Beyond APIs: Probing the Limits of MLLMs in Physical Tool Use

Anonymous ACL submission

Abstract

Multimodal Large Language Models (MLLMs) excel at utilizing digital APIs and increasingly serve as the "brain" of embodied AI, instructing robots to interact with the physical world. In such embodied settings, a central capability is the use of physical tools, which underpins MLLMs' ability to assist humans in real-world tasks. Despite the importance, MLLMs' proficiency in physical tool use remains largely unexplored. To address this gap, we introduce *PhysTool-Bench*, the first physical tool-use benchmark designed to evaluate MLLMs' ability to comprehend real-world scenarios, identify physical tools, and plan their use. *PhysTool-Bench* comprises 2,510 queries over 2,678 real-world physical tools spanning diverse domains, including manufacturing, electrical work, agriculture, and healthcare. Concretely, models are evaluated along two primary dimensions: 1) recognizing all physical tools present in the scene, and 2) planning the tool selection and use sequence based on the instruction and visual context. Across 13 leading MLLMs, even the strongest model (Gemini-3.1-Pro) identifies only 58.7% of tools in a scene and completes merely 21.0% of queries end-to-end. Our analysis reveals a two-level deficit: MLLMs struggle to perceive tools in realistic scenes, and the much larger drop at the planning stage further indicates a lack of functional commonsense for mapping perceived tools onto task semantics, pinpointing a critical bottleneck for the development of practical embodied AI.

1 Introduction

The ability to use tools has long been a core capability of intelligence, and Large Language Models (LLMs) have recently shown remarkable progress along this dimension. State-of-the-art LLMs now function effectively as autonomous digital agents, using software APIs to book flights, query databases, and navigate the web (Peng et al., 2023; Qin et al., 2024). However, these successes

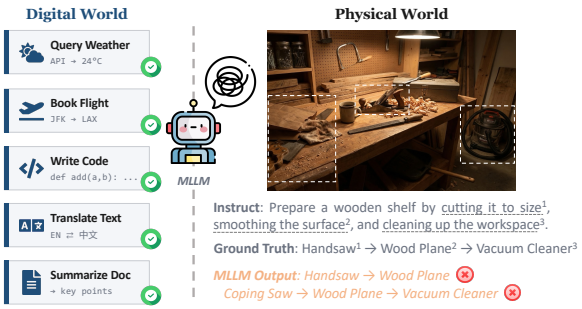


Figure 1: The capability divide between digital and physical tool use. MLLMs solve structured digital tasks reliably via APIs (left), but struggle with the visual reasoning and physical commonsense required to select and sequence tools in real-world scenes (right). *PhysTool-Bench* evaluates exactly this physical-world capability. Target tools are highlighted only for illustration.

are confined to the digital world with APIs. As an essential step toward deploying AI to assist human society, the capability of these models to follow instructions and utilize tools in the physical world must also be rigorously assessed.

Multimodal LLMs (MLLMs) are increasingly regarded as the reasoning core of embodied AI (Li et al., 2023). By integrating visual perception with language comprehension, MLLMs empower embodied agents to ground high-level instructions, such as "bring me the red mug on the kitchen counter", into actions that robots can execute. Recent systems have shown strong performance on indoor navigation (Driess et al., 2023a) and object manipulation (Liu et al., 2024), and the benchmarks driving this progress have largely focused on the same two capabilities (Xiang et al., 2020; Mu et al., 2021). Yet tool use in the physical world, arguably the next frontier for embodied AI, has received far less attention. Specifically, how well current MLLMs can recognize, comprehend, and utilize physical tools remains an open question.

To answer this question, we introduce *PhysTool-Bench*, a benchmark dedicated to evaluating phys-

ical tool use. *PhysTool-Bench* contains 2,510 queries over 2,678 distinct physical tools, drawn from manufacturing, electrical work, agriculture, healthcare, and beyond. Each query pairs a natural-language instruction with an image of a realistic environment, such as a workshop or kitchen, where the model must identify the appropriate tools for the task. As illustrated in Figure 1, given an instruction such as "prepare a wooden shelf...", the model must select the correct tools (a handsaw, plane, and vacuum cleaner) in the right order, while rejecting visually or functionally similar alternatives. We evaluate MLLMs on two progressive tasks: Task I (Physical Tool Recognition) asks the model to enumerate every tool visible in the scene; Task II (Tool Selection and Action Planning) further requires it to select the necessary tools and place them in the correct execution order, given the instruction. Together, the two tasks disentangle what the model can see from what it can reason about.

PhysTool-Bench mirrors the visual and conceptual complexity of real-world environments. Each scene contains on average 8.6 tools, of which only 3.1 are required by the instruction; the remaining items are everyday tools that may be visually or functionally related to the targets. 86.9% of queries further require multiple tools to be applied in a specific order, jointly evaluating selection and sequential planning. To capture both axes of difficulty, we report set-level F1 for tool selection and strict Exact Match (EM), which requires the predicted tools to match the ground-truth set *and* execution order. The full dataset is curated through a multi-stage quality-control pipeline (§3.2), and a human reference study confirms its quality: on queries rated highly familiar by an annotator, human EM reaches **75%**, indicating that the ground truth aligns with informed human judgment.

We benchmark 13 leading MLLMs on *PhysTool-Bench*, spanning commercial models (GPT-4o, GPT-5.2, Gemini-3.1-Pro, Qwen3-VL-Plus) and open-source models (Qwen3-VL, InternVL, Kimi-VL, DeepSeek-VL, and others). Four findings stand out. (i) **Recognition is non-trivial.** Even the strongest model identifies only 58.7% F1 of the tools in a scene; most open-source models miss more than half. (ii) **Action planning is far harder.** Gemini-3.1-Pro succeeds on merely 21.0% (EM) of queries, with EM collapsing from 34.5% on two-tool queries to 0.5% on queries requiring six or more. (iii) **Functional confusion drives failures.** 42–61% of errors stem from substituting target tools with functionally similar alternatives that are visible in the scene; a specialized open-vocabulary detector (Grounding DINO) even outperforms the best MLLM in recall by 13.4 pp, indicating that the bottleneck is physical commonsense, not perception. (iv) **The model gap is real.** Averaged across all familiarity levels (including unfamiliar domains), the human annotator reaches **38% EM**, far exceeding the best MLLM (21.0%), confirming the gap reflects model capability rather than task ambiguity.

In summary, our contributions are as follows:

- **A new dimension for evaluating MLLMs.** We introduce *PhysTool-Bench*, the first benchmark dedicated to physical tool use. This capability bridges digital tool mastery and real-world embodied deployment, yet has remained largely unexamined despite recent progress in Embodied AI.
- **A diagnostic evaluation framework.** Our two-task design, separating recognition from instruction-conditioned selection and planning, isolates failures along the perception-to-reasoning pipeline. The benchmark provides verified ground truth across **2,510 queries** spanning **2,678 tools** in everyday domains from manufacturing to healthcare.
- **A pointed empirical diagnosis.** Across 13 state-of-the-art MLLMs, we find that the bottleneck in physical tool use is not raw perception but **functional commonsense**: even when models correctly perceive a scene, they fail to map tools onto task semantics. This points to physical commonsense as the central research direction for practical embodied AI.

2 Related Work

2.1 Benchmarks for Digital Tool Learning

Recent studies have demonstrated the power of LLMs to master the use of external tools to solve complex problems (Schick et al., 2023; Yao et al., 2023). Early methods have confirmed the potential of tool learning in overcoming limitations of LLMs as a language processor while maintaining its generality (Schick et al., 2023).

Encouraged by the promising future of tool learning, a variety of benchmark and evaluation studies have been established to systematically define the problem. General benchmarks typically evaluate

LLMs' ability in tool selection and tool calling across various APIs and their diverse use cases (Patil et al., 2024). Subsequent studies expand the scope to include action planning and response generation stages (Qin et al., 2024), while later version has evolved to balance between stability and reality via a virtual API server (Guo et al., 2024). However, these existing benchmarks are predominantly confined to textual modalities and digital API environments. They fail to assess how agents visually perceive real-world scenarios and manipulate physical tools.

2.2 Evaluations for Embodied Action Planning

The transition from digital assistants to physical robots necessitates the evaluations of how well high-level reasoning can be grounded in robotic affordances. . Since the advent of SayCan (Ahn et al., 2022) which introduced pre-trained robotic value functions to assess the feasibility of each planned step, researchers have been working on bridging the gap between LLM's high-level semantic knowledge and long-horizon task planning and completion in real world. While both PaLM-E (Driess et al., 2023b) and RT-2 (Brohan et al., 2023) have achieved a tighter integration of perception and action planning, but still primarily focus on fundamental "pick-and-place" tasks or spatial rearrangements and inherently treat objects as passive targets without investigating into "tools", which play important roles in complex tasks and plans. BEHAVIOR-1K (Li et al., 2024) challenges agents with realistic physics and demanding interaction with rigid bodies, deformable materials, and complex thermal states. Yet it did not explicitly assess the zero-shot cognitive capacity of multimodal foundation models to comprehend and plan with specialized equipment.

More recently, studies have explicitly begun to explore the intersection of LLMs and robotic tool use. For example, RoboTool (Xu et al., 2024) leverages a multi-agent LLM pipeline to generate executable code, enabling robots to utilize objects creatively to overcome implicit physical constraints. Furthermore, its evaluation is severely limited in scale, encompassing a mere six task scenarios, which falls drastically short of providing a comprehensive assessment of tool-use capabilities. Because these frameworks often bypass the raw visual perception challenge by relying on predefined states or simplified environments, they fundamentally fail to evaluate an agent's capability to visually recognize diverse, professional physical tools from complex real-world scenes.

3 The Physical Tool Bench

This section details the construction and characteristics of our proposed benchmark. We first outline the definitions of two primary tasks (§ 3.1). Next, we describe the annotation pipeline and quality assurance procedures for benchmark construction, which encompass target tool combination, instruction design, the injection of confounding tools, and the generation of visual scenarios (§ 3.2). Finally, we present an analysis of the dataset statistics (§ 3.3).

3.1 Problem Formulation

Each evaluation instance (a *query*) is a tuple (I, L), where I is an image depicting a physical scenario with a set of available tools and L is a natural language instruction (e.g., "bond the cracked ceramic fragments"). Let $\mathcal{C} = \{c_1, \ldots, c_N\}$ denote the complete set of tools visible in I, which includes both task-relevant targets and other items present in the scene. We evaluate MLLMs f_θ on two progressive tasks.

Task I: Physical Tool Recognition. Given the image I and a recognition prompt P_{rec}, the model produces a predicted tool set $\hat{\mathcal{C}} = f_\theta(I, P_{rec})$, and the goal is to recover $\mathcal{C}$. This task isolates the model's ability to enumerate fine-grained physical tools from cluttered scenes, independent of any task instruction.

Task II: Tool Selection and Action Planning. Given I and L, the model outputs an ordered sequence $\hat{Y} = f_\theta(I, L) = (y_1, \ldots, y_K)$ with each $y_i \in \mathcal{C}$. The ground truth is $\mathcal{T}^* = \{(t_j, s_j)\}_{j=1}^{M}$, where $t_j \in \mathcal{C}$ and $s_j \in \mathbb{Z}_{\geq 1}$ is the execution-step index of t_j. Tools sharing the same s are interchangeable, while tools with different s values must follow their precedence ($s_j < s_k \Rightarrow t_j$ precedes t_k). This unifies ordered and order-free queries under one formulation. A prediction $\hat{Y}$ is correct iff (i) it forms a bijection with $\{t_j\}_{j=1}^{M}$ (no missing targets, no extras, no duplicates) and (ii) the step labels of its matched tools form a non-decreasing sequence.

3.2 Dataset Construction Pipeline

Constructing *PhysTool-Bench* requires balancing two goals: covering the diversity of real-world physical tools while ensuring each query admits a

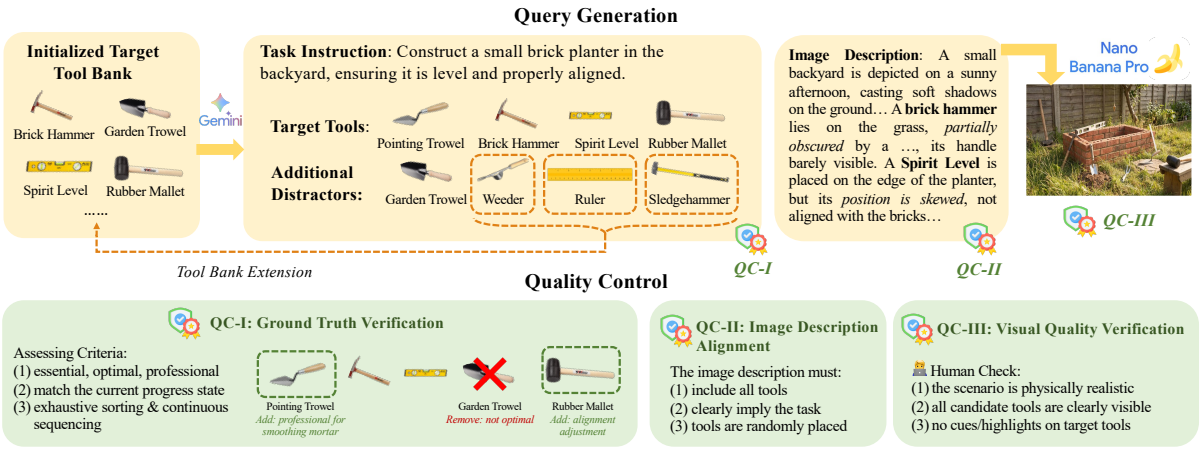


Figure 2: Overview of the PhysTool-Bench construction pipeline. Gemini generates each query (task instruction, target tools, distractors) from the tool bank, with novel distractors recycled back via *Tool Bank Extension*; Nano Banana Pro then renders the scene. Three quality-control stages follow: **QC-I** refines targets and assigns step labels; **QC-II** verifies tool-description alignment; **QC-III** applies human review for visual realism.

clear, verifiable ground-truth solution. We achieve this through a three-stage pipeline (Figure 2): tool bank initialization (§3.2.1), query generation (§3.2.2), and multi-stage quality assurance (§3.2.3). All the prompts utilized in this section are elaborated in Appendix C.

3.2.1 Tool Bank Initialization and Extension

We begin from a manually curated seed set of 310 commonly used physical tools and iteratively expand it during dataset construction along two complementary paths. First, we prompt the LLM to propose new tools across diverse application domains and functional categories, enforcing breadth across the bank. Second, novel distractor tools introduced by Gemini-3.1-Pro (Team et al., 2025) during query generation (§3.2.2) are recycled back into the bank (see the *Tool Bank Extension* loop in Figure 2), systematically capturing functionally adjacent and visually similar confounders rather than only canonical task-completing tools. The expansion terminates at saturation, yielding 2,678 distinct tools.

3.2.2 Query Generation

Target tool combinations. We prompt Gemini-3-Pro (Team et al., 2025) to formulate physical tool combinations restricted to the tool bank. Each query requires 1–3 target tools at generation time: 310 single-tool queries (one per tool, ensuring coverage), 500 two-tool queries, and 500 three-tool queries. When sampling these target tool combinations, we strictly control the selection frequency to prevent any specific tool from being overrepresented or underrepresented, thereby maintaining a balanced distribution.

Step labeling. For multi-tool combinations, we also assign each tool an execution-step index s_j: tools that must be used before others receive earlier step indices, while functionally interchangeable tools share the same step. This step structure forms part of the ground truth $\mathcal{T}^*$ defined in §3.1.

Instruction and addition-tool injection. For each target combination, GPT-4o (OpenAI et al., 2024) derives two distinct query scenarios. Instructions are phrased to describe the objective without naming required tools. Each scenario also receives 3–10 additional tools (i.e., distractors) selected for visual similarity, functional proximity, or domain relevance to the targets, reflecting the visual complexity of real-world environments.

Image description. To prepare each query for visual rendering, we synthesize a detailed image description d_i that specifies the scene composition. The description explicitly lists every candidate tool (both targets and additional tools) and instructs target tools to be randomly placed or partially obstructed to mimic real-world clutter.

Image rendering. Using each description d_i, we render the corresponding scene image $I_i = \text{ImgGen}(d_i)$ with Nano Banana Pro[1], supplemented with prompts enforcing adherence to physical laws.

[1]We additionally validate that our findings generalize beyond synthetically generated images on a real-world image subset in §5.4.

3.2.3 Multi-Stage Quality Assurance

To ensure the correctness of the ground truth and eliminate ambiguities, as illustrated in Figure 2, we implement three Quality Control (QC) checkpoints.

QC-I: Ground Truth Verification. We refine each ground-truth target set with Gemini-3.1-Pro. Given the instruction, the scene description, and a shuffled list of all candidate tools, the model evaluates each tool against three criteria: (1) essential and professional for the task, (2) consistent with the scenario state, and (3) supporting a valid execution sequence. Based on this audit, tools may be reassigned between the target and distractor sets to eliminate cases where a distractor could substitute for a target.

QC-II: Image Description Alignment. For each query, we run a programmatic check ensuring that every tool in the candidate set $\mathcal{C}$ appears as a literal mention in the image description d_i, preventing missing or hallucinated tools at rendering time.

QC-III: Visual Quality Verification. Each rendered image undergoes a final human verification stage to filter out: (1) physically unrealistic scenarios, (2) images where candidate tools are not clearly visible, and (3) critically, images containing artificial cues such as unnatural highlighting or centralizing of target tools, which would allow models to bypass physical reasoning. After filtering, the final dataset contains 2,510 verified scenarios.

3.3 Dataset Statistics

The *PhysTool-Bench* encompasses 2,510 distinct evaluation scenarios over a diverse pool of 2,678 unique physical tools, comprising 1,168 target (positive) tools and 1,519 tools that only appear as confounders. All tools are classified into 57 segments based on the United Nations Standard Products and Services Code (UNSPSC), spanning manufacturing, electrical work, healthcare, agriculture, and beyond. To evaluate resistance to visual distractors, each scenario presents a complex environment densely populated with candidate items, containing on average 8.62 tools (3.11 positive, 5.51 distractors). 86.9% of scenarios require a strict sequential execution order, while the remainder evaluate order-free combinations. Query instructions are concise (avg. 103 characters), whereas the synthesized image descriptions used for rendering are highly detailed (avg. 1,736 characters), ensuring physical realism and exact alignment with the candidate tool constraints.

4 Experimental Setup

To establish a comprehensive baseline for physical tool use, we rigorously evaluate a suite of state-of-the-art Multimodal Large Language Models (MLLMs). This section details the selected models, the prompting strategies, and the specific metrics used to quantify performance across our two primary evaluation tasks.

4.1 Implementation Details

We select a representative set of leading MLLMs, encompassing both proprietary (closed-source) and open-weight architectures. For proprietary models, we evaluate GPT-4o, GPT-5.2, Gemini 3.1 Pro, and Qwen3-VL-Plus (OpenAI et al., 2024, 2026; Team et al., 2025; Bai et al., 2025). For open-weight models, we include Qwen3-VL, InternVL3.5, Kimi-VL, DeepSeek-VL, mPLUG-Owl3, OpenFlamingo, MiniCPM, and Ovis 2.6 (Bai et al., 2025; Wang et al., 2025; Team et al., 2026; Wu et al., 2024; Ye et al., 2024; Awadalla et al., 2023; Yu et al., 2025; Lu et al., 2024) to assess the capabilities of publicly available architectures.

All evaluations are conducted in a *zero-shot* setting to test the models' inherent physical reasoning and zero-shot generalization capabilities without relying on query-specific fine-tuning or few-shot demonstrations. To ensure standardized outputs, we utilize a standardized prompt template that instructs the models to first analyze the visual scene before outputting the required tool list or sequence. The prompt templates are exact in the Appendix.

4.2 Evaluation Metrics

We define quantitative metrics for the two tasks formulated in § 3.1. For Task I, we evaluate the predicted tool set $\hat{\mathcal{C}}$ against the ground truth $\mathcal{C}$ using standard **Precision**, **Recall**, and **F1-score**. For Task II, we evaluate both *which* tools are selected and *whether* they are arranged in the correct order.

Selection (Order-Agnostic). We apply Precision, Recall, and F1 to compare the predicted tool set against the ground-truth target set $\{t_j\}_{j=1}^{M}$, ignoring order. This isolates selection accuracy from sequential planning.

Exact Match (EM). EM is a strict criterion that requires a prediction to perfectly match the ground truth. A prediction scores 1 only if (i) its selected tools exactly match the target set $\{t_j\}$, with no missing or extra tools, and (ii) the tools appear in an order consistent with their step labels s_j, i.e., tools

Table 1: Quantitative results on the proposed benchmark across various MLLMs. *Order-Agnostic* reports Task I (visual recognition: identify every available tool in the image) with Precision, Recall, F1. *Order-Aware* reports Task II (selection / planning) with Exact Match, Task-Completable Rate, Success Rate @ k. Subscripts on Task II cells are the Wilson 95% confidence half-widths over the scenario sample. "I" and "T" denote the Instruct and Thinking model. Best results are bolded.

	Order-Agnostic — Task I (%)			Order-Aware — Task II (%)				
Model	**Precision**	**Recall**	**F1-score**	**Overall EM**	**TCR**	**SR @ 1**	**SR @ 2**	**SR @ 3**
GPT-4o (OpenAI et al., 2024)	**65.15**	55.08	58.54	5.62 $\pm$ 0.90	23.04 $\pm$ 1.65	38.53 $\pm$ 2.04	15.14 $\pm$ 1.50	3.99 $\pm$ 0.82
Qwen3-VL-Plus (Bai et al., 2025)	61.93	**65.41**	**62.37**	5.66 $\pm$ 0.91	20.81 $\pm$ 1.59	39.05 $\pm$ 2.05	16.52 $\pm$ 1.56	4.59 $\pm$ 0.88
GPT-5.2 (OpenAI et al., 2026)	63.76	59.86	60.26	10.66 $\pm$ 1.21	24.72 $\pm$ 1.69	47.59 $\pm$ 2.10	22.07 $\pm$ 1.74	6.80 $\pm$ 1.06
Gemini-3.1-Pro (Team et al., 2025)	64.98	56.42	58.68	**20.96** $\pm$ 1.59	**32.12** $\pm$ 1.83	**55.83** $\pm$ 2.08	**33.35** $\pm$ 1.98	**13.90** $\pm$ 1.45
Deepseek-VL2 (Wu et al., 2024)	51.31	43.74	44.48	0.44 $\pm$ 0.27	12.48 $\pm$ 1.29	16.01 $\pm$ 1.54	4.50 $\pm$ 0.87	0.78 $\pm$ 0.38
MiniCPM (Yu et al., 2025)	48.39	56.90	49.86	1.00 $\pm$ 0.40	15.23 $\pm$ 1.41	26.24 $\pm$ 1.85	6.93 $\pm$ 1.07	1.79 $\pm$ 0.56
mPLUG-Owl3 (Ye et al., 2024)	43.32	22.18	27.60	1.12 $\pm$ 0.42	11.56 $\pm$ 1.25	16.97 $\pm$ 1.58	3.99 $\pm$ 0.82	0.73 $\pm$ 0.37
Qwen3-VL-32B-I (Bai et al., 2025)	47.55	57.17	49.83	1.24 $\pm$ 0.44	19.97 $\pm$ 1.56	30.46 $\pm$ 1.93	11.56 $\pm$ 1.34	3.07 $\pm$ 0.73
OpenFlamingo (Awadalla et al., 2023)	19.48	19.79	18.37	1.79 $\pm$ 0.52	3.59 $\pm$ 0.73	4.54 $\pm$ 0.88	0.69 $\pm$ 0.36	0.00 $\pm$ 0.09
InternVL3.5-38B (Wang et al., 2025)	50.87	42.41	44.70	2.51 $\pm$ 0.62	13.71 $\pm$ 1.35	27.02 $\pm$ 1.86	8.67 $\pm$ 1.18	1.70 $\pm$ 0.55
OVis 2.6 (Lu et al., 2024)	64.83	49.25	53.18	6.02 $\pm$ 0.93	15.46 $\pm$ 1.41	33.76 $\pm$ 1.98	12.57 $\pm$ 1.39	3.03 $\pm$ 0.72
Kimi-VL-A3B-T (Team et al., 2026)	58.60	50.82	52.91	6.78 $\pm$ 0.98	14.39 $\pm$ 1.37	31.56 $\pm$ 1.95	11.47 $\pm$ 1.34	2.61 $\pm$ 0.67
Qwen3-VL-32B-T (Bai et al., 2025)	64.16	47.87	53.15	9.33 $\pm$ 1.14	18.17 $\pm$ 1.51	40.50 $\pm$ 2.06	16.79 $\pm$ 1.57	4.63 $\pm$ 0.89

assigned to earlier steps precede those assigned to later steps. Tools sharing the same step may appear in any order. Any deviation yields a score of 0, and EM is reported as the average across all queries.

Task-Completable Rate (TCR). TCR relaxes EM by allowing additional tools beyond the ground truth. A prediction scores 1 if all target tools appear in a step-consistent order, even if extra unnecessary tools are included. TCR thus reflects whether an agent could still complete the task, while EM additionally requires a *minimal* plan.

Success Rate @ k (SR@k). SR@k ($k \in \{1, 2, 3\}$) measures EM restricted to the first k tools in the predicted sequence. SR@k captures how early in the sequence a model begins to fail and complements the all-or-nothing nature of EM.

5 Results and Analysis

We empirically evaluate the suite of MLLMs introduced in §4 on our benchmark. Our analysis proceeds from overall performance (§5.1) to fine-grained breakdowns (§5.2), targeted probing studies (§5.3), validation on real-world images (§5.4), and a fine-grained error analysis (§5.5).

5.1 Main Results

Table 1 reports overall performance across all evaluated MLLMs on both Task I (Physical Tool Recognition over all available tools in the scene) and Task II (Tool Selection and Action Planning conditioned on the task instruction). Three findings stand out.

Recognition is non-trivial, even for SOTA models. When asked to enumerate every tool visible in a real scene (Task I), no model exceeds 63% F1: the best score is Qwen3-VL-Plus at 62.37%, and the majority fall below 50%. Smaller open-weight models such as mPLUG-Owl3 (27.60%) and OpenFlamingo (18.37%) miss more than 70% of the tools present. Adding the task instruction (Task II) does not consistently help: only 4 of 13 models improve over their Task I F1, with the rest performing comparably or worse (full comparison in Appendix B). This is because Task II requires not only perceiving the tools, but reasoning about their *functional relevance* to the instruction. Many MLLMs recognize tools in Task I yet fail to map them onto task semantics in Task II, pointing to a more cognitive bottleneck that we examine in §5.3.

A large gap separates recognition from planning. Despite the perceptual advantage afforded by task instructions, the highest overall Exact Match (EM) on Task II is only 20.96% (Gemini-3.1-Pro). The order-aware metrics deteriorate even more sharply: the best Success Rate at k=3 is 13.90% (Gemini-3.1-Pro), and no model exceeds 56% even at k=1. This decoupling suggests that current MLLMs may *see* the right tools without being able to reason about which subset to use, in what order, and why.

Closed-source models lead, but the gap is narrowing. Proprietary models (GPT-4o, GPT-5.2, Gemini-3.1-Pro) consistently outperform their open-source counterparts on Task II EM, with Gemini-3.1-Pro leading on every order-aware met-

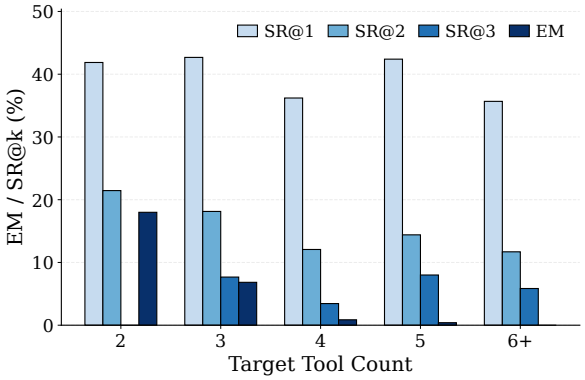


Figure 3: Exact-match performance of Qwen3-VL-32B-Thinking on Task II across the number of target tools k. SR@j requires the first j predicted tools to match the ground truth prefix; EM additionally forbids extra tools beyond the ground truth. SR@3 is undefined when k=2. While SR@1 stays at 54–57% across all complexities, EM collapses from 34.5% at k=2 to 0.5% at k=6+, exposing a sharp degradation in multi-step planning.

ric. Nevertheless, the strongest open-source reasoning models, Qwen3-VL-32B-Thinking (9.33% EM) and Kimi-VL-A3B-Thinking (6.78% EM), match or exceed GPT-4o (5.62% EM) on several order-aware metrics, narrowing the gap on planning-style tasks.

5.2 Fine-grained Analysis

5.2.1 Effect of Query Complexity

Figure 3 presents Gemini-3.1-Pro's Task II performance by the number of target tools. SR@1 remains nearly constant across complexities (54–57%), indicating that selecting the first appropriate tool is largely insensitive to query length. In sharp contrast, EM collapses from 34.5% at k=2 to 0.5% at k=6+, with SR@3 falling below 20% once the query requires four or more tools. The widening gap between SR@1 and EM thus directly quantifies the model's failure to maintain a globally consistent execution plan: even when individual tools are correctly identified at the start, the probability of completing the full sequence decays super-linearly with complexity. This pattern indicates that the dominant source of difficulty is multi-step physical planning rather than single-step tool recognition.

5.2.2 Performance Across UNSPSC Domains

We further disaggregate Task II EM across the seven broad UNSPSC domains (see Appendix A for the full breakdown). Models perform substantially better on *Healthcare* and *Office* scenarios, where procedures are well-defined and tool sets are small, but degrade markedly on *Manufacturing* and *Electrical Work*, where ordering constraints are strict and confounding tools share both visual and functional similarities. This pattern points to a systematic deficit in domain-specific physical commonsense rather than a uniform recognition limitation.

5.3 Probing Studies

5.3.1 Perception Ceiling

To localize the MLLM perception bottleneck, we evaluate state-of-the-art open-vocabulary object detectors on the same scenes. Given the candidate tool list as a text prompt, Grounding DINO achieves a recall of 70.53% — exceeding the best MLLM (Gemini-3.1-Pro at 57.09% on Task I) by 13.44 percentage points. This indicates that the visual evidence required for tool recognition is present in the images, and MLLM failures are not driven by raw perception but by the inability to enumerate visible tools or to ground them in the task instruction.

5.3.2 Human Reference

To contextualize model performance, one annotator from our research team completed a stratified sample of 100 queries, rating their domain familiarity from 1 to 5 per query. On items rated highly familiar (confidence 5), the annotator achieves **75% EM**, **75% TCR**, and **95% F1**, indicating that the benchmark admits well-defined answers aligned with informed human judgment. Across all familiarity levels, the annotator reaches **38% EM**, **49% TCR**, and **80.6% F1**, still substantially exceeding the best MLLM (Gemini-3.1-Pro at 21.0% EM). The model deficits thus reflect capability limitations rather than task ambiguity. We leave a multi-annotator study to future work.

5.4 Real-World Image Validation

A natural concern is whether our findings generalize beyond synthetically generated images. To address this, we construct a real-world image subset of 201 queries collected from web sources, manually matching the images to task instructions from the benchmark while preserving the original target labels.

Base on the evaluated results, precision drops by 8.95 percentage points on the real-world subset, whereas EM remains nearly unchanged (19.9% on generated images vs. 19.4% on real-world images). The degradation in the order-agnostic metrics is

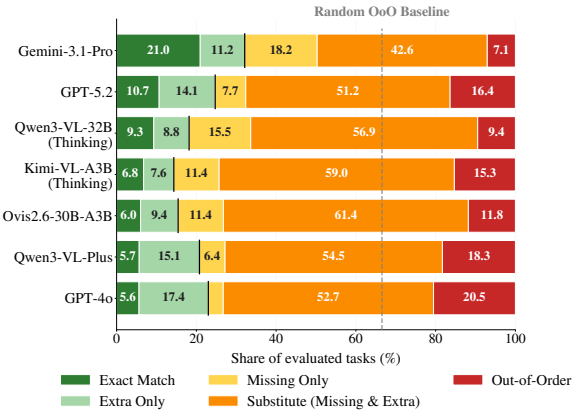


Figure 4: Task II failure decomposition across seven representative MLLMs, sorted by Exact Match (EM). Each prediction is assigned to one of five mutually exclusive outcomes. The first two (EM, Extra Only) are task-completable; the remaining three are task-blocking. The dashed line marks the expected Out-of-Order rate under random tool selection and ordering (33.5%).

consistent with the lower image quality of in-the-wild photographs, which exhibit varied resolution, lighting conditions, and motion blur. These results indicate that synthetic image generation, provides a charitable testbed: the capability gap exposed by our benchmark is not an artifact of the synthetic distribution and would likely be more pronounced under real-world deployment.

5.5 Error Analysis

Figure 4 decomposes each Task II prediction into five mutually exclusive outcomes, of which the first two (Exact Match, Extra Only) are **task-completable** and the remaining three are **task-blocking**. To probe the underlying causes, we additionally annotate 100 failure cases from Gemini-3.1-Pro. Three observations stand out, with qualitative examples for each error category provided in Appendix D.

Substitution dominates, with functional confusion as the primary driver. As shown in Figure 4, Substitute—where at least one target tool is replaced by a distractor—is the largest failure mode for every model. Our manual annotation reveals that the missing-target component of these failures is rarely caused by perception: only 22% of missed tools are visually occluded or too small to recognize, while **41.3% are *functional omissions***—tools correctly identified in Task I but excluded from the Task II plan because the model fails to recognize their functional relevance to the instruction. A further 36.7% are tools clearly visible in the scene but not recognized in either task, reflecting a fine-grained recognition gap rather than visual difficulty. On the spurious-selection side, 60% of incorrectly selected tools are distractors actually present in the scene, and 40% are hallucinated tools not visible at all. Together, these results indicate that the bottleneck is task-conditioned functional reasoning rather than raw perception.

Ordering competence exists but is fragile. Out-of-Order rates (Figure 4) sit well below the 33.5% random baseline, indicating non-trivial sequencing ability. However, root-cause analysis shows that 50% of OoO failures stem from misinterpreting the task instruction rather than generic ordering weakness, suggesting that improving instruction grounding may directly reduce ordering errors.

Failure profiles diverge across model families. Thinking models (Qwen3-VL-32B-Thinking, Kimi-VL-A3B-Thinking) trade lower ordering errors for higher Substitute rates, while GPT-4o and Qwen3-VL-Plus show the opposite pattern—high OoO with comparatively lower Substitute. Explicit reasoning thus improves sequential planning yet leaves the functional-disambiguation gap untouched. These contrasts are invisible at the aggregate EM level but become apparent in the per-category decomposition shown in Figure 4.

6 Conclusion

We introduce *PhysTool-Bench*, the first benchmark dedicated to evaluating physical tool use in MLLMs. Across 13 leading models, we find a substantial gap between digital and physical tool use: even the strongest MLLMs complete only a small fraction of queries end-to-end, and most failures arise from substituting target tools with functionally similar alternatives that are visible in the scene.

This bottleneck is not raw perception. Specialized detectors and humans both substantially outperform current MLLMs, and recognition recall persists on real-world images. The deficit lies in the functional commonsense required to map perceived tools onto task semantics. Closing this gap is unlikely to come from scaling visual encoders alone; we believe progress will require explicit grounding in multi-step physical reasoning, particularly for the long tail of specialized domains where embodied AI is most likely to be deployed.

Limitations

Coverage of tool categories. While *PhysTool-Bench* spans 57 UNSPSC segments and 2,678 distinct tools, certain specialized domains are underrepresented due to the difficulty of obtaining realistic visual references. Expanding to these long-tail domains is a natural direction for future iterations.

Static visual contexts. *PhysTool-Bench* evaluates tool use from a single static scene image, without modeling dynamic state changes (e.g., the workpiece evolving as it is processed) or interactive feedback (e.g., the model querying additional viewpoints). Extending to multi-turn, interactive tool-use evaluation is a promising direction for future work.

Ethical Considerations

Data sources and licensing. Synthetic images were generated by Nano Banana Pro in compliance with its terms of service. The real-world image subset (§5.4) was collected from publicly available web sources under fair use for academic research;

Model access. All evaluated proprietary models were accessed through their official APIs in accordance with each provider's terms of use.

Human reference. The human reference (§5.3.2) and QC-III visual verification were conducted by a research team member who consented to the task and were informed of the research purpose. The task involves only assessing visual scenes of physical tools and contains no personally identifiable information or sensitive content.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario M Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, and 24 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*.

Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. 2023. Openflamingo: An open-source framework for training large autoregressive vision-language models. *Preprint*, arXiv:2308.01390.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, and 7 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*.

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. 2023a. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

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. 2023b. Palm-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stable toolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

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 7 others. 2024. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*.

Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. 2023. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18061–18070.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui

Zhang, Yandong Guo, and Shanghang Zhang. 2024. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems 37*.

Shiyin Lu, Yang Li, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Han-Jia Ye. 2024. Ovis: Structural embedding alignment for multimodal large language model. *Preprint*, arXiv:2405.20797.

Tongzhou Mu, Z. Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. 2021. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *NeurIPS Datasets and Benchmarks*.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, and 1 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, :, Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, and 1 others. 2026. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*.

Yun Peng, Shuqing Li, Wenwei Gu, Yichen Li, Wenxuan Wang, Cuiyun Gao, and Michael R. Lyu. 2023. Revisiting, Benchmarking and Exploring API Recommendation: How Far Are We? . *IEEE Transactions on Software Engineering*, 49(04):1876–1897.

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. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, and 1 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, and 1 others. 2026. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

Weiyun Wang and 1 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *Preprint*, arXiv:2508.18265.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, and 8 others. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *Preprint*, arXiv:2412.10302.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Bernie Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. 2020. Sapien: A simulated part-based interactive environment. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11094–11104.

Mengdi Xu, Peide Huang, Wenhao Yu, Shiqi Liu, Xilun Zhang, Yaru Niu, Tingnan Zhang, Fei Xia, Jie Tan, and Ding Zhao. 2024. Creative robot tool use with large language models. In *International Conference on Learning Representations*.

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)*.

Jiabo Ye, Haiyang Xu, Haowei Liu, Anwen Hu, Ming Yan, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. 2024. mplug-owl3: Towards long image-sequence understanding in multi-modal large language models. *Preprint*, arXiv:2408.04840.

Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Yuanqian Zhao, Bokai Xu, Junbo Cui, Yingjing Xu, Liqing Ruan, Luoyuan Zhang, Hanyu Liu, Jingkun Tang, Hongyuan Liu, Qining Guo, and 7 others. 2025. Minicpm-v 4.5: Cooking efficient mllms via architecture, data, and training recipe. *Preprint*, arXiv:2509.18154.

A Per-Category Performance Analysis

To understand whether MLLMs exhibit uniform competence in physical tool use or whether their performance varies by tool category, we disaggregate the Task-Completable Rate (TCR) across the 28 UNSPSC functional segments covered by *PhysTool-Bench*. Figure 5 reports the TCR of six representative MLLMs (Gemini-3.1-Pro, GPT-4o, GPT-5.2, Qwen3-VL-Plus, Qwen3-VL-32B-Thinking, and Qwen3-VL-32B-Instruct), with segments sorted by mean score across models.

Overall trend. Performance varies dramatically across categories, spanning from above 30% TCR on the easiest segments to near-zero on the hardest.

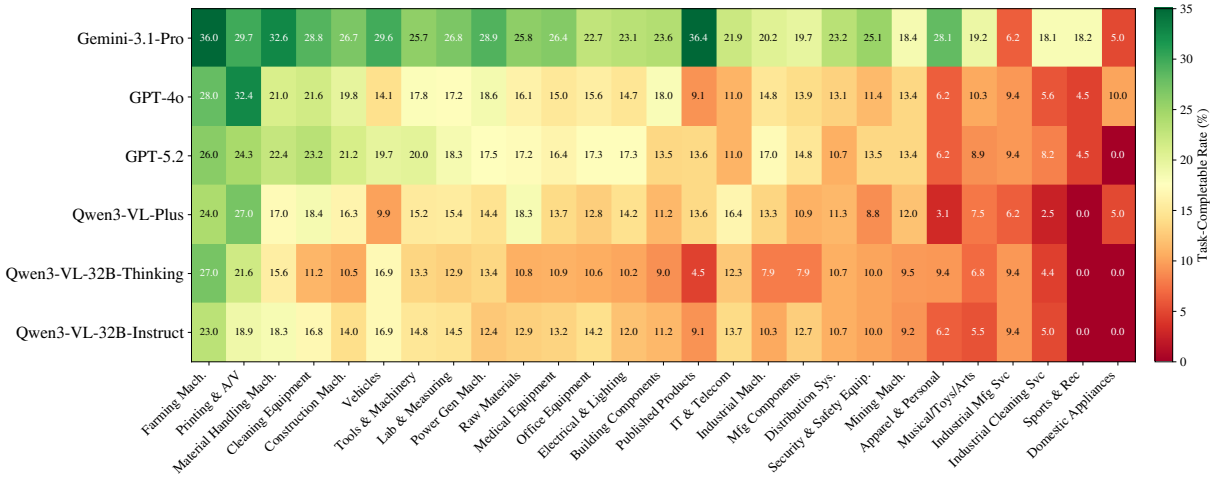


Figure 5: Per-segment Task-Completable Rate (TCR, %) across 28 UNSPSC functional segments, for six representative MLLMs. Segments are sorted left-to-right by mean TCR across models (descending). All models exhibit a consistent decline from left to right, with the cross-model ranking of segments highly correlated, indicating that category-level difficulty is intrinsic to the task rather than model-specific.

Gemini-3.1-Pro maintains the highest TCR on the leftmost segments (e.g., 30.8% on Farming Machinery, 27.5% on Cleaning Equipment) but collapses on the rightmost categories, falling to 4.8–9.1% on segments such as Industrial Cleaning Services, Sports & Recreation, and Electronic Components. This pattern is consistent across all six evaluated models, with cross-model correlation of segment rankings exceeding 0.85: the same segments are easy or hard for every model.

What makes a category easy or hard? The leftmost (easiest) segments — Farming Machinery, Cleaning Equipment, Construction Machinery, Vehicles, Power Generation — share two properties: (i) tools are *visually distinctive* (e.g., a tractor or a leaf blower is unlikely to be confused with another tool), and (ii) the mapping from instruction to tool is largely *one-to-one* (e.g., "mow the lawn" unambiguously implies a lawn mower). Under these conditions, even moderate functional reasoning suffices to recover the correct tool set.

In contrast, the rightmost (hardest) segments — Industrial Cleaning Services, Sports & Recreation, Apparel & Personal, Electronic Components — exhibit *fine-grained functional overlap* among candidates. For instance, distinguishing between an arbor press, a hydraulic press, and a punch press in Industrial Mfg Services requires specialized domain knowledge that current MLLMs do not reliably possess. Similarly, Electronic Components scenarios frequently demand multi-step procedures involving visually similar instruments (e.g., a multimeter, an oscilloscope probe, and a logic analyzer), which our analysis in §5.5 identifies as a primary trigger of functional substitution errors.

B Task I vs. Task II: Recognition Under Instruction Conditioning

To better understand whether task instructions help MLLMs identify relevant tools, we compare model performance on Task I (visual recognition: identify every tool visible in the image) and Task II (selection: identify only the tools required by the instruction) on the same 2,510 scenarios. Both tasks are evaluated with set-level Precision, Recall, and F1, computed against their respective ground-truth sets: Task I against the full set of available tools in the scene, and Task II against the task-relevant target tools. Table 2 reports the per-model breakdown along with $\Delta\text{F1} = \text{Task II F1} - \text{Task I F1}$, where positive values indicate that the model benefits from instruction conditioning.

Aggregate trend. Across 13 evaluated models, only 5 exhibit a positive ΔF1, while the remaining 8 perform comparably or worse on Task II. The largest gains are observed on OVis 2.6 ($+10.80$ pp) and Gemini-3.1-Pro ($+8.64$ pp), suggesting that these models are able to leverage the instruction as an effective attentional prior. In contrast, most other models, including GPT-4o (-2.68 pp), GPT-5.2 (-0.76 pp), Qwen3-VL-Plus (-6.11 pp), and several open-source models, show no improvement or a slight degradation.

Table 2: Comparison of Task I (visual recognition: identify every tool visible in the image) and Task II (tool selection / planning) on the same 2,510 scenarios. Task I GT = `shuffled_available_tools`; Task II GT = task-relevant target tools. ΔF1 = Task II F1 − Task I F1 — positive values indicate the model is stronger at closed-set selection than open-set recognition. Best results per column are bolded.

	Task I — Recognition (%)			Task II — Selection (%)			
Model	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**ΔF1**
GPT-4o (OpenAI et al., 2024)	**65.15**	55.08	58.54	53.56	63.90	55.86	−2.68
Qwen3-VL-Plus	61.93	**65.41**	**62.37**	56.78	61.04	56.26	−6.11
GPT-5.2	63.76	59.86	60.26	59.44	64.82	59.50	−0.76
Gemini-3.1-Pro	64.98	56.42	58.68	**71.61**	**67.87**	**67.32**	**+8.64**
Deepseek-VL2	51.31	43.74	44.48	34.86	51.03	39.34	−5.14
MiniCPM	48.39	56.90	49.86	37.96	54.45	42.25	−7.61
mPLUG-Owl3	43.32	22.18	27.60	29.28	50.09	33.35	+5.75
Qwen3-VL-32B-Instruct	47.55	57.17	49.83	40.81	63.31	47.32	−2.51
OpenFlamingo	19.48	19.79	18.37	15.37	11.42	12.05	−6.32
InternVL3.5-38B	50.87	42.41	44.70	46.89	49.64	45.99	+1.29
OVis 2.6	64.83	49.25	53.18	54.66	50.26	49.64	−3.54
Kimi-VL-A3B-Thinking	58.60	50.82	52.91	54.17	49.80	49.11	−3.80
Qwen3-VL-32B-Thinking	64.16	47.87	53.15	60.86	53.53	54.39	+1.24

Why does instruction conditioning not uniformly help? The result is initially counterintuitive: one might expect the instruction to narrow the model's attention to a smaller, task-relevant subset of tools and thereby simplify the problem. However, Task II imposes an additional reasoning demand on top of perception. The model must not only *see* the tools, but also judge their *functional relevance* to the instruction (e.g., recognizing that epoxy resin, rather than duct tape, is the appropriate adhesive for repairing ceramic). For models that lack robust physical commonsense, this additional reasoning step introduces errors that outweigh the benefit of a narrower target set: they may drop correct targets whose relevance is not obvious, or substitute them with functionally adjacent alternatives. Stronger models such as Gemini-3.1-Pro and OVis 2.6 appear better able to exploit the instruction without incurring these costs, whereas others are essentially neutralized or pulled down by the added cognitive burden.

Implications. This pattern reinforces our central diagnosis: the bottleneck in physical tool use is not raw visual perception, but the higher-level reasoning required to ground perceived tools in task semantics. We provide complementary evidence for this view through a perception-ceiling experiment with open-vocabulary detectors (§5.3) and an error decomposition that identifies functional substitution as the dominant failure mode (§5.5).

C Prompt Templates

This appendix lists the prompt templates used throughout the dataset construction pipeline (Sections C.1–C.5) and the evaluation procedure (Sections C.6–C.8). For brevity, system messages and minor formatting tokens are omitted; full versions are released with the dataset.

C.1 Target Tool Combination Generation

We further expand the tool bank by prompting Gemini-3.1-Pro to include common combination of 2 or 3 tools that usually worked together to complete a task. Notably, the task instructions generated here only serve as a guide to ensure the tool combination is feasible and commonly acknowledged in real-world, avoiding getting random combinations. These task instructions are dropped after we obtained the tool combinations.

Prompt: Tool Combination Generation

[You are an expert in tool selection and tool usage across diverse real-world domains. I have attached a set of tools. Your goal is to propose 100 distinct combinations of exactly 2 tools from this set. For each combination, design a specific, realistic target task that requires the usage of all tools to be successfully completed. For each combination, output a single JSON object containing exactly the following two fields: (1) task_instruct: A clear task instruction written in English. The task must require the use of all the 2 target tools to be completed. Do NOT mention or imply any specific tools, including any target tools listed in tools_target in (2). (2) tools_target: 2 required tools needed to complete the task. The tools must be exactly from the attached tool list. If the tools are used in a specific order, list them in the correct operational sequence.]

C.2 Task Instruction Generation

The task instruction is generated by prompting GPT-4o with the pre-determined initial target tools (in form of single tool or tool combination).

Prompt: Task Instruction Generation

["task_instruct": A clear task instruction in English. The task must require ALL the target tools [tools_list] to be completed. Do NOT mention or imply any specific tool or contain part of the tool name word.]

C.3 Distractor Selection

The distractors are selected by prompting GPT-4o with the target tool, along with the generation of task instruction and image descriptions. For each initial target tool(s), two distinct task scenarios will be constructed, and the two different numbers of distractors to include in each task scenario are randomly chosen between 3 and 10.

Prompt: Distractor Selection

["tools_negative": A list of tools that are NOT required for this task. - Scenario 1 must have exactly neg_counts[0] items. - Scenario 2 must have exactly neg_counts[1] items. These tools should be confusing or misleading - they might: - Look similar to the target tools - Have similar functions to the target tools - Be used on similar objects but be wrong choices - Be commonly associated with the same work domain Make these negative tools realistic distractors.]

C.4 Image Description Generation

The image description is generated simultaneously with the task instruction and distractors by prompting GPT-4o. Before a piece of task scenario (including target tool(s), task instruction, distractors and image description) is saved, we would check and ensure that all the tools are clearly mentioned and addressed in the corresponding image description.

Prompt: Image Description for nano-banana-pro

["img_desc": A detailed English description of a single image depicting the scenario. The image must: - Clearly imply the task to be completed - Show ALL tools from both tools_target and tools_negative - Make the correct target tools look randomly placed and partially hidden; they should NOT be highlighted, should not be placed conspicuously, and should not appear ready to complete the task. - When describing technical or professional workspaces, ensure that tools adhere to their mechanical function. - Include specific details about environment, lighting, angles, tool placement, and scene context - Be detailed enough to generate a realistic, plausible image.]

C.5 QC-I: Target Tool Verification

In QC-I, we refine the target tools and determine the chronological step orders in a more rigorous way by prompt Gemini-3.1-Pro with 'Task Instruction' , 'Current Scene' (the first paragraph of the image description), and 'Available Tools' (the combined set of initial target tool and distractors).

Prompt: QC-I Target/Distractor Audit

[You are an expert AI agent orchestrator evaluating tool selection capabilities across diverse professional domains. I will provide a 'Task Instruction', a 'Current Scene' description, and 'Available Tools'. Your objective is to identify the ABSOLUTE MINIMAL REQUIRED SET of professional tools and sequence them based on scene progress. THE THREE LAWS OF TOOL ORCHESTRATION: 1. THE UNIFIED VIABILITY TEST: A tool is strictly REQUIRED only if its removal causes the task to physically fail, violate safety, or violate professional industry standards. - Implicit Constraints: You must consider implicit constraints. (e.g., studying animals "without disturbing habitat" standardly requires an unattended tool like a 'Wildlife Camera Trap' to avoid human presence, making it professional necessity). - Technical Standards: You must prioritize professional-grade methods over amateur workarounds (e.g., prefer 'Heat Gun' over 'Electrical Tape' for professional automotive wiring). - Nice-to-Haves: Reject any tool that merely provides convenience but isn't required for success (e.g., GPS, Tripods). 2. SHARP REDUNDANCY ELIMINATION: If multiple tools overlap in fulfilling the requirement of Law 1 (e.g., Telescope vs. Field Binoculars for mobile observation), you MUST select ONLY the single most contextually appropriate tool and move all alternatives and their specific accessories to 'negative_tools'. 3. TASK LIFECYCLE TRACKING: Evaluate '<img_desc>' to determine what has already been completed. - REJECT tools meant ONLY for phases already finished in the image. - RETAIN and DELAY tools needed for remaining phases, final reassembly, or closing up to the LAST steps of the sequence.
Rules for Output (Strictly Follow JSON Schema): 1. 'tool_analysis': Step-by-step evaluation of EVERY available tool. - 'viability_and_standard_justification': Explain why this tool is a professional and physical necessity based on Law 1 and 2. Write 'Failed' if it is non-essential, amateurish, or redundant. - 'status': "Target" OR "Negative" (state exact reason: Non-Essential / Substandard / Redundant / Already Completed). - 'sequence_logic': Timing rationale based on scene progress, or 'None'. 2. 'target_tools': List of selected tools. 3. 'target_steps': Integers representing the execution order (starting at 1, continuous, same number for parallel tools). 4. 'negative_tools': List of rejected tools.]

C.6 Evaluation Prompt — Task I (Tool Recognition)

We test MLLM's ability in recognizing all available tools in the scene by the following prompt.

Prompt: Task I — Tool Recognition

[List all tools in this image. Please provide only the names of the tools, separated by commas. Do not include any explanations or extra text.]

C.7 Evaluation Prompt — Task II (Tool Selection and Planning)

We further evaluate MLLM's ability to address the task in the provided scene by the following prompt.

Prompt: Task II — Tool Selection and Action Planning

[Given the following TASK, which tool(s) in the image are most appropriate to complete the task? Please list the name(s) of the selected tools in the order they should be used and separate them by commas. No explanation needed. TASK: task_instruct. SELECTED TOOL(S) (in order of use):]

C.8 LLM-as-Judge Prompt

To match model predictions against the ground truth, we employ a hybrid pipeline combining case-insensitive string matching with the following LLM judge.

Prompt: LLM-as-Judge

[You are an expert evaluator. I have a list of 'Identified Tools' predicted by a model. Your task is to map each 'Identified Tool' to the correct 'Target Tool' name (if applicable) for the provided task, while ensuring it does not refer to any 'Negative Tools' (distractors).
Rules: 1. Only match if the Identified Tool is clearly the same tool as a Target Tool. 2. If the Identified Tool is ambiguous and could potentially refer to a Negative Tool, DO NOT match it. 3. Use the exact string from the Target Tools list for the value in your mapping. 4. Return ONLY a valid JSON object where keys are the Identified Tool strings and values are the corresponding Target Tool strings. 5. DO NOT map multiple Identified Tools to the same Target Tool - each target can appear at most once.]

D Qualitative Examples

D.1 Successful Cases

Figure 6 shows three queries solved correctly by the strongest model, illustrating the capabilities currently within reach.

Task: “Automate lawn maintenance to ensure even distribution of water over a large area without manual intervention.”

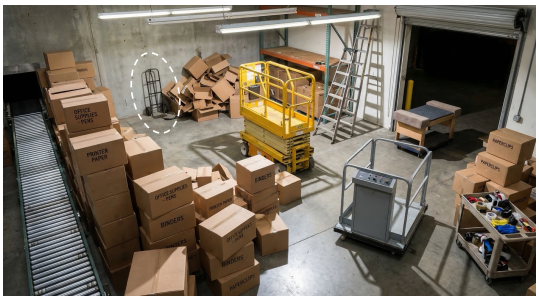
MLLM Output (*Target tools*):
- ✓ Step 1: water timer (*sprinkler timer - automating the water flow schedule*)
- ✓ Step 2: garden hose (*garden hose - transporting water*)
- ✓ Step 3: sprinkler (*irrigation sprinkler - evenly distributing*)

Negative tools:
- x Hedge Trimmer: Used for trimming hedges and bushes, which is **unrelated** to automating water distribution.
- x Rake: Used for gathering leaves or debris, **unnecessary** for watering tasks.
- x Fertilizer Spreader: Used for applying nutrients to the lawn, which does **not fulfill the requirement** of watering.
- x Shovel: Used for digging, **unnecessary** for setting up an above-ground automated sprinkler system.
- x Leaf Blower: Used for clearing leaves, **unrelated** to lawn watering.
- x Lawn Mower: Used for cutting grass, **unrelated** to lawn watering.
- x Rain Gauge: Measures rainfall does **not actively contribute to** mechanical automation or distribution of water.

Figure 6: The example and analysis of the "Exact Match" case.

D.2 Failure Cases by Error Type

Figures 7–10 present representative failure cases for each of the four error categories identified in §5.5.



Task: “Organize a large shipment of boxed office supplies from the delivery area to the storage room, ensuring all boxes are moved efficiently and safely.”

MLLM Output (*matched tools*):
- × Step 1: conveyor belt (*conveyor roller – **not efficient***)
- ✓ Step 2: hand truck (*hand truck – professional for moving stacked boxes*)
- × Step 3: scissor lift (*scissor lift - **irrelevant** for ground-level box transportation*)

Negative tools:
- × Platform Lift: **not required** for horizontally moving standard boxed office supplies across a warehouse floor.
- × Utility Cart: far **less efficient** for pre-stacked boxes as it would require manually unstacking and reloading every box, violating sharp redundancy elimination.
- × Furniture Mover: unsafe and substandard for moving precariously balanced stacks of loose boxes.
- × Warehouse Ladder: **unsafe and unnecessary**

Figure 7: The example and analysis of the "Extra Only" error.

Task: “Prepare a large agricultural field for planting and ensure the soil is enriched for optimal crop growth.”

MLLM Output (*target tool*):
- ✓ Step 1: plow (*plow – break the ground*)

Missed Target Tools:
- ✓ Step 2: Fertilizer Spreader - enrich the soil
- ✓ Step 3: Cultivator – necessary for secondary tillage to break up clods

Negative tools:
- × Harvester: **unrelated** to prepare the field for planting.
- × Seed Drill: **unrelated** to the task, which is not the actual sowing of seeds.
- × Sprinkler: **beyond the scope** of the task.
- × Tiller: **redundant** to the cultivator for secondary tillage and seedbed preparation.
- × Irrigation Pipe: **not required** for soil preparation and enrichment.
- × Hoe: **amateurish and physically unfeasible** for preparing a vast agricultural field.

Figure 8: The example and analysis of the "Missing Only" error.

Task: "Repair a broken trace on a PCB to restore electrical connectivity."

MLLM Output:
× Step 1: wire cutters
× Step 2: needle-nose pliers
✓ *Flux Pen (target tool) - professional necessity to clean oxidation*
✓ Step 3: soldering iron (*soldering station - melt solder and bond a jumper wire or solder bridge*)

Negative tools:
× Soldering Wick: **not physically necessary** to bridge a broken trace.
× Desoldering Pump: **not relevant** for surface trace repair.
× Hot Air Rework Station: **imprecise** for pinpoint trace repair compared to a soldering iron.
× Wire Cutters: **irrelevant** to broken trace repair
× Needle-nose Pliers: **irrelevant** to broken trace repair

Figure 9: The example and analysis of the "Substitute" error.

Task: "Create a custom wooden picture frame ensuring perfect corners and secure joints."

MLLM Output:	Target Tool Sequence:
✓ Step 1: Tape Measure	✓ Step 1: Tape Measure
✓ Step 2: Square	✓ Step 2: Square
✓ Step 3: Saw	✓ Step 3:Saw
✓ Step 4: Sandpaper	✓ Step 4:Wood Glue
✓ Step 5: Wood Glue	✓ Step 5: Bar Clamp
✓ Step 6: Bar Clamp	✓ Step 6: Nail Gun
✓ Step 7: Nail Gun	✓ Step 7: Sandpaper

→ Such Out-of-Order mistake will fail to ensure the perfect corners since the polishing using sandpaper is done in the middle of the process.

Figure 10: The example and analysis of the "Out-of-Order" error.