Reasoning under Vision: Understanding Visual-Spatial Cognition in Vision-Language Models for CAPTCHA

Python Song Luke Tenyi Chang Yun-Yun Tsai Penghui Li Junfeng Yang
Department of Computer Science
Columbia University
500 West 120th Street
New York, NY 10027, USA

Abstract

Modern visual CAPTCHA has evolved from simple perception tests into real-world spatial reasoning challenges. It remains unclear whether state-of-the-art vision-language models can solve such tasks through their own reasoning ability. *Existing benchmarks offer limited insight into this question as they often ignore intermediate reasoning steps, typically assess only final answers, and thus cannot know how models fail.* To address this gap, we present **CAPTCHA-X**, the first unified evaluation framework for real-world CAPTCHA solving with reasoning steps, enabling automatic assessment of solving accuracy and reasoning quality. Using this framework, we show that reasoning substantially improves performance. Beyond evaluation, **CAPTCHA-X** exposes characteristic reasoning failure modes of vision-language models and reveals reasoning scaling laws that connect multiple dimensions of reasoning. Furthermore, evaluating pure vision generation models reveals an interesting phenomenon: generation models also fail at multi-step logical planning on images. These findings motivate our super-simple and reasoning-centric agent built solely on vision-language models, designed to identify and correct errors in intermediate reasoning steps before translating reasoning into executable actions.

1 Introduction

Acting accurately in the visual world requires more than perception alone. Humans routinely solve spatial problems by identifying visual structure, reasoning over geometric relationships, and mapping that reasoning into precise actions. Modern visual CAPTCHAs naturally reflect this combination of abilities. Originally introduced as a security mechanism to distinguish humans from machines (Von Ahn et al., 2003), CAPTCHAs have evolved with the progress of OCR and computer vision systems, shifting from simple text recognition toward complex visual–spatial puzzles (Wang et al., 2018; Gao et al., 2021; Luo et al., 2026). Solving CAPTCHAs requires a model not only to perceive visual content, but also to reason over spatial relationships, ground decisions to precise target regions, and translate them into executable actions (Liu et al., 2023). As a result, modern visual CAPTCHAs have become a naturally occurring real-world test of spatial intelligence for vision-language models (Ding et al., 2025).

Existing strong CAPTCHA solvers often achieve performance by augmenting VLMs with specialized computer-vision tools or structured external pipelines. For example, recent systems combine VLMs with specific CV tools or rely on more structured neuro-symbolic procedures (Teoh et al., 2025; Deng et al., 2025). These systems show that strong CAPTCHA solving is possible when VLMs are augmented with these external tools, but they do not clearly answer a more fundamental question: **Can the VLM infer the correct spatial coordinates of a CAPTCHA end-to-end using only its own reasoning capability?**

Existing CAPTCHA benchmark studies also provide limited insight into this question. While recent work (Luo et al., 2026) introduces one reasoning-related metric, it lacks reasoning annotations and a comprehensive protocol, preventing a comprehensive evaluation of how models reason. Other benchmarks (Wu et al., 2025), a valid click region is reduced to a single annotated point, and predictions are judged by whether they fall within a fixed threshold around that point. As a result, existing benchmarks remain insufficient beyond the aforementioned limitations, with two additional weaknesses: they neither faithfully capture real-world CAPTCHA performance nor provide a principled method for diagnosing reasoning failures.

In this work, we introduce **CAPTCHA-X**, the real-world CAPTCHA benchmark with reasoning

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 803–824
November 6-10, 2026 ©2026 Association for Computational Linguistics

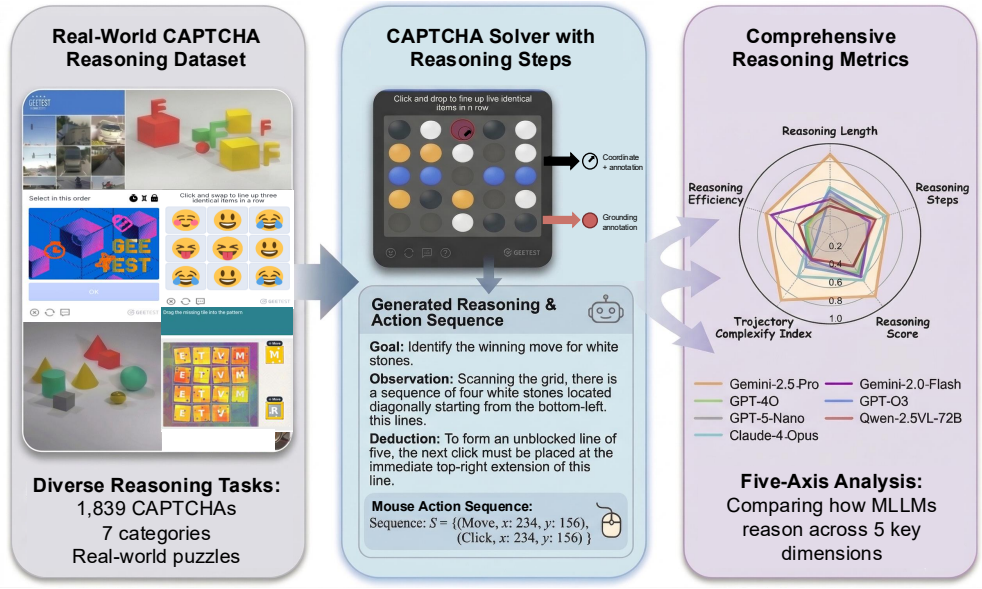


Figure 1: CAPTCHA-X collects multiple real-world CAPTCHA, provides reasoning steps and fine-grained metrics.

steps, and establish the protocol for comprehensive reasoning-centered evaluation for studying CAPTCHA solving in VLMs. Beyond measuring final solving accuracy, **CAPTCHA-X** enables systematic analysis of how models reason under realistic conditions. Through this benchmark, we provide evidence that reasoning is a fundamental factor of CAPTCHA solving.

In addition, our benchmark is accompanied by a diagnostic protocol for identifying reasoning errors. We also reveal reasoning scaling laws that connect reasoning complexity, efficiency, and task performance. We further evaluate pure visual generation models. Our experiments identify an interesting phenomenon: while generation models successfully eliminate spatial grounding errors, they suffer from logical hallucination when faced with multi-step reasoning tasks, whereas VLMs rarely exhibit this failure mode. This inherent dilemma between spatial accuracy and logical coherence, in turn, motivates our design of an agent built solely on VLMs to identify and correct intermediate reasoning hallucinations and grounding errors before execution. Our super simple agent achieves state-of-the-art performance on five CAPTCHA types and demonstrates strong generalization ability.

2 Related Work

CAPTCHA Benchmarking. Over two decades, CAPTCHAs have evolved from distorted text (Von Ahn et al., 2003) to image-based challenges, which were later broken by machine learning (Hitaj et al., 2020). Modern CAPTCHAs increasingly require logical reasoning and multi-step interaction. Recent benchmarks and methods (Wu et al., 2025; Teoh et al., 2025; Deng et al., 2025) have further highlighted multimodal reasoning and spatial intelligence as central capabilities for CAPTCHA solving. Yet, as shown in Table 1, existing efforts still leave important gaps: some rely mainly on synthetic CAPTCHA without real-world CAPTCHA coverage (Luo et al., 2026), while others use real-world data but do not provide detailed reasoning steps (Teoh et al., 2025; Deng et al., 2025). Real-world commercial CAPTCHAs are especially important for evaluation as they

Table 1: CAPTCHA Benchmark Comparisons.

Benchmark	Real world	Reasoning	Region Consistent	Scale
Open CaptchaWorld (Luo et al., 2026)	✗	✗	✗	225
Halligan (Teoh et al., 2025)	✓	✗	✓	2600
OEDIPUS (Deng et al., 2025)	✓	✗	✗	300
MCA-Bench (Wu et al., 2025)	✗	✗	✗	180000
CAPTCHA-X (Ours)	✓	✓	✓	1839

have already been validated at massive scale by hundreds of millions of users, demonstrating effectiveness under realistic, interactive conditions. By contrast, our CAPTCHA-X is one of the few large-scale real-world CAPTCHA datasets and is uniquely enriched with detailed reasoning steps and protocols, enabling analysis of reasoning dynamics.

Reasoning in Visual CAPTCHA Solving. Reasoning has become a decisive factor in solving modern CAPTCHA. Early VLM-based solvers emphasized perceptual accuracy but failed on tasks requiring spatial inference or multi-step logic (Shi et al., 2021). Later work explored adversarial and cognitive-inspired CAPTCHA designs, showing they follow reasoning capability (Bursztein et al., 2010; Algwil, 2023). Although such research begins to characterize reasoning difficulty with new metrics, it still lacks a comprehensive reasoning protocol and analytics.

3 Method

3.1 Data Collection and Generation Pipeline

Data Collection. To address the limitations of existing benchmarks, we developed our dataset through a systematic data collection pipeline with high-quality, reasoning-step annotations. We collect CAPTCHA data by programmatically interacting with websites using Selenium (Selenium Project, 2026) and PyAutoGUI (Sweigart, 2026), while recording comprehensive mouse actions and screenshots before and after each puzzle.

Grounding Annotation Generation. After solving a CAPTCHA, humans record the click coordinates, which may not fall exactly at the object center. In many prior benchmarks, each target is annotated with only a single point (shown as the black arrow point in Figure 1), and a prediction is considered correct if it falls within a fixed threshold around that point (shown as the black circle). However, this point-threshold scheme does not faithfully reflect the actual valid click region. The true acceptance region is the red area, which can be substantially larger or differently shaped than the thresholded neighborhood around the annotated point. As a result, visually and functionally valid clicks can be incorrectly judged as failures. To address this issue, we manually annotate the full valid acceptance regions using circles or boxes and count a click as correct if it falls within any of these true regions.

Reasoning Steps Generation. To obtain both accurate mouse actions and reasoning annotations, we use LLMs (GPT-5) to generate the reasoning steps, and only then derive the mouse click coordinates strictly following these steps. We choose LLM-based generation because manual annotation is highly labor-intensive, and manually written reasoning steps tend to lack diversity. Concretely, we generate the reasoning steps by employing carefully designed prompts that are (1) goal-directed, clearly describing the task objective and the criteria for identifying the correct regions without revealing any ground-truth click locations, (2) naturally expressed, encouraging concise reasoning steps, and (3) challenging, designed to maximally elicit the model's reasoning ability. (4) follows such prior work on LLM-generated textual annotations (Zhou et al., 2025; Plizzari et al., 2025).

Quality Assurance. To ensure reliability, every generated reasoning step underwent rigorous human verification by four domain experts. Each generated reasoning step was evaluated by four domain experts. The expert agreement reached 98%, and all remaining disagreements were resolved through discussion, yielding 100% consensus in the final annotations. Samples with unresolved disagreement or low scores were discarded. We conduct extensive experiments and statistical analyses in the Appendix B to validate the effectiveness of our reasoning steps. Finally, our benchmark comprises 1,839 CAPTCHA puzzles across seven categories, as shown in Figure 18.

3.2 CAPTCHA Evaluation Framework

To automatically evaluate models' capability in solving CAPTCHAs, we define a comprehensive evaluation metric in our framework. Specifically, our metrics consider both the correctness of actions and the reasoning by comparing with our annotated ground truth. We formalize the answer to a CAPTCHA puzzle as an ordered sequence: $\mathcal{S} = \{(a_1, c_1), (a_2, c_2), \ldots, (a_m, c_m);\, R\}$, where (a_i, c_i) denotes the i-th action and its associated coordinate; $R = \langle r_1, r_2, \ldots, r_k \rangle$ denotes the a sequence of reasoning steps.

Action Accuracy. Our metric measures if the predicted action and coordinate sequence $\{(a_1, c_1), (a_2, c_2), \ldots, (a_N, c_N)\}$ exactly matches the ground-truth sequence in both order and

correctness. Let a_i^* denote the ground-truth action at step i, $(\hat{x}_i, \hat{y}_i)$ denote the predicted coordinate c_i, and $\mathcal{RG}_i$ the corresponding acceptance region. We define sequence-level accuracy as: $\frac{1}{M}\sum_{j=1}^{M} \mathbf{1}\Big(a_i^{(j)} = a_i^{*(j)} \ \wedge \ (\hat{x}_i^{(j)}, \hat{y}_i^{(j)}) \in \mathcal{RG}_i^{(j)}, \ \forall i\Big)$, where M is the total number of CAPTCHA puzzles. Here $\mathbf{1}\{\cdot\}$ returns 1 only if the entire predicted sequence exactly matches the ground truth in both action order and coordinates, and 0 otherwise.

Reasoning Accuracy. To comprehensively evaluate the quality of model-predicted reasoning, we design multiple new metrics for reasoning steps, each motivated by a distinct aspect of reasoning quality. We argue that high-quality reasoning steps should achieve high solving accuracy. **Reasoning Steps:** To measure the granularity of reasoning, we count the number of reasoning steps in the generated textual reasoning. A larger number of steps typically implies a more complex reasoning trajectory, but also indicates reduced reasoning efficiency. **Reasoning Length:** We measure the total number of tokens in the generated reasoning text. In contrast to Reasoning Steps, which capture the structural depth of reasoning, this metric quantifies the overall textual length. **Reasoning Efficiency:** To assess the trade-off between predictive accuracy and reasoning cost, we define an efficiency metric. Let Acc_i denote the accuracy of model i, $\hat{L}_i = L_i/\overline{L}$ is the normalized reasoning length, and $\hat{S}_i = S_i/\overline{S}$ is the normalized reasoning steps. With equal weights $\alpha = \beta = 0.5$, efficiency is computed as $Efficiency_i = \frac{Acc_i}{\alpha \cdot \hat{L}_i + \beta \cdot \hat{S}_i}$. Values are further using min–max normalized to $(0, 1)$. In all, higher reasoning efficiency reflects the model achieving stronger accuracy with fewer steps or tokens, which is more efficient. **Trajectory Complexity Index (TCI).** To quantify the structural complexity of reasoning trajectories, we capture linguistic signals such as backtracking words (*but*, *however*, etc.) and symbolic markers (coordinates, grid indices, etc.). For each instance j in group i, we aggregate feature counts $F_{i,j}$ and normalize them by group-level averages:

$$z_{i,j} = \frac{\sum_F (F_{i,j} - \overline{F}_i)}{0.5 \cdot (s_i/\overline{s}) + 0.5 \cdot (t_i/\overline{t})}. \quad (1)$$

The final TCI is obtained by applying a sigmoid function, which maps the feature values into the normalized range of (0, 1): $TCI_i = \sigma\Big(\frac{1}{N_i}\sum_{j=1}^{N_i} z_{i,j}\Big)$, $\sigma(x) = \frac{1}{1+e^{-x}}$. A higher TCI indicates frequent backtracking or symbolic reasoning, demonstrating more complex reasoning behavior. **Reasoning Score.** To evaluate how well a model's reasoning aligns with the ground-truth reasoning steps, we use human annotations. Specifically, we recruited four domain experts to independently assess the quality of each generated reasoning sequence across multiple sub-dimensions. For each instance i, the final score is obtained by averaging the expert-provided scores. All experts evaluated the samples in a blinded manner, and disagreements greater than two points triggered a second-round review until consensus was reached. We additionally report inter-rater reliability, with Krippendorff's alpha of 0.82 across annotators, indicating substantial agreement. Unresolved cases after review were excluded.

4 Experiments

We conduct experiments to assess the role of reasoning in CAPTCHA solving by comparing model performance with and without reasoning and measuring spatial alignment via L_2 distance in a zero-shot setting(reasoning = model-generated reasoning only). All experiments use a fixed API configuration (temperature $= 0$, seed $= 41$) for reproducibility. We report results in two dimensions: **Action Evaluation**, which measures end-task accuracy, and **Reasoning Evaluation**, which analyzes the quality of intermediate reasoning steps. In addition, we include human performance as a reference point. Human results serve as a natural upper bound for CAPTCHA-solving accuracy and provide a meaningful context for interpreting model performance. Specifically, we recruited 20 participants, each of whom independently solved all our benchmark using the same web UI interface provided to the models. We recorded all trajectories across all participants, which serves as the human baseline.

4.1 Evaluation Analysis

As shown in Table 2, prompting models to generate reasoning steps almost always improves **solving accuracy**. Figure 2 illustrates this trend. Beyond accuracy, our dataset provides region centers to compute L_2 distance between predictions and ground truth. This metric directly measures spatial grounding: smaller distances indicate precise localization. Using both accuracy and L_2 distance yields a more reliable measure of solving quality.

Table 2: Model performance (WR = With Reasoning, WOR = Without Reasoning) across different CAPTCHA types.

Model	Gobang		Icon		Iconcrush		Recaptcha		Space Reasoning		hcaptcha		VTT	
	WR	WOR	WR	WOR	WR	WOR	WR	WOR	WR	WOR	WR	WOR	WR	WOR
Closed Source Models														
GPT-O3	2.00	0.00	22.00	29.79	3.67	3.67	10.67	1.82	10.00	1.50	27.67	0.00	7.00	3.67
GPT-4O	0.00	0.00	9.52	7.48	28.00	23.33	11.00	1.52	47.00	40.00	23.71	1.92	42.00	37.67
GPT-5-Nano	0.00	0.00	0.00	0.00	28.00	23.33	8.33	2.00	31.00	32.00	58.33	40.00	30.67	32.67
Gemini-2.5-Pro	57.00	48.00	59.30	46.30	75.00	66.67	64.00	56.52	68.00	64.67	80.95	81.35	63.00	56.00
Gemini-2.0-Flash	2.00	0.00	36.33	39.67	2.33	2.00	36.33	31.67	53.00	51.00	43.21	0.79	45.67	47.67
Claude-4-Opus	18.00	8.00	17.65	13.00	18.00	6.67	12.33	3.33	29.00	23.33	26.70	0.00	26.67	23.67
Open Source Models														
Qwen-2.5VL-72B	0.00	0.00	0.00	0.00	6.00	5.00	14.00	0.00	24.00	27.67	38.10	36.11	19.33	26.67
Qwen-3VL-8B	0.00	0.00	0.00	0.00	0.00	0.00	51.67	43.33	0.00	0.00	19.84	9.92	0.00	0.00
InternVL3-8B	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R-4B	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ours														
Captcha-X-Agent-O3 (Ours)	39.00	–	80.10	–	93.00	–	69.40	–	96.67	–	91.74	–	79.00	–
Captcha-X-Agent-2.5-Pro (Ours)	67.44	–	78.60	–	92.33	–	73.00	–	98.67	–	94.44	–	80.67	–

Our results show that Gemini-2.5-Pro achieves the smallest L_2 distance among all existing models. In contrast, weaker models such as R-4B produce very large spatial errors. Importantly, our agent obtains the lowest L_2 distance on every CAPTCHA type, even surpassing human performance. These findings demonstrate that L_2 distance provides complementary evidence of spatial grounding beyond accuracy alone. More importantly, the average solving rates on each panel in Figure 2 show that reasoning steps substantially improve final solving success, suggesting that they are not merely a meaningless trick. At the same time, reasoning also reduces spatial localization error, as reflected by the smaller L_2 distance. With reasoning, the strongest VLM (Gemini-2.5-Pro) achieves an average L_2 error that approaches, and in some cases even reaches, human-level performance. **This suggests that reasoning is a crucial factor in CAPTCHA solving**. At the same time, its overall accuracy still remains far below human performance. **However, we observe a counterintuitive phenomenon: If the model can predict coordinates with near-human precision in L2 distance, why does it still get so many questions wrong?** By taking a closer look at this failure case, we find that when VLMs output discrete numerical coordinates for target locations in images, the problem is not that they click inaccurately, but rather that they generate the logically flawed reasoning steps. We summarize this type of failure as **Logical Errors** or **Mapping Errors**. Figure 5 provides detailed examples of both error types. **This naturally raises another deeper question:** if we completely abandon the cross-modal interface for "textual reasoning steps" and instead allow the model to do

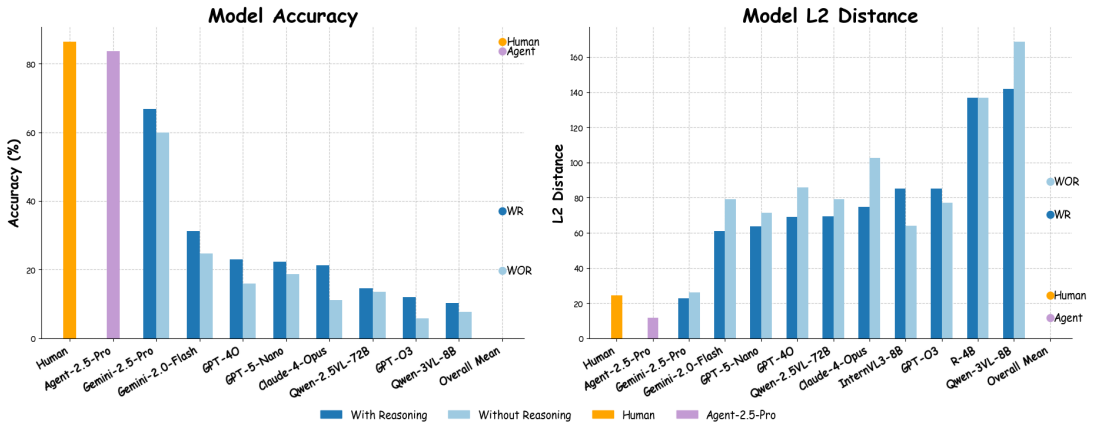


Figure 2: Model Accuracy and L2 Distance with and without Reasoning. Orange bars indicate human performance, while purple bars represent Agent's performance. Blue bars correspond to model results with reasoning (WR) and without reasoning (WOR), respectively. For accuracy, higher values are better, whereas for L2 distance, lower values are better instead. A clear trend emerges: in nearly all models, WR significantly outperforms WOR.

reasoning directly within a purely visual spatial space through **a pure visual generation model**, could it avoid these two types of errors? This question will be addressed more comprehensively in the next section through the reasoning scaling law we observe.

4.2 Reasoning Scaling Law

We observe **Three Reasoning Scaling Laws** across evaluated models: **(1) Linear Scaling Law.** We find a near-perfect linear relationship between reasoning score, reasoning length, and trajectory complexity index (TCI):

$$\begin{aligned} \text{Reasoning Length} &= a_1 \cdot \text{Score} + b_1, \\ \text{TCI} &= a_2 \cdot \text{Score} + b_2, \end{aligned} \tag{2}$$

where a_1, b_1, a_2, b_2 are regression coefficients. Empirically, $a_1 = 78.95$, $b_1 = -62.11$, $a_2 = 0.349$, and $b_2 = -0.333$, all significant at $p < 0.01$. Since reasoning score strongly predicts accuracy ($r = 0.88$), this law establishes a principled connection between reasoning cost and problem-solving ability, enabling accuracy to be directly forecasted from reasoning complexity.

(2) Power Law. We futher observe that accuracy exhibits a clear power-law relationship with the proposed reasoning-efficiency metric, which can be well fitted by

$$\text{Accuracy} = A + k \cdot \text{Eff}^{\alpha} \tag{3}$$

where A denotes the offset, k the scaling coefficient, and α the exponent. Across all evaluated models, the fitted parameters are $A = 11.30$, $k = 70.76$, and $\alpha = 1.90$, yielding $R^2 = 0.828$ ($p < 0.01$). This power-law fit suggests a superlinear trend in which models with higher reasoning efficiency tend to achieve disproportionately higher accuracy, as illustrated in Figure 3.

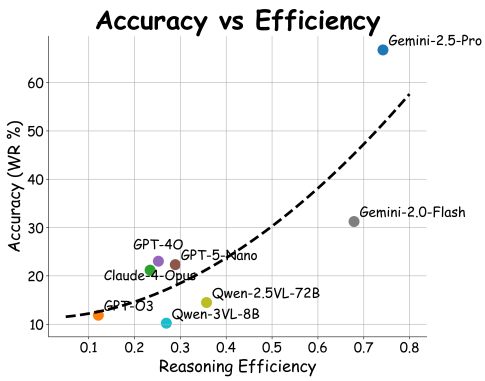


Figure 3: Reasoning Scaling Law.

(3) Difficulty–Gain Scaling Law. Finally, we examine how the benefit of reasoning varies with task difficulty, and we have observed that reasoning becomes more effective as tasks grow harder. To quantify the difficulty level of each CAPTCHA type, we define a difficulty level as

$$\text{Difficulty} = 1 - \overline{\text{Acc}}, \tag{4}$$

where $\overline{\text{Acc}}$ is the average accuracy achieved by our different vision-language models on that CAPTCHA type. This metric reflects the average "solving difficulty" observed across models. When models achieve low accuracy, it indicates that this type of task is hard to solve.

To fairly measure how reasoning contributes under varying levels of difficulty, we adopt the *relative gain* rather than the absolute improvement. Relative gain directly captures how much the reasoning-enhanced performance improves over the non-reasoning baseline. Formally, we compute:

$$\text{Relative Gain} = \frac{\text{WR} - \text{WOR}}{\text{WOR}} \times 100\%. \tag{5}$$

We observed that the relative gain exhibits a clear positive correlation with task difficulty. It is clear that CAPTCHA types with high difficulty yield substantially larger improvements from reasoning, while easier CAPTCHA types benefit only marginally. A linear regression gives $R^2 = 0.85$ ($p < 0.01$), and Spearman's rank correlation yields $\rho = 0.93$ ($p = 0.0025$), confirming a strong and statistically significant monotonic trend.

These findings empirically validate our proposed observations again: reasoning becomes increasingly effective as the problem difficulty rises. This naturally leads to a central question: why does reasoning become more effective as tasks grow harder?

(4) Why is reasoning more effective on harder tasks? We analyze the relationship between task difficulty and the average reasoning length used by the model. Empirically, we observe a strong linear trend between the two quantities:

$$L_{\text{reason}} = \alpha \cdot D_{\text{task}} + \beta, \quad R^2 = 0.92, \tag{6}$$

where L_{reason} denotes the average reasoning length and D_{task} represents the different Captcha-Solving task difficulty. we also obtain that $\alpha = 211.71$ and $\beta = -39.95$. This result shows that as task difficulty increases, models tend to generate longer reasoning sequences. These observations suggest that **harder tasks require more reasoning**

effort, and such increased effort leads to larger performance gains.

4.3 VLM vs. Generation Model?

Table 3: **The Modality Trade-off Evaluation.** Comparison of overall performance across different action space modalities. Generation models achieve near-perfect spatial precision (lowest L_2) but suffer a severe drop in overall accuracy due to logical hallucinations on complex reasoning tasks. Our Agent effectively bridges this gap.

Model / Modality	Overall Acc. (%) ↑	Overall L_2 Dist. ↓
Human Baseline		
Human	**88.5**	23.2
Textual Action Space (Best VLM)		
Gemini-2.5-Pro (WR)	66.8	22.8
Visual Action Space (Generation Models)		
Nano-Banana-Pro*	41.2	**3.2**
GPT-Image-2	36.7	6.4
Qwen-Image-Edit-2511	18.4	18.9

* Note: Officially referred to as Gemini-3-Pro-Image.

As our scaling laws established, complex tasks strictly demand explicit multi-step reasoning. This revisits the question from **Section 4.1**: Can bypassing the textual reasoning entirely resolve Mapping Errors? To investigate this, we evaluate pure visual-generation models as an extreme control group for reasoning conducted solely through vision. As Table 3 shows, this reveals an interesting phenomenon. First, even the generation model can mark which point or circle we should click through generation using an appropriate prompt. Secondly, we observe the phenomenon: generation models excel at spatial localization. For instance, in almost all CAPTCHA types, generation models consistently and precisely draw markers exactly on the intended targets, **almost eliminating the Grounding errors**. **However, there is no free lunch:** this spatial precision incurs a severe cost, leading to what we term **"Logical Hallucination"**. To illustrate, consider the multi-step Gobang CAPTCHA: despite explicit prompts instructing the model to preserve the original image structure, the generation model frequently hallucinates altered visual states to force a solution. For example, it arbitrarily changes the color of a yellow piece at row 2, column 4, simply to artificially create a winning sequence in the original picture, as shown in Figure 5.

4.4 Failure Case Analysis

Across our experiments, we identify three representative classes of failure in VLMs. **(1) Mapping errors**: the model misinterprets the underlying grid structure (e.g., treating a 5×5 Gobang CAPTCHA as a 3×3 or 8×8 grid), accounting for approximately 45% of all failures and constituting the dominant error type. **(2) Grounding errors**: even when the reasoning process is largely correct, the model fails to output an accurate final coordinate, contributing about 35% of failures. **(3) Logical errors**: the model produces incoherent or self-contradictory reasoning steps, representing only around 20% of failures. To facilitate this analysis, we further develop a lightweight diagnostic tool that can quickly identify and automatically categorize these three error types based on their observed failure patterns. This distribution strongly corroborates our *Modality Trade-off* analysis: the VLM is already highly capable of rigorous logical planning, as evidenced by the relatively small proportion of logical failures (20%), while its overall reasoning performance is predominantly corrupted during the cross-modal execution stage, with mapping and grounding errors jointly accounting for

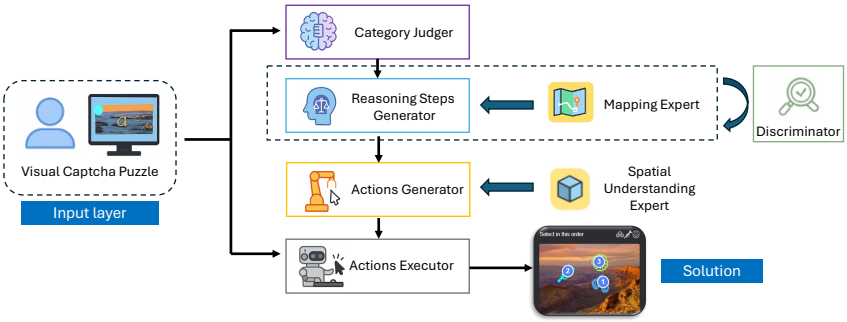


Figure 4: The **Mapping Expert** corrects grid-structure misunderstandings, and the **Spatial Understanding Expert** refine spatial localization to eliminate grounding errors. This design not only substantially improves performance over the direct-prediction baseline but also effectively resolves the major error types.

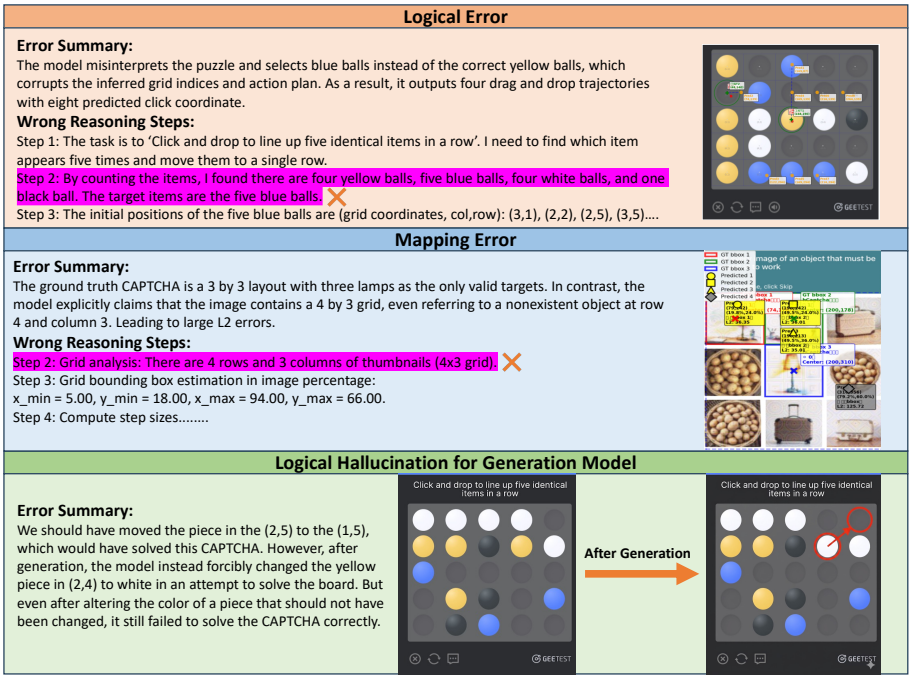


Figure 5: **Fail Cases**: three representative failure types, the first two reflect Logical Errors and Mapping Errors in VLM solving, while the third shows Logical Hallucination in generative models.

roughly 80% of all failures. To ensure logical consistency, our agent uses a **Discriminator** to validate that the generated reasoning is coherent before passing it forward. The validated reasoning is then handled by an **Action Generator**, which translates reasoning outputs into executable click coordinates. Finally, an **Action Executor** performs the actual clicks on the screen to solve the CAPTCHA. By explicitly distinguishing between grid-based and non-grid-based categories, this unified framework highlights the central role of reasoning in solving diverse visual CAPTCHA.

4.5 Agent Evaluation

The baseline uses Gemini-2.5-Pro without reasoning, where the model directly outputs click coordinates from the CAPTCHA image. Our agent achieves state-of-the-art performance on five of the seven tasks (Table 4). To assess generalization, we further test our agent on external CAPTCHA types from the Open CaptchaWorld (Luo et al., 2026) benchmark(the orange section in the table). These tasks are not part of our dataset and therefore represent a clear out of distribution setting. Our agent achieves 100 percent on Geometry, 85 percent on Click Order, and 90 percent on Animal, while the benchmark itself reports a highest average accuracy of only 40 percent. These results show that our agent enables strong transfer across unseen CAPTCHA formats. The ablation study of our agent pipeline can be found in the Appendix.

5 Conclusion

Our work shows that reasoning is a decisive capability for solving modern visual CAPTCHA. With our benchmark, we pair real-world CAPTCHA challenges with reasoning steps and introduce reasoning-oriented metrics. These findings highlight reasoning as central to advancing multimodal and AI safety in the future.

Table 4: Comparison of CAPTCHA Solving Accuracy for Different CAPTCHA Solvers

Model	Icon	Space Reasoning	VTT	Iconcrush	hCaptcha	Gobang	Recaptcha	Geometry	Click Order	Animal
Baseline Models										
Baseline	46.3	64.67	50.00	66.7	0	48	56.52	60	55	43.33
OEDIPUS-DSL (Deng et al., 2025)	–	65.4	–	67.4	–	80.2	–	–	–	–
Halligan (Teoh et al., 2025)	46	–	23	98	82	92	68	–	–	–
VTTsolver (Gao et al., 2021)	–	90.8	50	–	–	–	–	–	–	–
PhishDecloaker (Teoh et al., 2024)	–	–	–	–	74	–	72	–	–	–
Ours										
Captcha-X-Agent (Ours)	80.1	98.67	80.67	93	94.44	67.44	73	100	85	90

Limitations

Despite being the first benchmark to systematically evaluate the reasoning performance of both VLMs and generative models on CAPTCHA solving, our study does not design a CAPTCHA-specific generative model. Instead, we evaluate existing general-purpose models and show that current generative models remain limited for direct CAPTCHA solving. This suggests that generation model-based agents may currently be a more promising direction. More broadly, our pipeline only provides a simple reasoning-based framework, and future work could develop specialized model architectures or pipelines explicitly optimized for CAPTCHA reasoning and solving.

Ethics Considerations

This work studies CAPTCHA as a benchmark for visual-spatial reasoning and model robustness, rather than as a tool for bypassing authentication systems. We take misuse risks seriously and design the release protocol accordingly. CAPTCHA-X will only release processed benchmark images, anonymized action annotations, and reasoning labels. We will not release interaction scripts, website-specific metadata, crawling pipelines, or deployment details that could facilitate automated abuse. All data collection and annotation are conducted without personal identifiers and follow responsible-use principles. The intended use of CAPTCHA-X is to evaluate multimodal reasoning, spatial grounding, and failure modes of vision-language and generative models. We explicitly discourage using this benchmark to attack, automate, or weaken real-world security systems.

References

Abdalnaser Muhammad Algwil. 2023. A survey on captcha: Origin, applications and classification. , 36(1):1–37.

Elie Bursztein, Steven Bethard, Celine Fabry, John C Mitchell, and Dan Jurafsky. 2010. How good are humans at solving captchas? a large scale evaluation. In *2010 IEEE symposium on security and privacy*, pages 399–413. IEEE.

Gelei Deng, Haoran Ou, Yi Liu, Jie Zhang, Tianwei Zhang, and Yang Liu. 2025. Oedipus: Llm-enchanced reasoning captcha solver. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 6–20.

Ziqi Ding, Gelei Deng, Yi Liu, Junchen Ding, Jieshan Chen, Yulei Sui, and Yuekang Li. 2025. Illusioncaptcha: A captcha based on visual illusion. In *Proceedings of the ACM on Web Conference 2025*, pages 3683–3691.

Yipeng Gao, Haichang Gao, Sainan Luo, Yang Zi, Shudong Zhang, Wenjie Mao, Ping Wang, Yulong Shen, and Jeff Yan. 2021. Research on the security of visual reasoning {CAPTCHA}. In *30th USENIX security symposium (USENIX security 21)*, pages 3291–3308.

Dorjan Hitaj, Briland Hitaj, Sushil Jajodia, and Luigi V Mancini. 2020. Capture the bot: Using adversarial examples to improve captcha robustness to bot attacks. *IEEE Intelligent Systems*, 36(5):104–112.

Fangyu Liu, Guy Emerson, and Nigel Collier. 2023. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651.

Yaxin Luo, Zhaoyi Li, Jiacheng Liu, Jiacheng Cui, Xiaohan Zhao, and Zhiqiang Shen. 2026. Open captcha-world: A comprehensive web-based platform for testing and benchmarking multimodal llm agents. *Advances in Neural Information Processing Systems*, 38.

Chiara Plizzari, Alessio Tonioni, Yongqin Xian, Achin Kulshrestha, and Federico Tombari. 2025. Omnia de egotempo: Benchmarking temporal understanding of multi-modal llms in egocentric videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24129–24138.

Selenium Project. 2026. Selenium WebDriver. https://www.selenium.dev/documentation/webdriver/. Official documentation.

Chenghui Shi, Xiaogang Xu, Shouling Ji, Kai Bu, Jianhai Chen, Raheem Beyah, and Ting Wang. 2021. Adversarial captchas. *IEEE transactions on cybernetics*, 52(7):6095–6108.

Al Sweigart. 2026. PyAutoGUI. https://pyautogui.readthedocs.io/. Official documentation.

Xiwen Teoh, Yun Lin, Ruofan Liu, Zhiyong Huang, and Jin Song Dong. 2024. {PhishDecloaker}: Detecting {CAPTCHA-cloaked} phishing websites via hybrid vision-based interactive models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 505–522.

Xiwen Teoh, Yun Lin, Ruofan Liu, Zhiyong Huang, and Jin Song Dong. 2025. Bot-hard: Making captchas resilient against advanced solvers. In *Proceedings of the 34th USENIX Security Symposium, pp. xxx–xxx. USENIX Association*.

Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. 2003. Captcha: Using hard ai problems for security. In *International conference on the theory and applications of cryptographic techniques*, pages 294–311. Springer.

Haipeng Wang, Feng Zheng, Zhuoming Chen, Yi Lu, Jing Gao, and Renjia Wei. 2018. A captcha design based on visual reasoning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1967–1971. IEEE.

Zongyu Wu, Ming Zhang, Yifan Li, Hao Chen, and Wei Xu Mca-bench. 2025. A multimodal captcha benchmark for evaluating reasoning in vision-language models. *arXiv preprint arXiv:2506.05982*, 1(2).

Sheng Zhou, Junbin Xiao, Qingyun Li, Yicong Li, Xun Yang, Dan Guo, Meng Wang, Tat-Seng Chua, and Angela Yao. 2025. Egotextvqa: Towards egocentric scene-text aware video question answering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3363–3373.

Appendix Overview

In this appendix, we provide an extensive set of additional analyses that expand upon the main paper. We begin with a detailed examination of our reasoning-centric agentic architecture, including conceptual comparisons with prior CAPTCHA solvers and comprehensive ablation studies that clarify the functional role of each module. We then present deeper evaluations of the benchmark itself, including verification of human-annotated reasoning steps, permutation tests demonstrating their structural validity, correlation analyses on model-generated reasoning steps, and systematic failure case studies covering logical, mapping, and grounding errors. Finally, we include the full experimental setup, model configurations, and evaluation protocols to support complete reproducibility.

REF.	DESCRIPTION
§A	**Agent Architecture Studies:** conceptual comparison with existing agentic CAPTCHA solvers and ablation studies analyzing the contribution of each reasoning and grounding module.
§B	**Reasoning Annotation Verification:** human verification statistics, evaluation-independence clarification, and permutation tests validating the reliability and instance-specific structure of ground-truth reasoning steps.
§C	**Correlation Analysis:** relationships between model-generated reasoning metrics and final CAPTCHA-solving accuracy, including reasoning length, reasoning steps, reasoning score, efficiency, and trajectory complexity.
§D	**Failure Case Analysis:** representative examples of logical errors, mapping errors, grounding errors, and generation-model hallucinations across different CAPTCHA types.
§E	**Experimental Details:** implementation settings, model configurations, prompting protocols, evaluation criteria, and additional materials supporting reproducibility.

A Studies on the Vision-Language Agent Architecture

Due to space constraints in the main paper, we provide a detailed analysis of our agentic framework here. While the core methodology and performance results are presented in the main paper, this appendix offers: (1) a comprehensive conceptual comparison with existing agentic CAPTCHA solvers (Section A.1), and (2) extensive ablation studies across different backbone models (Section A.2). These materials provide essential context for understanding our agent's reasoning-centric paradigm and its differentiation from prior work.

A.1 Conceptual Differences from Existing Agentic CAPTCHA Solvers

Existing agentic CAPTCHA solvers can be grouped into two major categories. The first category includes DSL based symbolic systems such as OEDIPUS. These approaches depend on a domain specific language, a symbolic parsing toolchain and a fine tuned model that is trained directly on their own benchmark. This dependency restricts the solver to a fixed task structure and makes the system highly sensitive to dataset format. As a result, the agent does not generalize to new CAPTCHA styles that fall outside the DSL specification or the distribution seen during fine tuning.

The second category includes tool driven search pipelines such as Halligan. These systems require several external components, for example GroundingDINO, OCR engines and auxiliary detectors. Their performance relies on the quality of these tools and on dataset specific calibration, which makes cross dataset robustness very limited. Similar to symbolic systems, they lack mechanisms for identifying or correcting different types of reasoning failures.

In contrast, our agent follows a reasoning driven and VLM only paradigm that does not require symbolic parsing, external detectors or dataset specific fine tuning. The agent performs multi step reasoning entirely within the VLM, and this internal reasoning signals and guides the subsequent action generation process. Our framework introduces the first explicit decomposition of reasoning failures in CAPTCHA solving. Logic errors, mapping errors and grounding errors are detected and corrected by specialized expert modules, forming an error aware reasoning loop that is not present in previous systems.

Overall, our framework represents a fundamentally different agentic paradigm. It is reasoning driven, VLM first, tool free, free of dataset specific fine tuning, error aware and suitable for cross dataset general purpose CAPTCHA solving.

Table 5: High-level comparison of agentic CAPTCHA solvers. Our agent is the only **VLM-only**, **Tool and Fine-tune free**, and **Cross-dataset** solver.

Dimension	OEDIPUS	Halligan	Ours
Core Paradigm	Symbolic Solver	Tool-Driven Search	**Reasoning-Centric**
Without External Tools and Fine-tune	✗	✗	✓
Error Decomposition	✗	✗	✓
OOD Transferability	✗	✗	✓
VLM-only	✗	✗	✓

A.2 Ablation Studies

To better understand the contribution of each component in our agentic architecture, we conduct a systematic ablation study. Our agent consists of several modules that jointly enable reasoning-driven and spatially grounded click predictions. The Category Judger guides task routing and ensures that prompts are specialized for different CAPTCHA types. The Reasoning Steps Generator produces structured intermediate reasoning that supports consistent spatial decomposition. The Mapping Expert and the Spatial Understanding Expert refine global and local spatial grounding respectively, while the Discriminator suppresses logically inconsistent or low-quality reasoning traces. Since these modules interact in a nontrivial manner, it is important to quantify their individual influence on final performance and to verify whether the overall gains indeed originate from improved reasoning and spatial alignment rather than incidental prompt design. The following ablation results clarify the functional role of each component and demonstrate that the full performance of our agent emerges only when all reasoning and grounding stages operate together.

The ablation experiments conducted on both Gemini-2.5-Pro and GPT-O3 reveal clear and consistent trends that highlight the functional role of each component in our agentic architecture.

Removing the Category Judger results in the most severe performance drop, with accuracy falling to around forty to fifty percent. This happens because the Category Judger is the module that tells the system what kind of CAPTCHA it is dealing with and directs the input to the right reasoning pathway. In particular, grid-based CAPTCHAs need a very different prompting strategy: the prompt must explicitly tell the Reasoning Generator to calculate the center of the target grid cell based on the number of rows and columns. Our experiments show that this grid-aware prompting is crucial, because for grid tasks, the model often thinks through the logic correctly but still fails to ground the answer at the right location. Without the Category Judger, the model loses this specialization and ends up using generic reasoning on tasks that require precise geometric grounding, which leads to a surge in grounding errors even when the reasoning is correct. When the Category Judger is kept, especially together with the Spatial Understanding Expert, the model regains strong grounding ability and significantly improves its CAPTCHA-solving performance. Overall, these results show that correctly identifying the task type is not just a minor detail, it is also essential for allowing the full reasoning and grounding pipeline to work as intended.

The Spatial Understanding Expert is similarly critical. When this module is removed, the agent loses its ability to refine fine-grained localization, which leads to a sharp decline in accuracy and a substantial increase in L2 errors.

The Mapping Expert plays a moderately important role. Without it, the agent still retains global reasoning ability but struggles with accurate coordinate alignment, resulting in a notable but less catastrophic reduction in performance. The Reasoning Steps Generator has a comparatively smaller effect on accuracy, although the reasoning score collapses to zero. This suggests that the agent can still make partial progress on visual grounding even without explicit intermediate reasoning, but the overall spatial consistency becomes weaker. The Discriminator contributes the smallest performance drop. It improves robustness by filtering logically inconsistent reasoning traces, but its absence does not critically impair the agent's ability to perform the task.

Collectively, the results demonstrate that the full performance of our agent emerges from the coordinated interaction between category-level task routing, structured reasoning, and multi-stage spatial grounding. The consistency of ablation trends across two different backbone models further confirms that our design is not tied to a specific VLM,

Table 6: Ablation study of the proposed Captcha-X-Agent using **Gemini-2.5-Pro**. Accuracy (%), reasoning score (0–5), and L2 (pixels).

Setting	Accuracy ↑	Reasoning Score ↑	L2 Distance ↓
Full Agent (Gemini-2.5-Pro)	83.9	4.15	10.8
w/o Category Judger	48.3	3.92	25.6
w/o Spatial Understanding Expert	51.2	3.10	27.4
w/o Mapping Expert	63.7	4.12	19.3
w/o Reasoning Steps Generator	70.8	0.00	18.9
w/o Discriminator	75.6	3.70	14.4

Table 7: Ablation study of the proposed Captcha-X-Agent using **GPT-O3**. Accuracy (%), reasoning score (0–5), and L2 (pixels).

Setting	Accuracy ↑	Reasoning Score ↑	L2 Distance ↓
Full Agent (GPT-O3)	78.4	4.32	9.7
w/o Category Judger	45.1	4.10	23.8
w/o Spatial Understanding Expert	48.7	3.45	25.9
w/o Mapping Expert	60.3	4.22	18.4
w/o Reasoning Steps Generator	66.9	0.00	17.6
w/o Discriminator	71.4	3.85	13.8

and that the proposed architecture provides stable and transferable gains across diverse CAPTCHA scenarios.

B Verification of Reasoning Step Annotations

To ensure that our benchmark provides trustworthy and structurally meaningful reasoning supervision, we conduct a series of analyses to evaluate the quality, consistency, and instance-specific validity of the ground-truth reasoning annotations. This section presents both human verification statistics and a permutation-based test demonstrating that the GT reasoning steps reflect real task structure rather than superficial correctness.

B.1 Clarification on Evaluation Independence

A natural concern is that using model generated reasoning both to construct ground truth annotations and to evaluate With Reasoning performance might create a form of circular validation. Here we clarify that this concern does not apply to our setting, because the prompting used in evaluation and the reasoning used for annotation are fully decoupled.

The With Reasoning prompt used in our evaluation is designed entirely independently of the model generated ground truth reasoning annotations. The template shown in Appendix E is a general chain of thought instruction that requires the model to reason step by step, identify relevant objects, and then output the required action sequence. It does not reference, imitate, or follow the structure, phrasing, or style of any ground truth reasoning annotation. This prompt was finalized before any reasoning annotations were produced, and the two components were developed as separate parts of our pipeline.

The reasoning generated by each evaluated model in the With Reasoning setting arises solely from this fixed prompt. During evaluation, models never access the ground truth reasoning, and their outputs are not compared against those annotations. The performance comparison between the With Reasoning and Without Reasoning settings therefore measures only whether a model's own internally generated reasoning improves its spatial prediction accuracy. None of the evaluation steps involve matching, aligning, or adapting to the model generated ground truth annotations, and no part of the observed performance gain can be attributed to them.

This strict separation ensures that the improvement achieved in the With Reasoning setting is driven entirely by each model's inherent reasoning ability, rather than by any exposure to or influence from the model generated annotations. For completeness, Table 8 summarizes that all reported evaluation metrics are independent of the ground truth reasoning annotations and thus are free from circular evaluation

Table 8: Independence of evaluation metrics from model-generated ground-truth reasoning annotations.

Evaluation Metric	Uses GT Reasoning Annotations?	Circularity Risk
Action Accuracy	✗	✗
L2 Distance	✗	✗
Statistical Tests (McNemar, Wilcoxon)	✗	✗
Reasoning Steps (count)	✗	✗
Reasoning Length (tokens)	✗	✗
Reasoning Efficiency	✗	✗
TCI (trajectory complexity)	✗	✗
Reasoning Score (human-rated)	✗	✗

B.2 Human Verification Quality Analysis

To evaluate the reliability of the reasoning annotations in our benchmark, we analyze the full set of human-verified samples. The four annotators exhibit high agreement, with a Krippendorff's alpha of 0.82. Consistent with the setup in the main paper, **98%** of samples satisfy our agreement criterion without requiring further adjudication, and the remaining cases are resolved through joint discussion. Human reviewers also actively corrected model-generated reasoning, with **42.6%** of samples receiving at least one edit, while the rest were explicitly confirmed as acceptable without modification or being discarded. These statistics confirm that all released reasoning labels result from expert verification, rather than unreviewed adoption of raw LLM outputs.

Table 9: Key human verification statistics for reasoning annotations.

Metric	Value
Inter-annotator agreement (Krippendorff's alpha)	0.82
Samples meeting agreement threshold	98%
Samples with human edits	42.6%

B.3 Permutation Test on Ground-Truth Reasoning Steps

A key requirement of our benchmark is that the provided GT reasoning steps must serve as reliable and structurally meaningful reference annotations. Expert verification ensures their correctness, but does not guarantee that the GT reasoning steps truly capture the underlying task structure of each puzzle. In principle, a set of annotations could be uniformly correct yet fail to differentiate between easy and hard instances, or between distinct visual reasoning patterns. To establish that our GT reasoning is not only correct but also instance-specific and non-exchangeable, we perform a permutation test.

The core idea is to check whether the quality of GT reasoning steps varies systematically with the actual difficulty of each puzzle. If GT reasoning genuinely reflects task structure, its quality score should align with instance difficulty (measured via solver success). If we randomly reassign GT reasoning scores to different puzzles and the correlation collapses, this indicates that GT reasoning steps carry real, instance-specific semantic information rather than superficial correctness.

To support this analysis, we first assign each GT reasoning a content-based quality score using a strong LLM. Importantly, this score evaluates the reasoning itself (its logical soundness, correctness, and alignment with puzzle semantics), not any predicted actions or model outputs. Below is the prompt used to assess GT reasoning quality.

A central requirement of our benchmark is that the provided ground-truth reasoning steps must serve as *reliable and structurally meaningful* reference annotations. While expert verification ensures their correctness, it does not guarantee that the GT reasoning steps truly encode the underlying task structure of each puzzle. In principle, a set of annotations could be uniformly correct yet fail to differentiate between easy and hard instances. To establish that our GT reasoning is not only correct but also *instance-specific* and *non-exchangeable*, we perform a permutation test.

Following the definition used in the main paper (Sec. 4.1), we quantify the difficulty of each instance using a strong solver's binary success label:

$$A_i = \begin{cases} 1, & \text{if the solver succeeds on instance } i, \\ 0, & \text{otherwise.} \end{cases}$$

Prior experiments in the main paper demonstrate consistent difficulty variation across CAPTCHA types for state-of-the-art solvers such as Gemini-2.5-Pro, GPT-O3, and Qwen-VL, making A_i a reliable and independent proxy for instance difficulty.

Prompt for Permutation Reasoning Test Evaluation.
You are an expert CAPTCHA reasoning evaluator.
Your task is to judge the quality of the ground-truth reasoning steps, not to solve the problem.
Quality means correctness, logical soundness, and whether the reasoning matches the visual description of the puzzle.
You are given a CAPTCHA image description and its ground-truth reasoning steps.
Please score the reasoning quality from 0 to 5, using the following rubric:

- **5** — Perfect: reasoning is fully correct, matches the puzzle semantics, identifies the right regions, and follows a clear multi-step logic.
- **4** — Mostly correct: minor unclear phrases but overall correct and logically valid.
- **3** — Partially correct: some correct steps, but missing details or slightly misaligned with the puzzle.
- **2** — Poor: reasoning contains multiple mistakes, unclear logic, or incorrect interpretations.
- **1** — Incorrect: reasoning contradicts the puzzle or leads to the wrong target.
- **0** — Nonsensical or irrelevant.

Important:

- You are evaluating the reasoning steps only, not the click coordinates.
- Do not rescore based on formatting or writing style; evaluate only semantic correctness.

Output only a number from 0 to 5.
Input Format:
GT Reasoning Steps:
{insert GT reasoning as a list of steps}
Output Format:
Score: X

Figure 6: Prompt for permutation reasoning test evaluation.

To measure whether GT reasoning reflects this difficulty, we assign each GT reasoning step a *content-based quality score* using a strong LLM, evaluating correctness, logical soundness, and semantic alignment with the puzzle. This score is independent of any predicted actions or model outputs.

Observed correlation. Using the correct pairing between each puzzle and its GT reasoning score (S_i, A_i), we compute the Pearson correlation:

$$r_{\text{obs}} = \text{corr}(S_i, A_i).$$

We observe a strong positive correlation:

$$r_{\text{obs}} = 0.42,$$

indicating that puzzles which are empirically difficult for solvers tend to have GT reasoning that is naturally longer, more structured, and semantically richer.

Permutation distribution. To determine whether this alignment is structural rather than accidental, we perform $10{,}000$ random permutations of GT reasoning scores across puzzles:

$$r^{(k)} = \text{corr}(S_{\pi_k(i)}, A_i),$$

where π_k denotes a random permutation. The distribution of permuted correlations centers around zero:

$$\mathbb{E}[r^{(k)}] \approx 0.01, \qquad \text{std}(r^{(k)}) \approx 0.03.$$

Significance. The empirical p-value is:

$$p = \frac{1}{10000} \sum_{k=1}^{10000} \mathbb{1}\big(|r^{(k)}| \geq |r_{\text{obs}}|\big) < 10^{-3},$$

showing that the observed correlation cannot be explained by random assignment.

Conclusion. These results demonstrate that our GT reasoning steps are *not* interchangeable across instances. Their quality scores correlate strongly with empirically measured difficulty, and this relationship collapses when the pairing is randomized. Therefore, our GT reasoning annotations carry genuine instance-specific structural information, validating them as reliable ground-truth references for evaluating reasoning quality.

C Correlation Analysis

While Appendix B focuses on validating the correctness and instance specificity of our GT reasoning annotations, it is also important to study how the proposed reasoning metrics behave when they are produced by real models rather than by the ground truth. The metrics in this section are computed from the reasoning traces generated by various large vision language models that attempt to solve our CAPTCHA benchmark. These metrics therefore capture the reasoning behavior of actual solvers, not the GT annotations. To understand whether these model generated reasoning signals are meaningful, we examine their relationships with the corresponding solver accuracy across instances. If the metrics truly reflect reasoning quality, they should show consistent and interpretable correlations with model performance. The correlation heatmap presented below summarizes this analysis.

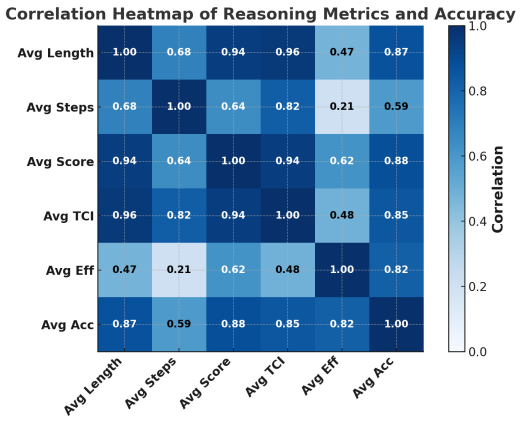


Figure 7: Correlation heatmap of model reasoning metrics and solver accuracy. The metrics are computed from model generated reasoning traces and summarize their relationship with final task performance.

The correlation analysis reveals several clear patterns. First, metrics that reflect the structural richness of model reasoning, including Length, Steps, Score, and TCI, exhibit strong positive correlations with solver accuracy. This indicates that models tend to succeed when their reasoning traces are longer, more explicit, and more logically structured. Second, Efficiency shows a moderate correlation with accuracy, suggesting that concise reasoning is helpful only when it preserves essential task information. Finally, all reasoning metrics demonstrate interpretable relationships with accuracy rather than behaving as arbitrary numerical outputs, which confirms that our metric design captures meaningful aspects of model reasoning behavior.

D Failure Case Analysis

Across all models, we observe that errors concentrate in three recurring patterns with clear differences in prevalence. Grounding errors constitute the largest portion at about 35 percent of all errors, logical errors follow at approximately 20 percent, and mapping errors account for 45 percent. These proportions motivate the case studies presented in this appendix.

D.1 Logical Errors

We present a representative failure case that illustrates how an early misunderstanding in visual reasoning can propagate through the entire decision process. In this example, the model misidentifies the target object category and builds a complete multi step plan on an incorrect premise. This leads to incorrect grid indexing, an invalid sequence of actions, and a final set of predicted coordinates that diverge entirely from the ground truth.

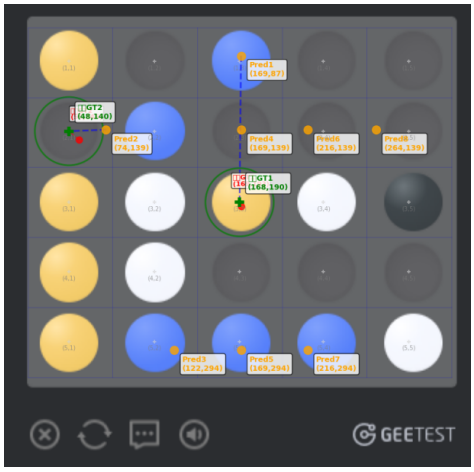


Figure 8: Failure case on a Gobang CAPTCHA. The model first misinterprets the puzzle and selects blue balls instead of the correct yellow balls, which corrupts the inferred grid indices and action plan. As a result, it outputs four drag and drop trajectories with eight predicted click coordinates, whereas the ground truth solution requires moving only a single yellow ball with one correct click trajectory.

The model's reasoning is incorrect because it fails at the most fundamental stage of semantic understanding. The puzzle contains five yellow balls that must be aligned in order to complete the task. However, the model mistakenly concludes that the target items are five blue balls. This single

Example Reasoning Output (Drag-and-Drop, With Reasoning).


```
{
  "reasoning": [
    "Step 1: The task is to 'Click and drop to line up five identical items in a row'. I need to find which item appears
    ↪  five times and move them to a single row.",
    "Step 2: By counting the items, I found there are four yellow balls, five blue balls, four white balls, and one black
    ↪  ball. The target items are the five blue balls.",
    "Step 3: The initial positions of the five blue balls are (grid coordinates, col,row): (3,1), (2,2), (2,5), (3,5),
    ↪  and (4,5).",
    "Step 4: I will move all the blue balls to row 2. This row already contains one blue ball at (2,2) and has four empty
    ↪  slots, which is enough for the other four blue balls.",
    "Step 5: The plan is to perform four drag-and-drop actions: move the ball from (3,1) to (1,2), from (2,5) to (3,2),
    ↪  from (3,5) to (4,2), and from (4,5) to (5,2).",
    "Step 6: I will now calculate the coordinates. The grid is 5x5, so `n_rows=5` and `n_cols=5`. I estimate the grid's
    ↪  bounding box as `[15.0, 16.0, 85.0, 83.0]` in percentage of the image.",
    "Step 7: The step sizes are computed: `x_step = (85.0 - 15.0) / 5 = 14.0` and `y_step = (83.0 - 16.0) / 5 = 13.4`.",
    "Step 8: First drag-and-drop from (3,1) to (1,2). Start: `x = 15.0 + (3 - 0.5)*14.0 = 50.00`, `y = 16.0 + (1 -
    ↪  0.5)*13.4 = 22.70`. End: `x = 22.00`, `y = 36.10`.",
    "Step 9: Second drag-and-drop from (2,5) to (3,2). Start: `x = 36.00`, `y = 76.30`. End: `x = 50.00`, `y = 36.10`.",
    "Step 10: Third drag-and-drop from (3,5) to (4,2). Start: `x = 50.00`, `y = 76.30`. End: `x = 64.00`, `y = 36.10`.",
    "Step 11: Fourth drag-and-drop from (4,5) to (5,2). Start: `x = 64.00`, `y = 76.30`. End: `x = 78.00`, `y = 36.10`."
  ],

  "mouseActions": [
    {"action": "move",      "coords": {"x": 50.0, "y": 22.7}},
    {"action": "mousedown", "coords": {"x": 50.0, "y": 22.7}},
    {"action": "move",      "coords": {"x": 22.0, "y": 36.1}},
    {"action": "mouseup",   "coords": {"x": 22.0, "y": 36.1}},

    {"action": "move",      "coords": {"x": 36.0, "y": 76.3}},
    {"action": "mousedown", "coords": {"x": 36.0, "y": 76.3}},
    {"action": "move",      "coords": {"x": 50.0, "y": 36.1}},
    {"action": "mouseup",   "coords": {"x": 50.0, "y": 36.1}},

    {"action": "move",      "coords": {"x": 50.0, "y": 76.3}},
    {"action": "mousedown", "coords": {"x": 50.0, "y": 76.3}},
    {"action": "move",      "coords": {"x": 64.0, "y": 36.1}},
    {"action": "mouseup",   "coords": {"x": 64.0, "y": 36.1}},

    {"action": "move",      "coords": {"x": 64.0, "y": 76.3}},
    {"action": "mousedown", "coords": {"x": 64.0, "y": 76.3}},
    {"action": "move",      "coords": {"x": 78.0, "y": 36.1}},
    {"action": "mouseup",   "coords": {"x": 78.0, "y": 36.1}}
  ]
}
```


Figure 9: Example reasoning output for a drag-and-drop CAPTCHA in the with-reasoning setting.

mistake causes all subsequent reasoning steps to be built on an incorrect premise.

First, the model misidentifies the target color category and asserts that there are five blue balls in the puzzle. This is factually incorrect. Since the model is now searching for blue items, it also extracts the wrong grid locations. The cell coordinates it lists as the positions of the blue balls correspond to completely different objects in the image. As a result, every later decision, including the choice of a target row and the plan for object movement, is based on inaccurate visual grounding.

This leads the model to generate an entirely incorrect high level plan that contains four drag and drop actions. In contrast, the ground truth requires only a single move of one yellow ball to create a vertical line. The model then proceeds to compute cell center coordinates and bounding box geometry. These computations are numerically consistent, but all results correspond to the wrong objects because the model began with an incorrect interpretation of the puzzle.

Therefore, a single perception mistake at the beginning produces a cascading failure in every later step of the reasoning chain. As a result, the model outputs eight incorrect coordinates since it plans four drag paths with a start point and an end point for each movement. This explains why the predicted click positions deviate significantly from the ground truth and why the L2 distance becomes extremely large.

D.2 Mapping Errors

This example illustrates a typical mapping error in which the model constructs its entire solution on an incorrect grid representation. The ground truth CAPTCHA is a 3 by 3 layout with three lamps as the only valid targets, while the remaining cells contain suitcases or baskets that do not require electrical power. In contrast, the model explicitly

Model Reasoning (Part of the Mapping Error Output).

```
Step 2: Grid analysis: There are 4 rows and 3 columns of thumbnails (4x3 grid).
Step 3: Grid bounding box estimation in image percentage:
        x_min = 5.00, y_min = 18.00, x_max = 94.00, y_max = 66.00.
Step 4: Compute step sizes:
        x_step = (94.00 - 5.00) / 3 = 29.67;
        y_step = (66.00 - 18.00) / 4 = 12.00.
Step 6: Target coordinates (in percent of image) for the clickable objects:
        - Lamp at row1, col1: (19.83, 24.00)
        - Lamp at row1, col2: (49.50, 24.00)
        - Lamp at row2, col2: (49.50, 36.00)
        - Toaster at row4, col3: (79.17, 60.00)
```

Figure 10: Example model reasoning showing a mapping error.

claims that the image contains a 4 by 3 grid, computes the vertical step size with four rows instead of three, and even refers to a nonexistent object at row 4 and column 3. As a result, its coordinate mapping is globally misaligned with the true grid, one lamp is never clicked, and an extra non target object is selected, leading to large L2 errors and degraded precision and recall despite the reasoning sounding locally coherent.

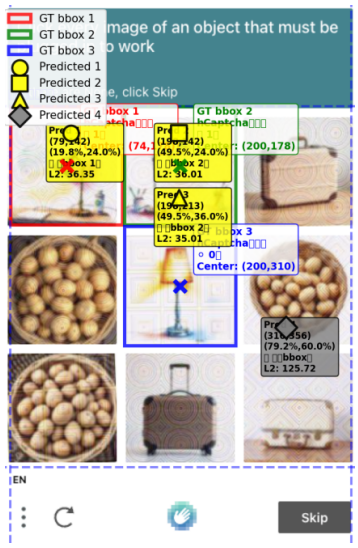


Figure 11: Mapping Error for hCaptcha.

D.3 Grounding Errors

Grounding errors occur when the model identifies the correct objects and produces a reasonable high level plan, but fails to map these decisions to accurate spatial coordinates in the final action sequence. In these cases, the reasoning is often coherent and the target category is correctly recognized, yet the predicted click locations deviate significantly from the true object centers. Such errors typically arise from inaccurate bounding box estimation, misaligned step size computation, or numerical drift in coordinate calculations, which leads to large L2 distances and incorrect actions even when the overall logic is sound.

E Prompt Templates With and Without Reasoning

This section presents the prompt templates used by the model under both the with-reasoning and without-reasoning settings. We begin by outlining the basic template structure, followed by concrete examples.

E.1 With Reasoning Prompt Template

We use this prompt by pairing it directly with the corresponding CAPTCHA image and feeding both into the model. Under this setup, every model is evaluated in a strict zero-shot manner, and all models receive exactly the same prompt template to ensure consistency and fairness.

In our design, the reasoning portion of the prompt instructs the model to first perform step-by-step analysis before producing any coordinates or actions. This forces the model to explicitly articulate its reasoning process, identify relevant objects or regions, and only then output the final mouse-action sequence with precise coordinates. This structure maximizes the model's reasoning capability while ensuring that the final outputs adhere to a clean and standardized format.

By enforcing this template across all models, we guarantee both the correctness of the output structure and a controlled evaluation protocol, enabling reliable comparison of reasoning behaviors and performance. Importantly, the prompt is fixed across all models and was not iteratively tuned for any specific architecture. This minimizes prompt-engineering bias and ensures that the evaluation faithfully reflects each model's inherent reasoning capability rather than template-specific optimization.

Prompt (With Reasoning).
Given this image,

- First, reason step by step what the task requires and which objects or regions should be clicked, based on the visual content. If you have located one object, you must output the accurate coordinates in each of the reasoning steps, as precise as possible.
- When estimating coordinates for any clickable object, always aim for the **visual center** of the object (not its outline, border, or shadow).
- Then output a JSON formatted text that contains a sequence of mouse actions, including "move", "mouseup", and "mousedown", and their corresponding coordinates in percentage of the whole image.
- At last, click on the `verify` or `Next` button when finished. Please do not click or locate the blue hand as a button; it is not a verification button. All buttons contain visible text.

Example Output:

```
{
  "reasoning": [
    "Step 1: Identify the two digits that sum to 10.",
    "Step 2: The digits 3 and 7 are present in the image.",
    "Step 3: Locate the digit 3 at [32.35, 22.28].",
    "Step 4: Locate the digit 7 at [23.89, 61.92]."
  ],
  "mouseActions": [
    {"action": "move",      "coords": {"x": 32.35, "y": 22.28}},
    {"action": "mousedown", "coords": {"x": 32.35, "y": 22.28}},
    {"action": "mouseup",   "coords": {"x": 32.35, "y": 22.28}}
  ]
}
```

Notes:

- "move" moves cursor to the given coordinate.
- "mousedown" presses the left mouse button.
- "mouseup" releases the left mouse button.
- "x" corresponds to width, "y" corresponds to height.
- (0,0) is the top-left corner of the image.
- All coordinates are percentages of the image size.

Figure 12: Prompt template for the with-reasoning setting.

E.2 With Reasoning Example Output

In this subsection, we present a concrete example from the `hCaptcha` category. We provide the model with the raw screenshot of the CAPTCHA together with our *with-reasoning* prompt. The example shown in Figure 16 corresponds to the task **"select images that must be plugged in to work."** In this instance, the lamp is the only object that satisfies this requirement. This example illustrates how the model interprets the visual scene, identifies the correct target, and performs step-by-step reasoning before producing the final action coordinates.

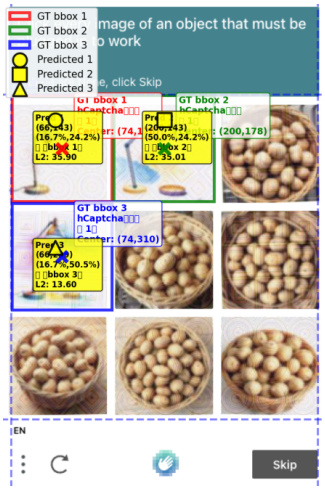


Figure 13: Example of the evaluation for an `hCaptcha` instance in our *with-reasoning* setting.

Example Reasoning Output.


```
{
  "reasoning": [
    "Step 1: The task requires clicking every image of an object that must be plugged in to work.",
    "Step 2: The images of lamps in (1,1), (1,2), and (2,1) fit the criteria.",
    "Step 3: Estimated grid as 3x3; bbox=[0, 11, 100, 90] (percent of image).",
    "Step 4: Compute step sizes: x_step=(100-0)/3=33.33, y_step=(90-11)/3=26.33.",
    "Step 5: Center(1,1): x=0+(1-0.5)*33.33=16.67, y=11+(1-0.5)*26.33=24.17.",
    "Step 6: Center(1,2): x=0+(2-0.5)*33.33=50, y=11+(1-0.5)*26.33=24.17.",
    "Step 7: Center(2,1): x=0+(1-0.5)*33.33=16.67, y=11+(2-0.5)*26.33=50.5."
  ],

  "mouseActions": [
    {"action": "move",      "coords": {"x": 16.67, "y": 24.17}},
    {"action": "mousedown", "coords": {"x": 16.67, "y": 24.17}},
    {"action": "mouseup",   "coords": {"x": 16.67, "y": 24.17}},

    {"action": "move",      "coords": {"x": 50, "y": 24.17}},
    {"action": "mousedown", "coords": {"x": 50, "y": 24.17}},
    {"action": "mouseup",   "coords": {"x": 50, "y": 24.17}},

    {"action": "move",      "coords": {"x": 16.67, "y": 50.5}},
    {"action": "mousedown", "coords": {"x": 16.67, "y": 50.5}},
    {"action": "mouseup",   "coords": {"x": 16.67, "y": 50.5}}
  ]
}
```


Figure 14: Example reasoning output.

In Figure 13, the colored crosses mark the ground-truth click centers, each located at the center of its corresponding ground-truth cell. The yellow shapes (triangle, square, and circle) denote the model's predicted click coordinates. A prediction is counted as correct if the predicted point falls anywhere inside the correct ground-truth cell, even if it does not align exactly with the annotated center. This matches the actual CAPTCHA verification behavior, where clicking anywhere within the correct region is sufficient to pass. Thus, the L2 distance between a prediction and its corresponding ground-truth is simply computed as the Euclidean gap between the predicted click location and the annotated cell center.

This example shows the actual output produced by `Qwen2.5-VL-72B` under the *with-reasoning* setting. As illustrated, the model first identifies which grid cells satisfy the task instruction and determines their corresponding indices within the inferred 3×3 layout. It then computes the center coordinates of each selected cell based on the estimated bounding box and step sizes. Finally, the model produces the required action sequence by clicking the computed centers in order.

E.3 Without Reasoning Prompt Template

In the **without reasoning** setting, the model is instructed to directly produce the final action sequence. That is, the model outputs only the required mouse actions and their coordinates, without generating intermediate reasoning or justification. This setup isolates the model's execution ability by removing all reasoning steps from the output

E.4 Without Reasoning Example Output

This example comes from the `Icon` category and is evaluated under our *without-reasoning* setting. As shown below, the model directly outputs the final click coordinates without producing any intermediate reasoning steps.

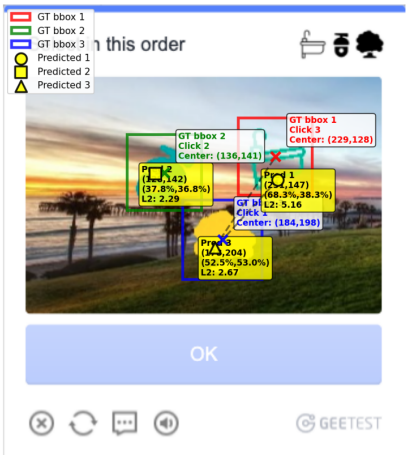


Figure 16: Example of the evaluation for an `Icon` instance in our *without-reasoning* setting.

Prompt (Without Reasoning).
Given this image,

- First, understand the task requirements and which objects or regions should be clicked, based on the visual content. If you have located one object, you must output the accurate coordinates directly, as precisely as possible.
- When estimating coordinates for any clickable object, always aim for the **visual center** of the object (not its outline, border, or shadow).
- Then output a JSON formatted text that contains a sequence of mouse actions, including "move", "mouseup", and "mousedown", and their corresponding coordinates in percentage of the whole image.
- At last, click on the `verify` or `Next` button when finished. Please do not click or locate the blue hand as a button; it is not a verification button. All buttons contain visible text.

Example Output:

```
{
  "mouseActions": [
    {"action": "move",      "coords": {"x": 32.35, "y": 22.28}},
    {"action": "mousedown", "coords": {"x": 32.35, "y": 22.28}},
    {"action": "mouseup",   "coords": {"x": 32.35, "y": 22.28}}
  ]
}
```

Notes:

- "move" moves cursor to the given coordinate.
- "mousedown" presses the left mouse button.
- "mouseup" releases the left mouse button.
- "x" corresponds to width, "y" corresponds to height.
- (0,0) is the top-left corner of the image.
- All coordinates are percentages of the image size.

Figure 15: Prompt template for the without-reasoning setting.

E.5 Data Distribution

Below is the distribution of our collected data, showing different categories and their proportions.

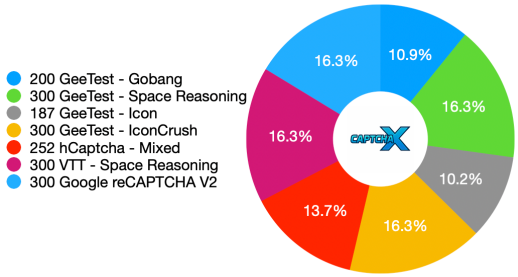


Figure 18: Distribution of our benchmark.