google DeepMind, 2025), Doubao-1.5-Thinking-Vision-Pro (Guo et al., 2025); (b) Open-source VLMs: Qwen3-VL-8B-Instruct (Bai et al., 2025), InternVL3.5-8B (Wang et al., 2025), Mimo-Embodied-7B (Hao et al., 2025).

**Implementation.** VGEBench is a test-only benchmark and does not define conventional train/test splits. For all models, we adopt ReAct (Yao et al., 2023) as the underlying reasoning framework and expose the complete chat history.

| Model | Task Performance | | Efficiency | | Visual Grounding and Perception | | | | Exploration |
|---|---|---|---|---|---|---|---|---|---|
| | SR ↑ | SSR ↑ | SPL ↑ | State-F1 ↑ | EIR ↑ | TIR ↑ | VSPS ↓ | GVPE ↓ | EER ↑ |
| MiMo-Embodied-7B | 1.48 | 2.79 | 1.10 | 47.86 | 7.34 | 3.44 | **0.88** | 21.66 | 15.78 |
| InternVL3.5-8B-Instruct | 3.72 | 7.71 | 2.28 | 51.07 | 23.14 | 13.12 | 1.23 | 11.26 | 20.89 |
| Qwen3-VL-8B-Instruct | 7.77 | 12.69 | 5.11 | 54.31 | 30.46 | 14.80 | 1.42 | 11.46 | 29.91 |
| GPT-5-mini | 10.78 | 17.51 | 5.88 | 57.16 | 22.86 | 10.21 | 1.20 | 6.72 | 28.81 |
| Doubao-1.5-Thinking-Vision-Pro | 16.68 | 24.30 | 12.50 | 60.61 | 46.62 | 23.93 | 1.21 | 3.21 | 37.24 |
| Gemini-3-Flash | **54.27** | **62.86** | **39.34** | **78.67** | **67.21** | **43.56** | 1.12 | **1.85** | **64.36** |

Table 3: **Main results on VGEBench. Bold** indicates the best results, and underline indicates the second-best results. ↑ indicates higher is better, and ↓ indicates lower is better.

For interaction budget control, the redundancy factor $\lambda$ is set to 5. More details about the reasoning framework and the interaction budget can be found in Appendices H and D.2.

## 4.2 Metrics

Inspired by classic works in embodied (Anderson et al., 2018) and vision (Yu et al., 2016) domains, we adopt a multi-dimensional assessment protocol covering task performance, execution efficiency, visual grounding, and reasoning stability to evaluate the capabilities of VLMs comprehensively. Detailed definitions and formulations are provided in Appendix F.

### 4.2.1 Task Performance

We primarily measure the **Success Rate (SR)** to denote the percentage of successfully completed episodes. To more accurately evaluate multi-turn tasks, we additionally report the **Sub-task Success Rate (SSR)** to assess step-wise correctness based on achieved sub-goals.

### 4.2.2 Efficiency

We report **Success Weighted by Path Length (SPL)** to jointly account for success rate and execution length. We also compute the **State-F1 Score** to measure the alignment between the model's state trajectory and the ground-truth optimal path.

### 4.2.3 Visual Grounding and Navigation

We assess visual interaction using the **Effective Interaction Rate (EIR)** (validity of component identification) and the **Target Interaction Rate (TIR)** (precision of goal-oriented grounding). Navigation efficiency is evaluated via **View Switches Per Success (VSPS)** for local efficiency and **Global View Switches Per Episode (GVPE)** for macro-level exploration cost.

### 4.2.4 Exploration

We define the **Effective Exploration Rate (EER)** as the ratio of valid operations to total steps, serving as an indicator of reasoning efficiency and safety awareness.

## 4.3 Main Results

Tab. 3 presents the comprehensive evaluation of representative VLMs on VGEBench. We report the results across nine metrics described in Section 4.2.

**VGEBench poses a significant challenge to VLM-based agents.** The result shows that only Gemini-3-Flash achieves a notable Success Rate of 54.27%, while the second-best model, Doubao-1.5-Thinking-Vision-Pro, drops sharply to 16.68%. All remaining models obtain success rates below 15%. Even when considering subtask completion, no model other than Gemini-3-Flash attains an SSR exceeding 25%. These results indicate that generalizable visually grounded exploration remains a difficult open problem for the majority of VLMs.

**Both visual grounding and exploration capabilities determine model performance.** The results demonstrate that advanced VLMs, especially **Gemini-3-Flash** and **Doubao-1.5-Thinking-Vision-Pro**, achieve significantly higher scores in both visual grounding (EIR, TIR) and exploration (EER) compared to other models. This superior capability to accurately perceive targets and efficiently explore the environment directly underpins their leading task completion rates. In contrast, **MiMo-Embodied-7B** presents a compelling counter-example. Benefiting from its specialized training in autonomous driving scenarios, it exhibits exceptional strength in View Navigation (VSPS), achieving a score of 0.88 which even surpasses Gemini-3-Flash. However, its overall task performance remains poor (SR: 1.48%) due to a critical lack of fine-grained visual grounding and

reasoning capabilities. This disparity highlights that while navigation efficiency is valuable, the integration of fine-grained visual perception and logical reasoning is the decisive factor for success in complex household environments.

## 5 Analysis

To better evaluate the ability of VLMs in exploring household devices and further investigate the underlying mechanisms, we propose four research questions (RQs). These RQs include four key dimensions in VGEBench: (1) the necessity of generalization capacity, (2) the improvement from extended interaction budgets, (3) the primary visual bottleneck, and (4) the gap between current VLMs and human performance.

### RQ1: The necessity of generalization capacity: Can VLMs immediately recognize target?

| Model | PSR | SSR |
|---|---|---|
| MiMo-Embodied-7B | 0.51% | 2.79% |
| Qwen3-VL-8B | 1.53% | 12.69% |
| InternVL3.5-8B | 0.91% | 7.71% |
| GPT-5-mini | 1.97% | 17.51% |
| Doubao-1.5-Thinking-Vision-Pro | 4.06% | 24.30% |
| Gemini-3-Flash | 12.94% | 62.86% |

Table 4: **Generalization Performance.** We report *SSR* and *PSR* (Perfect Sub-task success Rate): The proportion of sub-tasks completed using exactly the optimal number of interaction steps.

To investigate the necessity of generalization capacity in VLMs, we evaluate whether models can achieve "one-shot success" in Table 4. The results reveal a pronounced discrepancy between PSR and SSR across all models. Even for the best-performing model, Gemini-3-Flash, the PSR is only 12.94%, whereas the SSR reaches 62.86%. This gap indicates that current models exhibit little to no immediate, zero-shot understanding of the operational logic of household devices. Instead, successful performance mainly arises from feedback-driven exploration, where agents iteratively test hypotheses, correct errors, and gradually infer device-specific functional logic.

### RQ2: Do VLMs continuously benefit from extended exploration budgets?

Fig. 3 illustrates the relative performance gains as the interaction limit increases (the detailed relationship between $\lambda$ and the interaction limit can

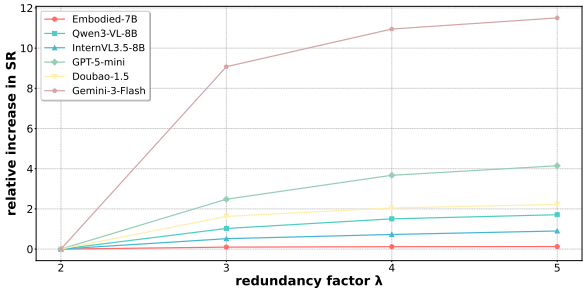



Figure 3: **Impact of interaction budget on SR score.** We normalize the performance at the baseline budget ($\lambda = 2$) to zero to isolate the relative gains. The curves illustrate the relative improvement as $\lambda$ increases.

be found in Appendix D.2). All models exhibit a pattern of improvement that gradually tapers off, indicating that current VLMs are able to convert additional interactions into better task completion, especially during the early stages of exploration. However, weaker models (e.g., Qwen3-VL-8B) show a more pronounced and earlier performance plateau, whereas advanced models such as Gemini-3-Flash, benefiting from stronger exploratory reasoning and more stable long-term memory maintenance, demonstrate a more robust ability to exploit deep exploration for effective self-correction. In addition to the interaction budget, we further examine how the available interaction history affects agents' self-correction ability. The detailed analysis is provided in Appendix E.1.

### RQ3: Where lies the visual bottleneck: Coarse-grained navigation or fine-grained grounding?

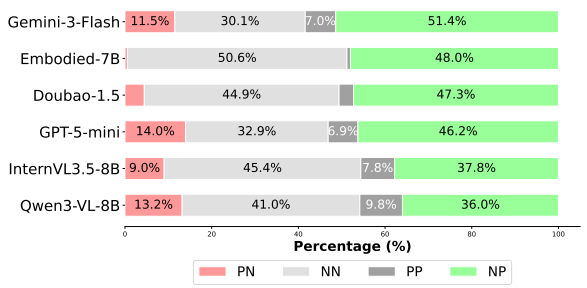



Figure 4: **Distribution of view switching actions.** We classify `Switch_view` actions into four categories based on whether the target component is visible in the start and end views (defined as "Positive" or "Negative"). **NP (Green)**: Successful navigation to the target view (Negative → Positive view); **PN (Red)**: Losing the target; **NN**: Ineffective searching; **PP**: Maintaining focus.

To isolate the bottleneck, we compared perfor-

| Model | Avg Dist ↓ | Avg W-Dist ↓ |
|---|---|---|
| Mimo-Embodied-7B | 274.04 | 441.69 |
| InternVL3.5-8B-Instruct | 188.35 | 312.43 |
| Qwen3-VL-8B | 178.09 | 312.63 |
| Doubao-1.5-Thinking-Vision-Pro | 188.53 | 290.53 |
| GPT-5-mini | 178.96 | 290.33 |
| Gemini-3-Flash | **109.18** | **163.75** |

Table 5: **Fine-grained Action Precision Analysis.** Metrics are calculated on a normalized $1000 \times 1000$ scale. *Avg Dist*: The average distance to the closest target BBox. *Avg W-Dist*: The weighted average distance that penalizes repeated failed attempts on the same target.

mance in coarse-grained navigation (Fig. 4) with fine-grained grounding (Tab. 5). The performance gap in navigation is relatively moderate: For example, GPT-5-mini's successful navigation (*NP*) is only 12.5% lower than that of Gemini-3-Flash. However, the disparity in fine-grained grounding is drastic. GPT-5-mini's coordinate error (*Avg Dist*) is 63.9% worse than Gemini-3-Flash. Crucially, the even wider gap in *Avg W-Dist* (77.3%) reveals that weaker models fail to efficiently correct their coordinates after initial failures. This empirical result confirms that the primary bottleneck in VGEBench is not navigating to the correct view, but precisely localizing the interactive component within it. To further examine the fine-grained grounding bottleneck, we additionally evaluate model robustness under common visual perturbations. The detailed results are reported in Appendix E.2.

**RQ4: How large is the gap between humans and current VLMs?**

To contextualize current VLM performance, we evaluate a human baseline on 149 episodes (approximately 1% of VGEBench) under the same interface and interaction budgets as Gemini-3-Flash. As shown in Tab. 6, humans achieve consistently higher task performance, reaching 74.50% SR and 81.43% SSR, compared with 56.38% and 64.32% for Gemini-3-Flash on the same episodes. Human performance nevertheless remains below saturation, indicating that manual-free operation of unfamiliar devices remains non-trivial even for humans under the constrained interaction setting.

Beyond task success, the complete metric and error analyses reveal a notable behavioral difference: humans produce fewer Invalid Coord Errors and therefore more frequently interact with valid component regions, enabling more informative component-level feedback for subsequent hypothesis refinement. This provides additional evidence that fine-grained visual grounding is important for efficient exploration. Full results and error-distribution comparisons are provided in Appendix C.

| Model | SR | SSR | SPL |
|---|---|---|---|
| Gemini-3-Flash | 56.38 | 64.32 | 39.23 |
| Human Baseline | **74.50** | **81.43** | **54.86** |

Table 6: Comparison between Gemini-3-Flash and Human baseline on 149 episodes under the same interface and interaction budgets.

## 6 Conclusion

To investigate the capabilities of current VLMs in visually grounded device exploration, this paper introduces VGEBench, a comprehensive benchmark designed to evaluate generalizable visually grounded exploration without reliance on manuals. We construct a Logic-Driven State Machine framework which simulates multi-turn interaction loops to provide deterministic feedback. Extensive experiments reveal that despite strong semantic knowledge, current VLMs still struggle significantly with fine-grained visual grounding and maintaining long-horizon state consistency. We hope that VGEBench will inspire further research to bridge the gap between abstract world knowledge and precise device execution.

## Limitations

VGEBench introduces a dynamic, interactive evaluation framework involving multi-turn agent-environment feedback loops. Consequently, conducting a full-scale evaluation requires significant computational resources and inference time. Additionally, the necessity for high-resolution visual observations to resolve fine-grained grounding details imposes high demands on the visual encoding efficiency and context window of current VLMs.

VGEBench employs interactive components annotated on high-fidelity rendered views, discrete atomic actions, and fixed viewpoint switching. This abstraction of low-level control to focus on high-level reasoning avoids the confounding effects of factors such as continuous robotic control and free-form camera navigation, thereby enabling a more focused evaluation of models' capabilities in the generalizable visually grounded exploration task. Although this design enables deterministic evalua-

tion, it does not fully capture all sim-to-real challenges encountered in physical robotic deployment.

The current observations are device-centered rendered images and do not fully capture complex backgrounds, natural illumination, or object occlusion in real environments. Although Appendix E.2 provides controlled robustness tests under compression, blur, and random occlusion, these perturbations cannot fully substitute for real-world observations.

## Ethics Statement

We are committed to strict ethical standards in the construction and distribution of VGEBench.

**Data Compliance.** Our dataset leverages 3D assets sourced from open platforms (Sketchfab and 3D Warehouse). We strictly adhered to the platforms' ethical and licensing guidelines. Specifically, we utilized only assets available under permissible licenses and explicitly excluded any assets marked with "NoAI" tags to respect creators' rights regarding generative AI training. The dataset focuses solely on household objects and contains no Personally Identifiable Information (PII) or offensive content.

**Professional Annotation.** We ensure that all employed human annotators underwent rigorous training specific to the categories of household devices involved in the dataset. All participants were fully informed of the intended data usage prior to their involvement. Additionally, we have provided fair and reasonable compensation for their work, ensuring that their efforts are appropriately rewarded and that the quality of the annotated content is guaranteed.

## Acknowledgments

We thank all the reviewers for their insightful and valuable comments. This work is supported by the National Natural Science Foundation of China (Grant No.U21B2009).

## References

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

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. 2017. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. 2023. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*.

Zihao Cheng, Hongru Wang, Zeming Liu, Yuhang Guo, Yuanfang Guo, Yunhong Wang, and Haifeng Wang. 2025. Toolspectrum: Towards personalized tool utilization for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20679–20699.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. 2022. A survey of embodied AI: from simulators to research tasks. *IEEE Trans. Emerg. Top. Comput. Intell.*, 6(2):230–244.

Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. 2022. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. *arXiv preprint arXiv:2211.05272*.

Google DeepMind. 2025. Gemini 3. https://deepmind.google/models/gemini/. Accessed: 2026-05-26.

Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, Jingji Chen, Jingjia Huang, Kang Lei, Liping Yuan, Lishu Luo, Pengfei Liu, Qinghao Ye, Rui Qian, Shen Yan, and 178 others. 2025. Seed1.5-vl technical report. *Preprint*, arXiv:2505.07062.

Xiaoshuai Hao, Lei Zhou, Zhijian Huang, Zhiwen Hou, Yingbo Tang, Lingfeng Zhang, Guang Li, Zheng Lu, Shuhuai Ren, Xianhui Meng, Yuchen Zhang, Jing Wu, Jinghui Lu, Chenxu Dang, Jiayi Guan, Jianhua Wu, Zhiyi Hou, Hanbing Li, Shumeng Xia, and 25 others. 2025. Mimo-embodied: X-embodied foundation model technical report. *Preprint*, arXiv:2511.16518.

Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.

Tianwei Lan, Jiaqi Wu, Zeming Liu, Zhaoxin Fan, Haifeng Wang, and Yuhang Guo. 2026. Peap: Proactive embodied action sequence planning with joint understanding of vision and audio perception. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23118–23138.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, and 16 others. 2024. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*.

Silin Li, Yuhang Guo, Jiashu Yao, Zeming Liu, and Haifeng Wang. 2025. Homebench: Evaluating llms in smart homes with valid and invalid instructions across single and multiple devices. *arXiv preprint arXiv:2505.19628*.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. 2024. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer.

Liang Ma, Jiajun Wen, Min Lin, Rongtao Xu, Xiwen Liang, Bingqian Lin, Jun Ma, Yongxin Wang, Ziming Wei, Haokun Lin, et al. 2025. Phyblock: A progressive benchmark for physical understanding and planning via 3d block assembly. *arXiv preprint arXiv:2506.08708*.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. 2024. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16488–16498.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. 2024. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*.

OpenAI. 2025. Gpt-5 system card. `https://openai.com/index/gpt-5-system-card/`. Accessed: 2026-05-26.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. 2023. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*.

Shengyi Qian, Weifeng Chen, Min Bai, Xiong Zhou, Zhuowen Tu, and Li Erran Li. 2024. Affordancellm: Grounding affordance from vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 7587–7597.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. 2019. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Xiaoming Shi, Zeming Liu, Chuan Wang, Haitao Leng, Kui Xue, Xiaofan Zhang, and Shaoting Zhang. 2023. MidMed: Towards mixed-type dialogues for medical consultation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8145–8157, Toronto, Canada. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. 2024. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

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, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, and 56 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *Preprint*, arXiv:2508.18265.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

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)*.

Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. 2023. Ferret: Refer and ground anything anywhere at any granularity. *arXiv preprint arXiv:2310.07704*.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. 2016. Modeling context in referring expressions. In *European conference on computer vision*, pages 69–85. Springer.

Zixin Zhang, Kanghao Chen, Xingwang Lin, Lutao Jiang, Xu Zheng, Yuanhuiyi Lyu, Litao Guo, Yinchuan Li, and Ying-Cong Chen. 2025. Phystoolbench: Benchmarking physical tool understanding for mllms. *arXiv preprint arXiv:2510.09507*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008.

# Appendix

## A Supplementary Statistics

This section reports supplementary statistics and explains the criteria used for selecting device categories in VGEBench.

| Category | Total | SF | 3DW |
|---|---|---|---|
| CoffeeMachine | 72 | 52 | 20 |
| DigitalAlarmClock | 71 | 66 | 5 |
| Lighter | 63 | 46 | 17 |
| Microwave | 59 | 48 | 11 |
| WashingMachine | 55 | 50 | 5 |
| MechanicalAlarmClock | 54 | 51 | 3 |
| WeighingScale | 49 | 44 | 5 |
| Heater | 48 | 37 | 11 |
| Lamp | 46 | 45 | 1 |
| ElectricKettle | 44 | 39 | 5 |
| ElectricFan | 41 | 32 | 9 |
| AirFryer | 36 | 27 | 9 |
| Flashlight | 34 | 22 | 12 |
| FilmCamera | 32 | 24 | 8 |
| Razor | 29 | 27 | 2 |
| DigitalCamera | 28 | 24 | 4 |
| HairDryer | 27 | 21 | 6 |
| Humidifier | 27 | 23 | 4 |
| Landline | 23 | 19 | 4 |
| PowerBank | 23 | 19 | 4 |
| CarKey | 22 | 14 | 8 |
| Vacuum | 21 | 19 | 2 |
| AirPurifier | 20 | 18 | 2 |
| RiceCooker | 19 | 13 | 6 |
| ElectricToothbrush | 14 | 13 | 1 |
| ThermometerGun | 11 | 11 | 0 |
| **Total** | **968** | **804** | **164** |

Table 7: Category-wise distribution of 3D models. (SF: Sketchfab; 3DW: 3DWarehouse)

**Category Selection Criteria.** We prioritize household devices that are commonly encountered in everyday life, such as *CoffeeMachine*, *Microwave*, *Lamp*, and *DigitalAlarmClock*. This choice ensures that evaluated models can rely on world knowledge rather than domain-specific expertise. At the same time, we restrict the selection to devices with relatively simple and compact physical structures, avoiding objects with deeply nested or highly articulated components, so as to prevent incomplete visual coverage from becoming a confounding factor in evaluation.

**Data Source Distribution.** As shown in Tab. 7, the benchmark contains 968 tool instances spanning 26 categories, sourced from both Sketchfab and 3DWarehouse. In our data sources, Sketchfab accounts for a higher proportion due to its greater availability of high-quality 3D models. On the other hand, 3DWarehouse, as a completely free 3D model website, has a relatively smaller number of high-quality models.

**Action Type Distribution.** Tab. 8 presents the statistical distribution of atomic actions in SSM transitions across 26 device categories. `Press` serves as the most fundamental action, appearing ubiquitously across all device types and accounting for approximately 45.20% of the total dataset. In contrast, `Rotate`, `Grasp` and `Timed_wait` exhibit significant device-dependency; they are sparse across the dataset and appear only in specific categories compatible with such affordances. Furthermore, `Pull` and `Push` demonstrate a notable correlation in their occurrence frequencies within many devices (e.g., *ElectricFan*, *Powerbank*), as these actions typically function as complementary pairs in mechanical operations, such as opening and closing drawers.

## B Error Analysis

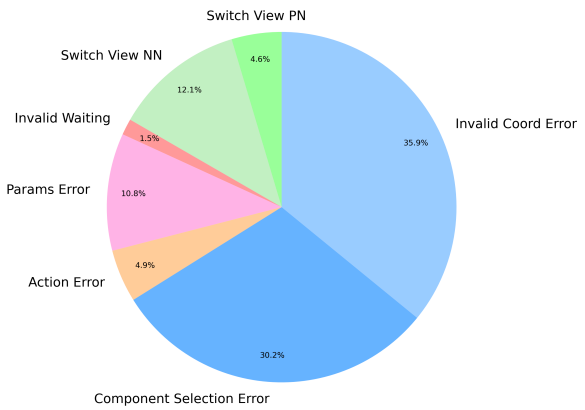



Figure 5: Distribution of error types. The errors are grouped into Fine-grained Visual Grounding (blue), Action Execution (red), and Visual Navigation (green) categories.

To deeply understand the limitations of current advanced VLMs in performing generalizable visually grounded exploration, we conducted a systematic error analysis on the failure cases collected during the evaluation of Gemini-3-Flash, which serves as the best-performing model in our benchmark. We categorize the observed errors into three

| Category | press | pull | push | rotate | grasp | timed_wait | Total |
|---|---|---|---|---|---|---|---|
| AirFryer | 813 | 44 | 1 | 1051 | 0 | 0 | 1909 |
| AirPurifier | 268 | 0 | 0 | 6 | 0 | 0 | 274 |
| CarKey | 147 | 0 | 0 | 0 | 0 | 0 | 147 |
| CoffeeMachine | 752 | 86 | 55 | 63 | 10 | 366 | 1332 |
| DigitalAlarmClock | 314 | 11 | 118 | 42 | 0 | 45 | 530 |
| DigitalCamera | 92 | 19 | 34 | 99 | 0 | 0 | 244 |
| ElectricFan | 809 | 111 | 103 | 242 | 0 | 0 | 1265 |
| ElectricKettle | 85 | 4 | 167 | 9 | 0 | 43 | 308 |
| ElectricToothbrush | 66 | 6 | 34 | 56 | 14 | 0 | 176 |
| FilmCamera | 77 | 0 | 8 | 245 | 0 | 0 | 330 |
| Flashlight | 90 | 23 | 7 | 141 | 0 | 0 | 261 |
| HairDryer | 298 | 0 | 672 | 0 | 296 | 0 | 1266 |
| Heater | 177 | 7 | 2 | 270 | 0 | 0 | 456 |
| Humidifier | 194 | 0 | 0 | 8 | 0 | 0 | 202 |
| Lamp | 92 | 0 | 0 | 0 | 0 | 0 | 92 |
| Landline | 66 | 0 | 0 | 10 | 65 | 0 | 141 |
| Lighter | 124 | 0 | 92 | 165 | 0 | 49 | 430 |
| MechanicalAlarmClock | 47 | 9 | 57 | 182 | 0 | 16 | 311 |
| Microwave | 818 | 39 | 0 | 1304 | 0 | 0 | 2161 |
| PowerBank | 56 | 45 | 45 | 0 | 0 | 0 | 146 |
| Razor | 76 | 58 | 66 | 0 | 0 | 0 | 200 |
| RiceCooker | 174 | 56 | 2 | 0 | 0 | 9 | 241 |
| ThermometerGun | 21 | 2 | 2 | 0 | 0 | 0 | 25 |
| Vacuum | 43 | 10 | 51 | 42 | 0 | 0 | 146 |
| WashingMachine | 508 | 345 | 265 | 91 | 0 | 62 | 1271 |
| WeighingScale | 304 | 14 | 210 | 12 | 0 | 0 | 540 |
| **Total** | **6511** | **889** | **1991** | **4038** | **385** | **590** | **14404** |

Table 8: Statistical distribution of atomic actions across 26 device categories.

distinct groups: Fine-grained Visual Grounding, Action Execution, and Visual Navigation. These error categories are defined as follows:

**(1)Fine-grained Visual Grounding Errors:** Failures in pixel-level localization, including *Invalid Coord Error* (clicking on non-interactive regions) and *Component Selection Error* (interacting with incorrect or non-optimal components).

**(2)Action Execution Errors:** Correct targeting but incorrect operation logic, comprising *Params Error* (wrong execution parameters, e.g., rotation direction), *Action Error* (mismatched atomic actions), and *Invalid Waiting* (waiting without triggering state changes).

**(3)Visual Navigation Errors:** Ineffective view exploration, specifically *Switch View NN* (ineffective searching) and *Switch View PN* (losing the target component).

Statistical analysis reveals that fine-grained visual grounding is the dominant failure mode, accounting for approximately 66.1% of all errors. This is significantly higher than Action Execution (17.2%) and Visual Navigation (16.7%). The high prevalence of coordinate-related errors indicates that precise spatial localization remains the primary bottleneck for current VLMs within the device-centric interaction setting of VGEBench.

In conclusion, our analysis highlights a dichotomy in model performance: while agents demonstrate competent global navigation and high-level planning, they exhibit significant weakness in **fine-grained visual perception**. Regarding reasoning, the high rate of parameter errors (10.8%) indicates a struggle with precise physical constraints even when the correct component is identified. Typical examples and the proportions for each category are shown in Tab. 9 and Fig. 5.

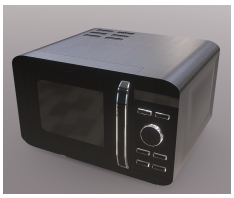
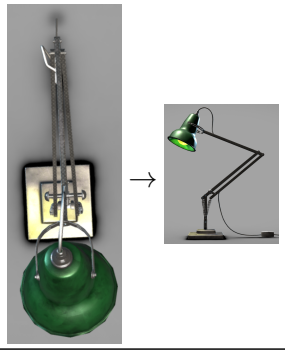
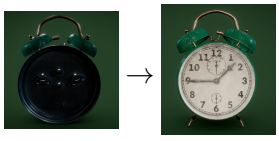
| Error Type | Current View | Model Output |
|---|---|---|
| Invalid Coord Error | | **Thinking**: To detach the blade head from the shaver, I need to locate the release mechanism. In the front view, there is a prominent red button located just below the blade head assembly. Pressing this button should release the latch holding the blade head. **Action**: Press(coord=[500, 280], params=[0]) |
| Component Selection Error | | **Thinking**: To set the dryer to hot air mode, I need to adjust the temperature switch, which is the upper sliding switch on the handle. The lower switch controls the fan speed and should remain in the 'off' position as per the instruction. I will push the upper switch to its highest position. **Action**: Push(coord=[285, 620], params=[0]) |
| Params Error | | **Thinking**: To load the food, I first need to open the air fryer door. I will pull the handle located in the center of the door downwards. **Action**: Pull(coord=[500, 420], params=[1]) |
| Action Error | | **Thinking**: The flashlight has a button visible in the current 'top' view. It's located on the neck of the flashlight. I will click on it to turn it on. **Action**: Click(params=[701, 303]) |
| Invalid Waiting |  | **Thinking**: The microwave is currently operating at medium power in short-time mode with the food inside, which fulfills the user's instruction. I will wait for the cooking process to complete. **Action**: Timed_wait() |
| Switch View NN |  | **Thinking**: The user wants to turn on the desk lamp. In the top view, I can see the lamp's structure, but the switch on the cord is not clearly visible. I will switch to the right view to get a better look at the base and the cord, which might reveal the switch more clearly. **Action**: Switch_view(params=[right]) |
| Switch View PN |  | **Thinking**: The previous action adjusted the alarm hand instead of the main time hand. To set the correct time, I need to identify the appropriate knob. I will switch to the front view to check the hand positions before adjusting the other knob. **Action**: Switch_view(params=[front]) |

Table 9: Typical examples of each error type on VGEBench. For actions that include coordinate parameters, we mark the corresponding locations with a red "×". For `Switch_view` actions, we display both the start view and the end view.

| Model | SR ↑ | SSR ↑ | SPL ↑ | State-F1 ↑ | EIR ↑ | TIR ↑ | VSPS ↓ | GVPE ↓ | EER ↑ |
|---|---|---|---|---|---|---|---|---|---|
| Gemini-3-Flash | 56.38 | 64.32 | 39.23 | 78.51 | 64.79 | 42.77 | 1.18 | 1.94 | 63.11 |
| Human Baseline | **74.50** | **81.43** | **54.86** | **85.65** | **80.22** | **52.89** | **1.12** | **1.36** | **75.72** |

Table 10: Complete human baseline results on 149 episodes. Human participants and Gemini-3-Flash are evaluated under the same interface and interaction budgets.

| Error Type | Gemini-3-Flash | Human Baseline | Difference (pp) |
|---|---|---|---|
| Invalid Coord Error | 35.5% | 26.2% | −9.3 |
| Component Selection Error | 30.6% | 36.8% | +6.2 |
| Params Error | 10.8% | 10.5% | −0.3 |
| Action Error | 4.9% | 4.5% | −0.4 |
| Invalid Waiting | 1.5% | 1.8% | +0.3 |
| Switch View NN | 12.1% | 15.4% | +3.3 |
| Switch View PN | 4.6% | 4.8% | +0.2 |

Table 11: Comparison of error distributions between Gemini-3-Flash and the human baseline. Percentages denote the proportion of each error type among all observed errors for the corresponding evaluator. Difference is computed as Human Baseline minus Gemini-3-Flash in percentage points.

# C Human Baseline Analysis

To contextualize current VLM performance, we evaluate a human baseline on 149 episodes (approximately 1% of VGEBench) under the same interface and interaction budgets as Gemini-3-Flash.

Table 10 reports the complete results across all evaluation metrics. Humans consistently outperform Gemini-3-Flash across the major task-performance, visual grounding, and exploration metrics. Human performance nevertheless remains below saturation, indicating that manual-free operation of unfamiliar devices remains non-trivial even for humans under the constrained interaction setting.

**Error Distribution Comparison.** We further compare the distributions of interaction errors between humans and Gemini-3-Flash using the same error taxonomy introduced in Appendix B. As shown in Tab. 11, Humans and Gemini-3-Flash show similar overall distributions of Action Execution Errors and Visual Navigation Errors, with the main differences concentrated in Fine-grained Visual Grounding Errors: the Invalid Coord Error rate for humans is 26.2%, which is 9.3 percentage points lower than that of Gemini-3-Flash, while the Component Selection Error rate is 36.8%, which is 6.2 percentage points higher.

This pattern can be explained from the perspective of the information gained during interaction. Humans make fewer invalid coordinate errors, indicating that **humans can localize actual interactive regions more precisely, thereby triggering valid component-level feedback and completing a more informative and effective hypothesis-interaction-refinement loop under the same interaction budget**. In contrast, the model more frequently selects invalid coordinates, causing the feedback to remain at the lower-information "invalid coordinate" level and preventing it from fully entering subsequent functional inference and feedback-driven refinement. Therefore, the human baseline not only shows that the model continues to lag behind humans, but also provides supporting evidence for the importance of fine-grained visual grounding to exploration efficiency.

# D Environment Mechanism Details

## D.1 Environment Feedback Mechanism

The environment feedback is generated deterministically based on the matching result between the agent's action and the Specific State Machine (SSM). We categorize the feedback into three distinct levels to guide the agent's error recovery:

- **Successful Transition:** If a transition occurs, the system returns the description of the transition and the new state (e.g., "Power connected, the screen has turned on").
- **Operational Errors:** If the agent interacts with a valid component but uses an incorrect action or parameters (e.g., rotating a button that should be pushed), the system provides a hint: "It seems that this component can be operated, but not in this way."

- **Null Effect:** If the coordinates do not intersect with any interactive region, the system returns a "null" feedback.

### D.2 Interaction Budget Configuration

To ensure efficient task execution, the episode terminates if all sub-tasks are completed or if any of the two budget constraints is violated:

**Global Interaction Budget ($B_{\text{global}}$):** To limit the absolute resource consumption, the total number of interaction steps across the entire episode must not exceed a static upper bound $B_{\text{global}}$. This bound is determined by the total complexity of the task, defined as

$$B_{\text{global}} = \lambda \cdot \sum C_{\text{opt}}, \tag{1}$$

where $\sum C_{\text{opt}}$ represents the cumulative optimal cost across all sub-goals. Here, the optimal cost function is calculated as

$$C_{\text{opt}} = d(S_{\text{start}}, S_{\text{end}}) + k_{\text{view}}, \tag{2}$$

composed of the shortest transition distance $d(\cdot)$ on the ground-truth state machine graph and the minimum number of view switches $k_{\text{view}}$ required along the path. $\lambda$ is a *redundancy factor* defined to tolerate reasonable exploration, which is set to 5 in our main experiments.

**Local Interaction Budget ($B_{\text{local}}$):** To prevent inefficient looping within specific sub-tasks, we maintain a dynamic interaction budget $B_{\text{local}}$ for the current execution phase. $B_{\text{local}}$ is initialized to a base value and decrements by 1 at each step. When the agent successfully completes a sub-goal and proceeds to the next target state $S_{\text{next}}$, $B_{\text{local}}$ is adaptively replenished based on the topological complexity of the upcoming task:

$$B_{\text{local}} \leftarrow \max(B_{\text{local}}, \lambda \cdot C_{\text{opt}}). \tag{3}$$

This mechanism ensures that the local budget is dynamically aligned with the difficulty of the current sub-task.

This dual-constraint mechanism enforces a rigorous standard for efficiency: the *Global Constraint* ensures the agent manages resources over the long horizon, while the *Dynamic Local Constraint* demands that the agent makes progress on specific sub-tasks without getting stuck in local optima. By grounding the budget in the ground-truth state machine ($C_{\text{opt}}$), the protocol ensures the allowance is always proportional to the actual task difficulty.

### D.3 View Mechanism

The views include six orthographic views (`front`, `back`, `left`, `right`, `top`, `bottom`), two axonometric views (`front_top`, `back_bottom`) and a special "`dashboard`" view. Because common rectangular buttons are prone to distortion in axonometric views, which makes bounding box annotation difficult, we restrict the agent's interactions to the orthographic views. The axonometric views are only used to provide the agent with global visual information.

Figure 6: Illustration of the Dashboard View mechanism. For large devices where descriptive text or buttons are too small relative to the global view, this specialized view simulates the human perspective of approaching the object for closer inspection, enabling precise fine-grained interactions.

**Dashboard View Mechanism.** For large household devices such as washing machines, the descriptive text or buttons are often too small relative to the device's global view, making interactive components difficult to recognize in the standard view. To address this challenge, we introduce a specialized **Dashboard View** that simulates the human

perspective of approaching an object for closer inspection, as illustrated in Fig. 6. To prevent the agent from taking shortcuts, we do not provide the Dashboard View directly. Instead, we design two distinct unlocking mechanisms:

**Active Unlock:** Triggered when the agent explicitly executes a Switch_view command with target coordinates falling within the dashboard region, resulting in an immediate transition to the Dashboard View.

**Passive Unlock:** Triggered implicitly when the agent attempts a spatial interaction (e.g., `Press`) within the locked region. In this case, the intended action is aborted and the view remains unchanged, but the dashboard state is updated to "unlocked" for subsequent operations.

Once unlocked, the dashboard functions as a standard view, enabling precise fine-grained interactions.

### D.4 Temporal and Chain Transitions

The simulator supports non-atomic state changes to model complex device behaviors:

**Immediate Transitions:** The state machine supports "immediate" nodes. If the system transitions to a state marked as immediate, it automatically evaluates the next transition without waiting for agent input. This allows for causal chains to be represented as a single step to the user.

**Timed Actions:** To support processes that require duration (e.g., a machine booting up), the system includes a `Timed_wait` action. This action is only valid when the current state defines a temporal transition; otherwise, it returns feedback indicating that waiting had no effect.

## E Additional Analysis

In this section, we conduct analysis experiments on a randomly sampled 10% subset of VGEBench.

### E.1 History Length Sensitivity

We vary the history length provided to the agent during multi-turn interaction, where "No Limit" denotes access to the complete interaction history, and smaller values indicate that only the most recent interaction turns are retained.

As shown in Table 12, Gemini-3-Flash consistently benefits from longer interaction history. Its SR and SSR increase from 22.61% and 31.83% with no history to 55.04% and 63.29% when the complete history is retained. This suggests that

| History Length | Gemini-3-Flash | | Doubao-1.5 | |
|---|---|---|---|---|
| | SR | SSR | SR | SSR |
| No Limit | 55.04 | 63.29 | 16.30 | 24.15 |
| 10 | 51.10 | 59.88 | 18.19 | 25.99 |
| 3 | 48.70 | 58.29 | 18.66 | 26.21 |
| 1 | 42.47 | 51.69 | 17.19 | 24.58 |
| 0 | 22.61 | 31.83 | 13.24 | 20.26 |

Table 12: The result of history length sensitivity analysis. "No Limit" denotes that the complete interaction history is provided to the agent.

Gemini-3-Flash can effectively use long-horizon interaction traces to accumulate evidence and perform self-correction. In contrast, Doubao-1.5-Thinking-Vision-Pro achieves its best performance with a shorter history window of 3 turns, slightly outperforming the no-limit setting. This indicates that long histories may introduce distracting information for weaker models, especially when early failed attempts or irrelevant visual states remain in the context. Overall, the results show that interaction history is not merely a context-length variable, but also affects how different models balance useful feedback against historical noise.

### E.2 Visual Perturbation Robustness

Following ImageNet-C (Hendrycks and Dietterich, 2019) and Random Erasing (Zhong et al., 2020), we apply three common visual perturbations to the input observations: image compression, Gaussian blur, and random occlusion. We designed the severity levels to ensure equal intervals and observable gradients:

- **Compression (Image Quality):** We tested quality levels at **50** (mild artifacts) and **20** (severe blocking), simulating low-bandwidth transmission or low-quality sensors.
- **Blur (Gaussian):** We applied Gaussian blur with $\sigma \in \{1.0, 3.0, 5.0\}$, simulating motion blur or defocusing.
- **Occlusion:** We employed a "Coarse Dropout" strategy (Zhong et al., 2020) with 8 random black patches, covering total area ratios of **10%**, **25%**, and **40%**, to simulate environmental clutter blocking the view.

Tables 13 and 14 show that both models are particularly sensitive to Gaussian blur. For Gemini-3-Flash, SR drops from 55.04% to 38.53% under strong blur with $\sigma = 5.0$, and SSR decreases from

| Perturbation Type | Severity Level | SR | SSR |
|---|---|---|---|
| Baseline | None | 55.04 | 63.29 |
| Compression | Quality = 50 | 53.29 | 62.83 |
| | Quality = 20 | 51.12 | 60.16 |
| Blur | $\sigma = 1.0$ | 51.85 | 60.94 |
| | $\sigma = 3.0$ | 40.92 | 51.18 |
| | $\sigma = 5.0$ | 38.53 | 48.12 |
| Occlusion | Ratio = 0.10 | 50.14 | 59.44 |
| | Ratio = 0.25 | 50.73 | 58.69 |
| | Ratio = 0.40 | 47.05 | 56.72 |

Table 13: The result of visual perturbation robustness analysis of Gemini-3-Flash.

| Perturbation Type | Severity Level | SR | SSR |
|---|---|---|---|
| Baseline | None | 16.30 | 24.15 |
| Compression | Quality = 50 | 16.88 | 25.22 |
| | Quality = 20 | 16.31 | 23.27 |
| Blur | $\sigma = 1.0$ | 14.65 | 22.56 |
| | $\sigma = 3.0$ | 10.28 | 17.40 |
| | $\sigma = 5.0$ | 10.16 | 15.60 |
| Occlusion | Ratio = 0.10 | 15.94 | 22.97 |
| | Ratio = 0.25 | 17.23 | 25.83 |
| | Ratio = 0.40 | 14.57 | 22.57 |

Table 14: The result of visual perturbation robustness of Doubao-1.5-Thinking-Vision-Pro.

63.29% to 48.12%. Doubao-1.5-Thinking-Vision-Pro shows a similar trend, with SR decreasing from 16.30% to 10.16% under the same blur severity. In contrast, compression and occlusion lead to more moderate degradation. These results further support our main finding that fine-grained visual grounding is a central bottleneck in VGEBench: when high-frequency details such as small component boundaries, icons, and textual cues are degraded, models struggle to maintain reliable interaction performance.

### E.3 Cross-Model Instruction Robustness

To examine whether the evaluation is sensitive to the choice of instruction generator, we regenerate the task instructions using two additional LLMs, DeepSeek-v4-pro and Qwen3.5-Plus. We evaluate both Gemini-3-Flash and GPT-5-mini using instructions generated by GPT-5-mini, DeepSeek-v4-pro, and Qwen3.5-Plus.

As shown in Tab. 15, changing the instruction generator results in only minor performance variations for both evaluated models. Gemini-3-Flash achieves SR between 55.04% and 56.86% and SSR between 63.29% and 64.19%, while GPT-5-mini

| Generator | Evaluated Model | SR | SSR |
|---|---|---|---|
| GPT-5-mini | Gemini-3-Flash | 55.04 | 63.29 |
| DeepSeek-v4-pro | | 55.52 | 63.38 |
| Qwen3.5-Plus | | 56.86 | 64.19 |
| GPT-5-mini | GPT-5-mini | 12.58 | 19.20 |
| DeepSeek-v4-pro | | 10.71 | 17.20 |
| Qwen3.5-Plus | | 11.04 | 18.05 |

Table 15: The result of cross-model instruction robustness analysis on a randomly sampled 10% subset of VGEBench.

achieves SR between 10.71% and 12.58% and SSR between 17.20% and 19.20%. In particular, using GPT-5-mini-generated instructions to evaluate GPT-5-mini does not produce a disproportionate performance increase compared with instructions generated by the other models. These results indicate that the evaluation is robust to the choice of instruction generator and does not exhibit a strong generator-specific performance advantage.

### E.4 Instruction Style Robustness

To address the concern that the current fixed language pipeline might introduce template and stylistic biases, we design an additional experiment comparing multiple instruction rewriting prototypes. The results are reported in Table 16.

The experimental results demonstrate that our framework exhibits strong robustness to rewriting styles. The standard deviations across both metrics are consistently **below 0.6%**, and the narrow confidence intervals (CIs) further confirm the stability of the relative model ordering.

## F Detailed Evaluation Metrics

In this section, we provide the rigorous definitions and calculation details for the metrics used in our main evaluation.

### F.1 Task Performance

We primarily measure the **Success Rate (SR)**, defined as the percentage of test episodes where the model successfully achieves the goal state. For better evaluating step-wise correctness in multi-turn tasks, we additionally report the **Sub-task Success Rate (SSR)** which calculates the ratio of successfully achieved sub-goals to the total number of required sub-goals within an episode.

| Model | Metrics | Version 1 | Version 2 | Version 3 | Version 4 | Std Dev | 95% CI |
|---|---|---|---|---|---|---|---|
| Gemini-3-Flash | SR | 51.64 | 51.51 | 52.71 | 52.44 | 0.59 | [49.89, 54.26] |
| | SSR | 60.09 | 59.84 | 60.95 | 59.99 | 0.50 | [58.25, 62.19] |
| Doubao-1.5-Thinking-Vision-Pro | SR | 16.66 | 17.26 | 17.39 | 18.06 | 0.58 | [15.64, 19.04] |
| | SSR | 23.96 | 24.50 | 24.50 | 25.09 | 0.46 | [22.78, 26.25] |

Table 16: The result of instruction style robustness analysis. Performance is evaluated across four different instruction rewriting versions.

### F.2 Efficiency

Following Anderson et al. (2018), we evaluate execution efficiency using two metrics:

**Success Weighted by Path Length (SPL):** A metric that penalizes successful but inefficient executions. It is calculated as

$$\text{SPL} = S \times \frac{L_{opt}}{\max(L_{act}, L_{opt})}, \tag{4}$$

where $S$ indicates success (1 or 0), $L_{opt}$ is the length of the theoretical shortest path derived from the state machine, and $L_{act}$ is the model's actual path length.

**State-F1 Score:** To measure the alignment between the model's trajectory and the optimal solution, we treat the sequence of visited states as a set and compute the F1 score between the model's state trajectory and the ground-truth optimal path.

### F.3 Visual Grounding and Perception

Inspired by RefCOCO (Yu et al., 2016), we evaluate the model's visual interaction capabilities using four metrics:

**Effective Interaction Rate (EIR):** Measures the proportion of actions where the output coordinates $(x, y)$ fall within the bounding box of *any* interactive component. This indicates the model's fundamental ability to identify and interact with valid UI elements.

**Target Interaction Rate (TIR):** Measures the precision of goal-oriented grounding. An interaction is deemed accurate only if the coordinates fall within the bounding box of a component that triggers a transition to a state with a strictly shorter shortest-path distance to the current goal.

**View Switches Per Success (VSPS)** is an episode-level metric that calculates the number of view switches executed prior to a successful action, normalized by the number of goals achieved. VSPS isolates *local navigation efficiency*, measuring how directly the model navigates to the target view when it correctly understands the task.

**Global View Switches Per Episode (GVPE)** is a dataset-level metric computed as the total number of view switches across all test episodes divided by the total number of achieved goals. Unlike VSPS, GVPE incorporates the cost of invalid exploration in failed episodes. It serves as a *macro-level indicator* of the exploration-to-success ratio.

### F.4 Exploration

Inspired by Webshop (Yao et al., 2022), we assess the model's ability to explore devices' functionality effectively by defining the **Effective Exploration Rate (EER)**. We classify operations into *valid* and *invalid* categories. Invalid operations include repetitive actions, dangerous actions, and null-transition actions that trigger no state change. The EER is calculated as the ratio of valid operations to the total number of operation steps, reflecting the model's reasoning efficiency and safety awareness.

## G Data Validation Details

### G.1 Annotation Quality Control

Our annotation process consists of two parts. First, annotators labeled the bounding boxes of all visible interactive components across 7,888 rendered views. Second, annotators annotated the trigger conditions for 15,861 state transition edges in the instantiated SSMs. Each transition trigger specifies the reactive coordinate region, the corresponding atomic action type, and the required action parameters. We use transition edges rather than components as the annotation unit because the same physical component may correspond to different state-dependent transitions or require different actions and parameters under different device states (e.g., *pull* to open vs. *push* to close).

We adopted a batch-level quality-control protocol for both annotation stages. For bounding-box

annotation, annotators first completed an initial familiarization batch of approximately 100 views, all of which were manually reviewed. In regular annotation, each batch contained no fewer than 1,000 views; with a 10% random sampling rate, at least 100 views were reviewed per batch. For transition-trigger annotation, each batch contained no fewer than 2,000 transition edges; therefore, at least 200 transition edges were reviewed per batch.

A batch was accepted only when its error rate was below 5%. Otherwise, the batch was rejected and reassigned for correction. This protocol provides reliable quality control for both visual component annotation and state-machine trigger annotation.

### G.2 Dataset Quality Validation

| Evaluator | VGA | LC | IA | Avg. |
|---|---|---|---|---|
| Gemini-3-Flash | 1.944 | 1.997 | 1.991 | 1.978 |
| GPT5-mini | 1.906 | 1.964 | 1.970 | 1.947 |
| Human | 1.910 | 1.954 | 1.965 | 1.943 |

Table 17: Data validation results on the validation subset. Scores are evaluated on a 3-level scale (0/1/2) and reported as the mean value across all samples.

In addition to batch-level annotation quality control, we established a rigorous quality validation scoring criteria to ensure dataset reliability. We defined three specific metrics to assess the quality of the visual grounding, logical feasibility, and instruction alignment. The detailed scoring criteria are described as follows:

**Visual Grounding Accuracy (VGA)**: Measures whether the annotated bounding box (visual context) accurately identifies the specific physical UI element corresponding to the execution action.

- **bad(0)**: The highlighted region is clearly incorrect, unrelated to the target element, or physically implausible.
- **fair(1)**: The highlighted region partially matches the target or presents slight ambiguity regarding the specific element.
- **good(2)**: The highlighted region clearly and correctly corresponds to the required trigger for the transition.

**Logical Consistency (LC)**: Evaluates whether the state transition and action sequence are physically feasible and logically sound for the specific device configuration.

- **bad(0)**: The transition from the start state to the end state is impossible or clearly contradicts physical constraints.
- **fair(1)**: The transition is mostly logical but may involve minor practical issues or unlikely aspects.
- **good(2)**: The transition is fully logical, feasible, and consistent with real-world physical behavior.

**Instruction Alignment (IA)**: Assesses whether the symbolic state transition represents a valid subtask that effectively contributes to the user's overall high-level intent.

- **bad(0)**: The transition is clearly unrelated to any part of the user's overall task.
- **fair(1)**: The transition is partially related or exhibits minor mismatches with the user's specific intent.
- **good(2)**: The transition is fully contained within the user's intended task and logically aligns with the execution plan.

## H Details of Exploration framework

Our exploration framework is based on a basic **ReAct** paradigm (Yao et al., 2023), which employs an iterative reasoning–action loop to guide interaction with the environment. Given visual observations, the agent first analyzes the scene to identify potential interaction targets and infer device functionality. It then executes a single atomic action grounded on this reasoning, receives environment feedback, and continues the exploration process until the task objective is achieved.

## System Prompt

You are an intelligent robotic agent capable of manipulating a wide variety of household devices based on visual inputs.

**1. Output Format**
You must structure your response strictly in the following two parts:
**Thinking:**
Analyze the visual information to identify the target interactable area. Reason about the device functionality and the physics of the interaction (e.g., whether it is a button, a knob, a door, or a purely temporal process). If the required interaction target is not visible in the current view, or the task cannot yet be executed, reason about the appropriate non-physical action.
**Action:**
Execute the operation using a precise function call format.
Format: ActionName(coord=[x, y], params=[...])
For actions that do not require a spatial target, omit coord.

**2. Coordinate System**

- **Origin (0,0):** Top-Left corner of the image.
- **X-axis:** Extends horizontally to the Right.
- **Y-axis:** Extends vertically Downwards.
- **Scale:** Normalized integer coordinates ranging from **0 to 1000** (Top-Left: [0, 0], Bottom-Right: [1000, 1000]).

**3. Atomic Action Space**
Select the most appropriate action from the atomic operations below.
**Press(coord, params)**

- **Description:** Interacting with buttons, touchscreens, switches, or triggers.
- **Params:** [duration_type]
  - 0: Click (Standard press)
  - 1: Long Press (Hold for >1s)
  - 2: Double Click (Rapid toggle)

**Rotate(coord, params)**

- **Description:** Turning knobs, dials, keys, or mechanical winders.
- **Params:** [direction, extent]
  - direction: 0 (Clockwise), 1 (Counter-clockwise), * (Unconstrained or irrelevant)
  - extent: Integer representing the number of discrete rotational steps (e.g., notches or detents). * (Rotation magnitude is unspecified).

**Push(coord, params)**

- **Description:** Applying force to slide objects, close doors, or insert plugs.
- **Params:** [direction]
  - 0: Up, 1: Down, 2: Left, 3: Right
  - 4: In / Forward (e.g., closing a door, pushing a drawer in, inserting a cable)

**Pull(coord, params)**

- **Description:** Applying force to open doors, remove lids, or unplug cables.
- **Params:** [direction]
  - 0: Up, 1: Down, 2: Left, 3: Right
  - 4: Out / Backward (e.g., opening a fridge, pulling a drawer out, unplugging)

Table 18: System Prompt Part 1: Output Format, Coordinate System and Action Space. Format constraint, constraints & boundaries, logical mechanics are highlighted.

## System Prompt

**Grasp(coord, params)**

- **Description:** Holding a handle or gripping a handheld device.
- **Params:** [state] (0: Hold / Pick up, 1: Release / Put down)

**Switch_view(params)**

- **Description:** Switching to a different camera angle or viewpoint when the required interaction target is not visible in the current view.
- **Params:** [view] (Symbolic view identifier selected from optional view list).
- Common views include six standard views (front, back, left, right, top, bottom), two axonometric views (front_top, back_bottom), and a special dashboard view. You cannot perform any actions except Switch_view when the current view is an axonometric view.
- **Dashboard View:** Some devices provide a dashboard view for interfaces where text or buttons are dense relative to the overall device size. If listed as dashboard(locked) in optional view list, it must be unlocked before use.

  Unlocking methods:

  1. **Active unlock:** Switch_view(coord=[x, y], params=[dashboard]) If coord falls within the dashboard region, the dashboard is unlocked and the view switches immediately.
  2. **Passive unlock:** Using another action (e.g., Press) with coord inside the dashboard region also unlocks the dashboard. In this case, the current action does not execute and the view does not switch.

  Once unlocked, dashboard behaves like a standard view. You are encouraged to perform interactions directly within the dashboard view when available.

**Timed_wait (params)**

- **Description:** Waiting for a time-dependent process to complete before further interaction.
- **Params:** [] (No parameters are required).

**4. Constraint:** The agent must output exactly **one atomic action per response**. Each response may contain only one "Action: " entry, corresponding to a single atomic operation from the defined action space.
**Example Output**
**Example 1**
Thinking: The user wants to change the mode to "Quick". I see a rotary dial labeled "Mode" at the bottom right. The current mode is "Medium", and switching to "Quick" requires rotating the dial two discrete levels clockwise. I will rotate it accordingly to fulfill the user's request.
Action: Rotate(coord=[850, 720], params=[0, 2])
**Example 2**
Thinking: The user wants to charge the phone. I cannot find a charging port in the current view, which suggests it may be located on another side of the device. Since charging ports are typically located at the bottom of smartphones, I should switch to the bottom view to continue the task.
Action: Switch_view(params=[bottom])
**Example 3**
Thinking: The user wants to take out the toasted bread. The machine is still operating, and the toasting process has not finished yet. I need to wait until the operation completes before performing any further interaction.
Action: Timed_wait()
**Example 4**
Thinking: The user wants to change the mode to "Quick". I see a rotary dial, but I cannot clearly recognize the labels next to it. I should switch to the dashboard view to inspect the controls more closely.
Action: Switch_view(coord=[850, 720], params=[dashboard])

Table 19: System Prompt Part 2: Action Space, Constraint and Examples. Format constraint, constraints & boundaries are highlighted.

**System Prompt**

You are a helpful assistant generating user commands for interacting with a household device.

**Input Data**

- **Device Type:** {device_type}
- **State Transitions Sequence:** {transition_display}
- **Context Information:** {context_info} (Detailed descriptions of states and actions involved)

**Task**
Convert this sequence of state transitions into a list of natural language user commands. There should be exactly one command for each transition step.

**Requirements**

1. Output a JSON list of strings, e.g., ["Command for step 1", "Command for step 2"].
2. The commands should be imperative and natural.
3. If it makes sense, you can add natural flow or connection between steps (e.g., "The wind is not strong enough, switch to high mode"), but ONLY if the transition logic supports it. Otherwise, faithfully describe the transition.
4. Avoid explicitly mentioning button names (like "power button"), focus on function/intent.
5. Do not output anything other than the JSON list.

**Examples**
**Example 1**

- **Transitions:** ["Off->Active", "Active->High"]
- **Output:** ["Turn on the flashlight", "The light is too dim, switch to high mode"]

**Example 2**

- **Transitions:** ["Standby->Running"]
- **Output:** ["Start the washing cycle"]

Table 20: Prompt for Generating User Commands from State Transitions. Format constraint, constraints & boundaries, logical mechanics are highlighted.

# I Data Sample

In this section, we present a detailed example of the state machine structure used in VGEBench, taking the *Electric Kettle* as a case study. We illustrate both the Universal Category State Machine (UCSM) and the derived Specific State Machine (SSM).

## I.1 Data Structure

Each device entry in our dataset consists of three primary fields: `name`, `initial_state`, and `states`. The `states` field is a dictionary that defines the topology of the logic graph. Each state object includes:

- `description`: A natural language description of the current device status.
- `on`: A dictionary defining valid transitions. The keys correspond to **interactive components** (e.g., *power_button*), and the values are lists of transition objects. A list is used because interacting with a single component can lead to different states depending on the action parameters (e.g., rotating a knob to position 1 vs. position 2).

**Universal Category State Machine (UCSM).** In the UCSM, each transition object primarily defines the `next_state` and a `transition_description`. Notably, some UCSMs include a special `Error` state. This state captures dangerous or invalid operations (e.g., opening the lid while the device is running). If an agent triggers such a transition, the device enters the `Error` state and is subsequently reset to the `initial_state`, simulating a safety mechanism.

**Specific State Machine (SSM).** The SSM is derived from the UCSM by pruning transitions associated with components that are not visually present on the specific 3D model instance. Unreachable states and transitions are subsequently removed to ensure graph connectivity. Crucially, the SSM enriches each transition with ground-truth execution details:

- `bbox`: A list of bounding boxes for the interactive component. This is a list because a single component may be visible and actionable from multiple viewpoints (e.g., *front*, *top*, *right*).
- `atomic_action`: The specific action type required (e.g., `push`, `rotate`).
- `parameter`: The precise parameters for the action (e.g., `["1"]`).

## I.2 JSON Examples

Listing 1 shows the structure of the UCSM for the Electric Kettle category, and Listing 2 displays the instantiated SSM for a specific kettle instance, featuring detailed visual grounding annotations. For data privacy reasons, the "path" field has been anonymized.

```
{
  "name": "ElectricKettle",
  "initial_state": "Standby",
  "states": {
    "Off": {
      "description": "Disconnect the power cord; the electric kettle is powered off",
      "on": {
        "power_input_port": [
          {
            "next_state": "Standby",
            "transition_description": "Power connected, entering standby mode"
          }
        ]
      }
    },
    "Standby": {
      "description": "Standby state, power connected but not started",
      "on": {
        "power_button": [
          {
            "next_state": "Boiling",
            "transition_description": "Start the electric kettle; begin heating"
          }
        ],
        "lid_release_button": [
          {
            "next_state": "LidOpen",
            "transition_description": "Open the kettle lid and prepare to add water"
          }
        ],
        "temperature_control": [
          {
            "next_state": "StandbyTemperatureControl",
            "transition_description": "Adjust the keep-warm/target temperature setting"
          }
        ],
        "power_input_port": [
          {
            "next_state": "Off",
            "transition_description": "Unplug the power cord; enter a power-off state"
          }
        ]
      }
    },
    "StandbyTemperatureControl": {
      "description": "Adjusting keep-warm/target temperature settings",
      "on": {
        "immediate": [
          {
            "next_state": "Standby",
            "transition_description": "Temperature setting completed; return to standby state"
          }
        ]
      }
    },
    "Boiling": {
      "description": "Heating or boiling water",
      "on": {
        "timed_wait": [
          {
            "next_state": "BoilingComplete",
            "transition_description": "Water has boiled"
          }
        ],
        "power_button": [
          {
            "next_state": "Standby",
            "transition_description": "Press the power button to stop heating"
          }
```

```
    ],
    "temperature_control": [
      {
        "next_state": "BoilingTemperatureControl",
        "transition_description": "Adjust the keep-warm/target temperature during heating"
      }
    ],
    "power_input_port": [
      {
        "next_state": "Off",
        "transition_description": "Forced power-off caused heating interruption"
      }
    ],
    "lid_release_button": [
      {
        "next_state": "Error",
        "transition_description": "Forcibly opening the top cover during heating, dangerous
operation"
      }
    ]
  }
},
"BoilingTemperatureControl": {
  "description": "Adjust the keep-warm/target temperature during heating",
  "on": {
    "immediate": [
      {
        "next_state": "Boiling",
        "transition_description": "Finished adjusting temperature; return to heating state"
      }
    ]
  }
},
"LidOpen": {
  "description": "Open the kettle lid and add water",
  "on": {
    "lid_release_button": [
      {
        "next_state": "Standby",
        "transition_description": "After adding water, close the kettle lid and return to standby
"
      }
    ],
    "power_button": [
      {
        "next_state": "Error",
        "transition_description": "Attempting to heat with the lid open, erroneous operation"
      }
    ]
  }
},
"BoilingComplete": {
  "description": "Water has boiled, boiling process completed",
  "on": {
    "immediate": [
      {
        "next_state": "Standby",
        "transition_description": "Pour out the brewed water and return to standby"
      }
    ]
  }
},
"Error": {
  "description": "Error operation state",
  "on": {
    "immediate": [
      {
        "next_state": "Standby",
        "transition_description": "An incorrect operation occurred; automatically reset to the
initial state"
```

```
          }
        ]
      }
    }
  }
}
```


Listing 1: Example of the Universal Category State Machine (UCSM) for Electric Kettle.


```
{
  "name": "ElectricKettle",
  "path": "image_view\\ElectricKettle\\device number",
  "initial_state": "Standby",
  "states": {
    "Standby": {
      "description": "Standby state, power connected but not started",
      "on": {
        "power_button": [
          {
            "next_state": "Boiling",
            "transition_description": "Start the electric kettle; begin heating",
            "bbox": [
              {
                "coordinate": [199, 815, 143, 57, 1138, 1122],
                "view": "back"
              },
              {
                "coordinate": [915, 373, 160, 135, 1195, 908],
                "view": "bottom"
              },
              {
                "coordinate": [852, 955, 184, 76, 1209, 1369],
                "view": "front"
              },
              {
                "coordinate": [324, 978, 135, 66, 786, 1191],
                "view": "right"
              }
            ],
            "atomic_action": "push",
            "parameter": ["1"]
          }
        ],
        "lid_release_button": [
          {
            "next_state": "LidOpen",
            "transition_description": "Open the kettle lid and prepare to add water",
            "bbox": [
              {
                "coordinate": [216, 61, 98, 27, 1138, 1122],
                "view": "back"
              },
              {
                "coordinate": [881, 81, 117, 34, 1209, 1369],
                "view": "front"
              },
              {
                "coordinate": [336, 87, 84, 44, 786, 1191],
                "view": "right"
              },
              {
                "coordinate": [1005, 377, 126, 96, 1292, 760],
                "view": "top"
              }
            ],
            "atomic_action": "push",
            "parameter": ["1"]
          }
        ]
```

```
    }
  },
  "Boiling": {
    "description": "Heating or boiling water",
    "on": {
      "timed_wait": [
        {
          "next_state": "BoilingComplete",
          "transition_description": "Water has boiled"
        }
      ],
      "power_button": [
        {
          "next_state": "Standby",
          "transition_description": "Press the power button to stop heating",
          "bbox": [
            {
              "coordinate": [199, 815, 143, 57, 1138, 1122],
              "view": "back"
            },
            {
              "coordinate": [915, 373, 160, 135, 1195, 908],
              "view": "bottom"
            },
            {
              "coordinate": [852, 955, 184, 76, 1209, 1369],
              "view": "front"
            },
            {
              "coordinate": [324, 978, 135, 66, 786, 1191],
              "view": "right"
            }
          ],
          "atomic_action": "push",
          "parameter": ["0"]
        }
      ],
      "lid_release_button": [
        {
          "next_state": "Error",
          "transition_description": "Forcibly opening the top cover during heating, dangerous
operation",
          "bbox": [
            {
              "coordinate": [216, 61, 98, 27, 1138, 1122],
              "view": "back"
            },
            {
              "coordinate": [881, 81, 117, 34, 1209, 1369],
              "view": "front"
            },
            {
              "coordinate": [336, 87, 84, 44, 786, 1191],
              "view": "right"
            },
            {
              "coordinate": [1005, 377, 126, 96, 1292, 760],
              "view": "top"
            }
          ],
          "atomic_action": "push",
          "parameter": ["1"]
        }
      ]
    }
  },
  "LidOpen": {
    "description": "Open the kettle lid and add water",
    "on": {
      "lid_release_button": [
```

```
      {
        "next_state": "Standby",
        "transition_description": "After adding water, close the kettle lid and return to standby
",
        "bbox": [
          {
            "coordinate": [216, 61, 98, 27, 1138, 1122],
            "view": "back"
          },
          {
            "coordinate": [881, 81, 117, 34, 1209, 1369],
            "view": "front"
          },
          {
            "coordinate": [336, 87, 84, 44, 786, 1191],
            "view": "right"
          },
          {
            "coordinate": [1005, 377, 126, 96, 1292, 760],
            "view": "top"
          }
        ],
        "atomic_action": "push",
        "parameter": ["0"]
      }
    ],
    "power_button": [
      {
        "next_state": "Error",
        "transition_description": "Attempting to heat with the lid open, erroneous operation",
        "bbox": [
          {
            "coordinate": [199, 815, 143, 57, 1138, 1122],
            "view": "back"
          },
          {
            "coordinate": [915, 373, 160, 135, 1195, 908],
            "view": "bottom"
          },
          {
            "coordinate": [852, 955, 184, 76, 1209, 1369],
            "view": "front"
          },
          {
            "coordinate": [324, 978, 135, 66, 786, 1191],
            "view": "right"
          }
        ],
        "atomic_action": "push",
        "parameter": ["1"]
      }
    ]
  }
},
"BoilingComplete": {
  "description": "Water has boiled, boiling process completed",
  "on": {
    "immediate": [
      {
        "next_state": "Standby",
        "transition_description": "Pour out the brewed water and return to standby"
      }
    ]
  }
},
"Error": {
  "description": "Error operation state",
  "on": {
    "immediate": [
      {
```

```
            "next_state": "Standby",
            "transition_description": "An incorrect operation occurred; automatically reset to the
    initial state"
          }
        ]
      }
    }
  }
}
```


Listing 2: Example of the Specific State Machine (SSM) for a specific kettle instance. Transitions now include fine-grained bounding boxes across multiple views, atomic actions and corresponding parameters.