Closing the Spatial Execution Gap in Digital Whiteboards via Verifiable Reinforcement Learning

Chang Liu, Benjamin Wagley, Zibo Wang, Mehmet E. Belviranli, Bo Wu
Department of Computer Science
Colorado School of Mines
Golden, CO 80401, USA
{liuchang, bwagley, zibowang, belviranli, bwu}@mines.edu

Abstract

While multi-modal large language models such as GPT-5 demonstrate exceptional general understanding, they suffer from a fundamental *Spatial Execution Gap*, failing to translate visual semantics into precise, schema-valid coordinate operations in interactive environments. In this work, we show that model scale alone cannot close this gap; instead, verifiable structured reasoning provides the key to spatial precision. We present a comprehensive pipeline that leverages Group Relative Policy Optimization to enforce a strict *Identify-Reason-Verify* protocol, effectively shifting the computational burden from parameters to test-time reasoning. By utilizing a multi-agent system to distill optimal reasoning schemas and training on execution-verifiable rewards, our specialized 3B agent achieves 100% format coherence and 81.12% operation accuracy on digital whiteboard tasks. Crucially, our approach outperforms a state-of-the-art frontier model, GPT-5, by 16.75% in operation accuracy. The results suggest that for complex user interface manipulation, small, RL-aligned models with dedicated reasoning protocols are superior to generalist frontier models, offering a promising direction for building reliable web agents.

1 Introduction

Large Language Models (LLMs) and Vision-Language Models (VLMs) have revolutionized information retrieval and content generation (Sancheti et al., 2024; Wanigasekara et al., 2024). However, a critical disconnect remains between *seeing* an interface and *acting* upon it. While frontier models such as GPT-5 (OpenAI, 2025b) can semantically describe a complex diagram with high fidelity, they frequently fail when tasked with manipulating it, struggling to output precise coordinate-based operations (e.g., JSON schemas) that satisfy spatial constraints (Hutchinson et al., 2024; Fu et al., 2024; Yun et al., 2024). We term this the *Spatial Execution Gap*: the inability of generalist models to align semantic intent with the strict geometric and syntactic requirements of web interfaces. This gap persists because pre-training objectives prioritize plausible text generation over the verifiable precision required to interact with a Document Object Model (DOM).

Digital whiteboards (excalidraw.com, 2025; tldraw.com, 2025) serve as a challenging testbed for addressing this challenge. Unlike static image generation or simple button-clicking web agents, whiteboard manipulation requires an agent to `Search`, `Create`, `Update`, and `Delete` objects while maintaining complex spatial consistency (e.g., "connect the red circle to the blue square without overlapping the yellow star"). In this domain, approximation is failure; a single pixel error or syntax hallucination renders an operation invalid. Prior approaches relying on Supervised Fine-Tuning (SFT) often fail to generalize, as models tend to memorize specific layout patterns rather than learning the underlying spatial logic. Furthermore, relying solely on model scale yields diminishing returns: as our experiments show, even large frontier models often prioritize plausible text generation over verifiable geometric accuracy.

To address this, we propose a paradigm shift from *imitation* to *reasoning*. We hypothesize that reliable whiteboard manipulation requires a "System 2" approach: allocating a test-time reasoning protocol in which the agent explicitly identifies constraints, computes coordinates, describes the scene after operations, and verifies the outcome before execution. Instead of hoping a larger model "intuits" the coordinates, we force a smaller model to "measure" them through structured thought.

Herein, we present a comprehensive framework to enable this capability in efficient, small-scale models. Our approach synthesizes three key innovations: (1) a benchmark of eight rigorous spatial tasks with mathematical verifiability; (2) a multi-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13834–13849

agent optimization system that acts as a curriculum generator, autonomously discovering effective Chain-of-Thought (CoT) protocols; and (3) a Group Relative Policy Optimization (GRPO) training pipeline (Guo et al., 2025). Unlike traditional reinforcement learning from human feedback (RLHF) which relies on opaque reward models, our GRPO implementation utilizes deterministic, environment-aware rewards (e.g., Euclidean distance, JSON validity), allowing the model to explore and lock in successful reasoning paths.

Our specialized agent, built on a `Qwen2.5-VL-3B-Instruct` backbone, achieves 81.12% operation accuracy, surpassing the commercially available GPT-5 (64.37%) by a significant margin. This suggests that a small model, when trained to expend tokens on structured reasoning, can outperform a frontier model that attempts to solve the problem intuitively.

Our contributions are summarized as follows:

- We construct a suite of eight mathematically verifiable tasks designed to probe the limits of spatial reasoning on digital whiteboards, ranging from simple object creation to complex visual balancing and maze navigation.
- We introduce an integrated pipeline that combines multi-agent prompt distillation with GRPO. This allows us to train a 3B model to strictly adhere to an *Identify-Reason-Verify* protocol with 100% format compliance.
- We demonstrate that our RL-finetuned 3B agent outperforms GPT-5 by 16.75% in operation accuracy. We provide extensive analysis on the trade-off between inference latency (reasoning budget) and accuracy, proving that specialized reasoning beats generalist intuition in precision UI tasks.

2 Related Work

2.1 LLM-based Spatial Reasoning

Recent research has extensively addressed the limitations of LLMs in spatial reasoning across images and Scalable Vector Graphics (SVGs) (Wang et al., 2024; Cheng et al., 2024; Chen et al., 2024). To bridge the gap between semantic understanding and spatial grounding, prior work has focused on data curation strategies, such as providing synthetic data or additional depth information for 3D reasoning tasks (Ogezi and Shi, 2025; Cheng et al., 2024). Other studies explore inference-time strategies that encourage reasoning with visual reference objects (Liao et al., 2024) or "draw-to-reason" paradigms (Wu et al., 2025). Furthermore, prompt engineering and fine-tuning have been widely used to enhance VLMs' ability to interpret and generate SVGs (Zou et al., 2024; Rodriguez et al., 2025). However, broader surveys continue to report deficiencies in depth reasoning, spatial orientation, and spatial visualization among multimodal models (Xu et al., 2025; Li et al., 2025). Despite these advances, existing studies predominantly focus on passive analysis or static generation, failing to address the dynamic, schema-constrained manipulation required in digital whiteboard environments.

2.2 Sketch Generation and Vector Graphics

While limited work addresses interactive whiteboard manipulation, there is a rich history of using AI to generate sketch-style visual outputs (Ha and Eck, 2017; Das et al., 2020; Cao et al., 2019; Yu et al., 2017; Ribeiro et al., 2020). SketchRNN (Ha and Eck, 2017) introduces the QuickDraw dataset with an RNN-based generation framework. More recent approaches use LLMs to generate structured outputs such as SVGs (Polaczek et al., 2025; Xing et al., 2025b; Cai et al., 2023). For instance, SketchAgent (Vinker et al., 2025) enables LLMs to generate string representations convertible into SVGs, and Painter (Pourreza et al., 2023) utilizes supervised fine-tuning to autoregressively generate stroke sequences. Chen et al. (Chen et al., 2025a) apply reinforcement learning (RL) to train LLMs to generate structured strings. While these approaches successfully generate visual content (Chen et al., 2025a), their focus remains on *de novo* generation rather than the *manipulation* of existing elements or interactive user engagement on digital whiteboards. Our work targets this harder problem of modifying a persistent state within a strict UI schema.

2.3 Reinforcement Learning using GRPO

The recent success of DeepSeek-R1 has popularized GRPO (Guo et al., 2025) as a method to enforce verifiable reasoning. Prior work has adapted GRPO to multimodal domains: Vision-R1 (Huang et al., 2025) applies it to enhance image reasoning, while Video-R1 (Feng et al., 2025) achieves consistent improvements on multi-frame reasoning tasks. In the vector graphics domain, Reason-SVG (Xing et al., 2025a) combines supervised fine-tuning with GRPO and a hybrid reward design to encourage

semantically aligned SVG generation. Other approaches use GRPO to directly translate natural language into SVGs (Wang et al., 2025). Beyond specific applications, analysis of DeepSeek-R1 highlights the advantages of a critic-free, group-based policy design (Guo et al., 2025), while variants such as TGRPO extend the framework to explore vision-language-action models (Chen et al., 2025b). These developments indicate a broader movement toward environment-aware reward functions. We build upon this by designing specific format and accuracy rewards to solve the "Spatial Execution Gap" in web UI tasks.

3 Methodology

To bridge the Spatial Execution Gap, we propose a holistic framework that transforms LLMs from passive observers into active, verifiable agents. Our methodology is structured into four stages: (1) formalizing the whiteboard manipulation problem as a schema-constrained decision process; (2) establishing a mathematically verifiable benchmark and reward system; (3) distilling an optimal reasoning curriculum via multi-agent interaction; and (4) optimizing the agent's spatial policy using GRPO.

3.1 Problem Formulation

We formalize whiteboard manipulation as an observable Markov Decision Process. The environment state $\mathcal{S}$ represents the DOM of the whiteboard, containing the precise geometric properties (coordinates x, y, dimensions w, h, style) of all objects. The agent receives a dual-modality observation $\mathcal{O}$: a rasterized screenshot $V \in \mathbb{R}^{H \times W \times 3}$ capturing spatial layouts, and a serialized JSON object list L_{json} capturing semantic attributes.

The action space $\mathcal{A}$ is non-trivial. Unlike standard text generation, operations must adhere to the digital whiteboard's strict API requirements. We define S_{wb} as the whiteboard schema, representing the set of all syntactically valid JSON structures accepted by the execution engine (specifying required keys such as `id`, `type`, and `props`). A generated action a_t is considered valid if and only if it conforms to this schema and executes without runtime errors on the whiteboard instance:

$$\begin{aligned} \text{Valid}(a_t) \iff & a_t \in S_{wb} \\ & \wedge \text{Exec}(a_t, \mathcal{S}_t) \neq \text{Error} \end{aligned} \tag{1}$$

Our objective is to learn a policy $\pi_\theta(a_t|o_t)$ that maximizes cumulative reward. However, direct mapping from $o_t \to a_t$ is prone to "spatial hallucination." Therefore, we enforce a latent reasoning step z_t (Chain-of-Thought), decomposing the generation into $z_t \sim \pi_\theta(\cdot|o_t)$ followed by $a_t \sim \pi_\theta(\cdot|o_t, z_t)$. The reasoning trace z_t must explicitly calculate spatial constraints before the action a_t is committed.

3.2 Benchmark and Verifiable Reward Engineering

To rigorously evaluate and train this capability, we introduce the Whiteboard-8 Benchmark (Figure 1). Unlike semantic benchmarks, every task in our benchmarks is paired with a deterministic operation accuracy reward function $R_{acc} \in [0, 1]$ and a format reward function R_{format}. This transforms the benchmark from a static test set into a dynamic reinforcement learning environment.

3.2.1 Task-Specific Accuracy Rewards

We target four core manipulation primitives: `Search`, `Create`, `Update`, and `Delete`. The tasks and their verifiable reward formulations are:

Constructive Tasks: These tasks require creating new objects at precise coordinates.

(1) Line Connection: The agent must draw a line connecting two existing objects. Correctness is measured by the Euclidean distance between the line's endpoints and the target object centroids.

(2) Labeling: The agent must add a text annotation to a target object. The reward verifies that the text object is correctly positioned within the target's bounding box.

(3) Visual Balance: The agent must place an object to shift the canvas's Center of Mass (CoM) toward the geometric center. The reward is proportional to the reduction in CoM offset.

(4) Maze Navigation: A strict constraint satisfaction task. The agent must create a new object at a specified position inside the maze. The reward is binary: 1 if the new object is within the target cell AND has zero Intersection-over-Union (IoU) with any maze walls, else 0.

Analytical Tasks: These tasks require reasoning about relationships and attributes.

(5) Pattern Sorting: The agent must identify and `Delete` an outlier object that differs in color or structural pattern from the others. Reward is 1 if the correct unique ID is removed.

(6) Overlap Detection: The agent must identify and remove objects positioned behind or above a

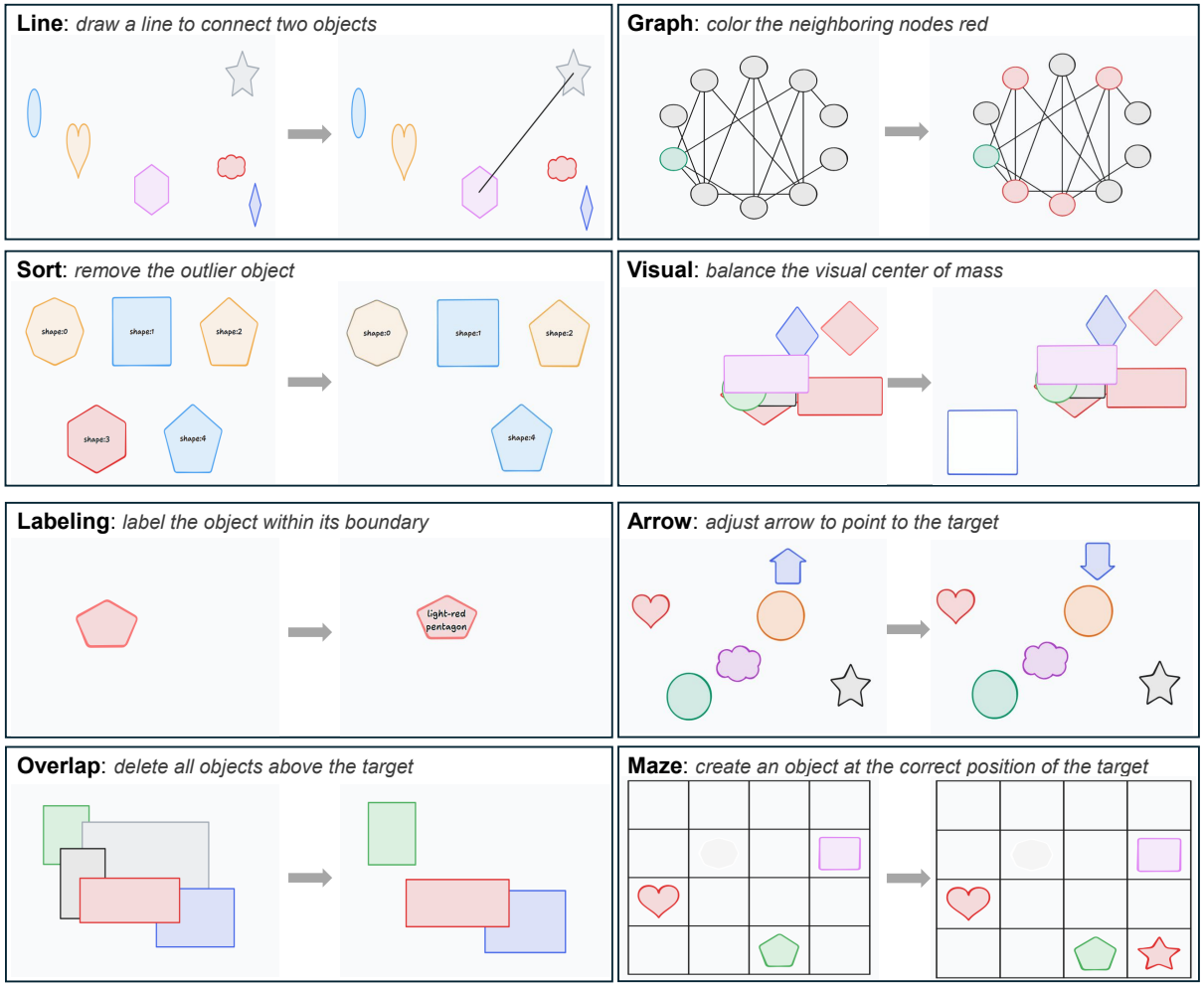

Figure 1: The Whiteboard-8 Benchmark. We define eight verifiable tasks ranging from simple object creation (Line, Maze) to complex relationship reasoning (Graph, Sort). Each task is paired with a deterministic reward function.

target object, requiring understanding of z-order and spatial layering. Reward is based on correctly identifying all overlapping elements.

(7) Graph Coloring: The agent must identify neighbors of a target node in a graph structure (e.g., flowchart). The reward is defined as the average of the precision and recall of the selected set with respect to the ground-truth adjacency matrix.

Geometric Tasks: **(8) Arrow Alignment:** The agent must understand an arrow's current direction and rotate it to point at a specified target object. The reward penalizes the angular error: $R_{arrow} = \max(0, \cos(\theta_{pred} - \theta_{target}))$.

3.2.2 Format and Reasoning Reward

To enforce the "System 2" reasoning protocol, we define a structured format reward R_{format} and an operation accuracy reward R_{acc}. The format reward consists of a task-aware structured reasoning reward and a web-alignment reward.

$$R_{format} = \rho R_{\text{task-aware}} + (1 - \rho) R_{\text{web}} \quad (2)$$

$R_{\text{task-aware}}$ penalizes the agent when the response does not follow the *Identify-Reason-Verify* protocol. We further enforce a minimum reasoning depth τ_{char}, a character threshold that ensures the model expends sufficient computation characters on spatial reasoning before producing the final answer.

$$R_{\text{task-aware}} = \alpha \cdot \mathbb{I}[\text{think-answer tags}(T, A)] + \beta \cdot [\text{structured reasoning}(T)] + \gamma \cdot \mathbb{I}[len(T) \geq \tau_{char}] \quad (3)$$

where α, β, and γ are weights assigned to each criterion, T refers to the reasoning content, A denotes the final answer, and $\mathbb{I}[\cdot]$ denotes the indicator function, which evaluates to 1 if the condition is true and 0 otherwise.

Furthermore, R_{web} enforces that the final answer conforms to the correct JSON schema that can be understood by the digital whiteboard APIs.

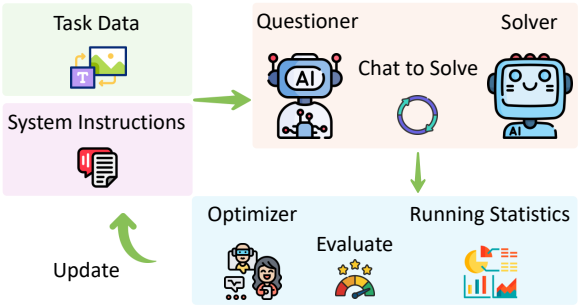

Figure 2: Multi-Agent Curriculum Discovery. The Optimizer agent analyzes statistics from the Questioner-Solver interaction to iteratively refine the system instructions, distilling the *Identify-Reason-Verify* protocol.

The total reward is defined as a weighted sum: $R_{total} = \lambda R_{format} + (1 - \lambda) R_{acc}$.

3.3 Curriculum Discovery via Multi-Agent Interaction

Before engaging in large-scale training, we must determine the optimal reasoning protocol for the agent. Bare LLMs often struggle to adhere to complex JSON schemas or fail to perform sufficient spatial reasoning. To solve this, we design a multi-agent system that autonomously evaluates and refines the agent's prompts and reasoning strategies (Figure 2). This system serves as a curriculum generator, producing the high-quality seed data and system instructions used for GRPO training.

The system consists of three agents: a Questioner, a Solver, and an Optimizer. The Questioner utilizes the task descriptions to prompt the Solver. It maintains the conversation history and provides feedback based on the Solver's performance. The Solver (the target LLM) attempts to solve the task by generating text-based manipulations. Critically, to explore the solution space, the Solver generates $K = 3$ candidate responses for each query. The Optimizer serves as the critic. It reviews the candidates and the summarized runtime statistics (format errors, accuracy scores) to iteratively refine the system instructions. The interaction proceeds in rounds. In each round, the Solver generates candidates which are rigorously evaluated using the reward functions defined in Sec. 3.2. The candidate with the highest combined format and accuracy score is selected as the "winning" response. The Questioner then provides specific feedback to the Solver based on the *missing or erroneous components* of the selected candidates (e.g., "You failed to include the 'Verify' section"). This feedback is added to the chat history, guiding the Solver in the next turn. The Optimizer analyzes the aggregated failure modes across the dataset and updates the global system instruction (e.g., enforcing a stricter `<think>` verification step). By default, this optimization is performed for two global iterations. The result is a robust, validated system instruction for reinforcement learning training.

3.4 Optimization

The final component of our pipeline (Figure 3) is to internalize the distilled reasoning protocol into the model's weights using reinforcement learning. We employ GRPO (Guo et al., 2025), which is suited for tasks with verifiable rewards as it eliminates the need for a separate value network critic, reducing memory overhead and training instability.

We initialize the policy π_θ with the same model as the Solver in the multi-agent system. For each training step, we sample a batch of tasks $Q = \{q_1, \ldots, q_B\}$. For each task q, the policy generates a group of G outputs $\{o_1, \ldots, o_G\}$ by sampling from the distribution $\pi_{\theta_{old}}(\cdot|q)$. Each output o_i consists of the full chain-of-thought and the final JSON action. We evaluate each output against the environment to obtain a reward scalar r_i.

Instead of using a learned value function baseline, GRPO uses the group average as the baseline to compute the advantage $\hat{A}_i$:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\}) + \epsilon} \quad (4)$$

This group-relative advantage encourages the model to reinforce trajectories that are better than the average of its own current attempts. The objective function maximizes this advantage while constraining the policy update via KL divergence. By training directly on the deterministic R_{total} (combining format adherence and operation accuracy), GRPO effectively searches the reasoning space for strategies that consistently yield valid execution, thereby closing the spatial execution gap.

4 Experimental Settings

To evaluate the effectiveness of our pipeline in closing the spatial execution gap, we conduct a rigorous comparative analysis against both open-source and frontier commercial models.

4.1 Environment and Dataset

Our experiments are conducted on a locally deployed instance of `TLDraw` (tldraw.com, 2025), a

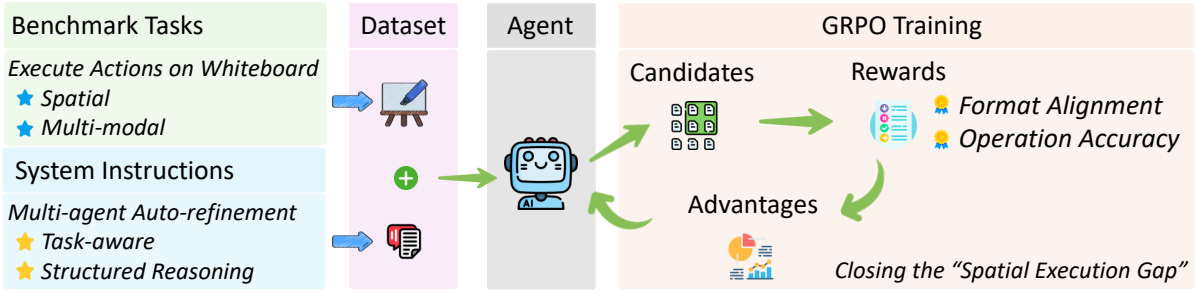

Figure 3: The GRPO Training Pipeline. We initialize the policy with the multi-agent curriculum and optimize it using verifiable geometric rewards. The critic-free GRPO update reinforces successful reasoning traces.

popular open-source digital whiteboard. This setup ensures deterministic rendering and allows for precise geometric verification of object states. We developed an auto-generator that programmatically creates diverse whiteboard layouts corresponding to our eight benchmark tasks. To prevent the model from memorizing specific layouts, we employ procedural scene generation with extensive randomization. Each scene is constructed by sampling object types from 11 geometric primitives, colors from the full TLDraw palette, and positions via uniform random placement within the canvas. Train and test sets are generated with different random seeds, ensuring no layout overlap. Our reward functions evaluate *geometric correctness* (e.g., Euclidean distance, IoU) rather than pattern matching, so the model must learn generalizable spatial reasoning rather than memorizing coordinate templates.

We have generated two datasets: a small set of 600 samples used by the multi-agent system for prompt distillation, and a larger corpus of 6,000 samples for GRPO training, with an additional 600 samples each for validation and testing. All data is stored in a standardized JSON format compatible with the HuggingFace dataset structure, enabling seamless integration across training frameworks.

4.2 Model Configurations

We compare our approach against a range of models to study the impact of scale versus reasoning:

- **Our Agent:** Based on *Qwen2.5-VL-3B-Instruct* (Bai et al., 2025b). We select this lightweight backbone to show that efficient models can achieve high performance suitable for private, local deployment.
- **Open-Source Baselines:** We evaluate the `Qwen2.5-VL-Instruct` family (3B, 7B, 72B) to measure the effect of parameter scaling in a standard instruction-tuned setting.
- **Frontier Baselines:** We compare against `GPT-4o` (Hurst et al., 2024), `Gemini-2.5-Pro` (Google DeepMind, 2025), and the state-of-the-art model `GPT-5` (OpenAI, 2025b).

To ensure a rigorous evaluation, we do not simply prompt the baselines with zero-shot instructions. Instead, all baseline models (including GPT-5) are prompted with the *final, optimized system instruction* discovered by our multi-agent system, including the same few-shot examples. This ensures that any performance gap is due to the model's intrinsic spatial reasoning capability and our GRPO training, rather than unfair prompt engineering.

4.3 Implementation Details

The GRPO training pipeline is built on `EasyR1` (Zheng et al., 2025). We train for 500 steps with a learning rate of 5×10^{-6}, a group size of $G = 5$, a KL coefficient of $\beta = 0.01$, and a data type of `bfloat16`. All experiments were performed on a cluster of four NVIDIA A100 (40 GB) GPUs.

5 Results and Analysis

We present empirical evidence demonstrating that our RL-aligned 3B agent outperforms frontier models by shifting the computational burden to verifiable test-time reasoning.

5.1 Main Results: Closing the Execution Gap

Table 1 summarizes the performance of our pipeline against open-source and commercial baselines. The results reveal a stark contrast between generalist capabilities and precise spatial execution.

Our `Qwen2.5-VL-3B-Instruct` agent, which is trained using GRPO with structured spatial reasoning, achieves an operation accuracy of 81.12%. This result surpasses `Gemini-2.5-Pro` (60.78%) by 20.34% and the state-of-the-art

Model	Format (%)	Accuracy (%)
Qwen-3B	30.60	1.59
Qwen-7B	87.69	24.74
Qwen-72B	94.32	30.78
Gemini-2.5-Pro	97.70	60.78
GPT-4o	99.04	42.90
GPT-5	98.71	64.37
Ours (3B)	**100.0**	**81.12**

Table 1: Comparative Results. Our 3B model, trained with GRPO, significantly outperforms the larger-scale GPT-5 on spatial operation accuracy. Qwen-3B/7B/72B belong to the Qwen2.5-VL-Instruct family.

model `GPT-5` (64.37%) by 16.75%. This result challenges the prevailing assumption that model scale is the primary driver of performance. While `Qwen2.5-VL-72B-Instruct` improves over smaller base models (30.78% vs 1.59%), it still falls significantly short of the RL-tuned agent. This confirms our hypothesis: the spatial execution gap cannot be closed by parameter scaling alone; it requires environment-aware feedback optimization.

Remarkably, our pipeline achieves 100% format coherence, generating valid JSON for every test case. While GPT-4o and GPT-5 also show high adherence (~99%), their failures in operation accuracy indicate that they often generate *syntactically valid* but *semantically incorrect* JSON (e.g., valid rectangles placed at wrong coordinates).

5.2 Qualitative Analysis: The Hallucination Problem

To understand why frontier models underperform despite their massive knowledge base, we analyzed failure cases in the *Overlap* and *Maze* tasks.

We observe a phenomenon of *Spatial Hallucination*. GPT-5 typically identifies the correct intent (e.g., "I need to draw a box around the red circle") but fails to ground this intent in the pixel coordinate space. For example, in the *Maze* task, GPT-5 often generates coordinates that seemingly "teleport" through walls, violating collision constraints. It prioritizes plausible-looking coordinates over calculated ones. In contrast, our GRPO-trained agent exhibits measured execution. By enforcing the `<think>` protocol, the agent explicitly calculates boundaries in its reasoning trace (e.g., *"The wall ends at x=300, so I must place the object at x>305"*) before committing to the JSON output. This "measure-then-act" behavior is the direct result of the reinforcement learning signal penalizing geometric violations during GRPO training.

Stage	Format (%)	Accuracy (%)
Base	30.60	1.59
+ Questioner-Solver	99.40	23.93
+ Optimizer (2 rounds)	99.55	25.16
+ GRPO Training	**100.0**	**81.12**

Table 2: Pipeline Progression. Each stage contributes cumulatively: the multi-agent system drives format compliance ($30.60\% \rightarrow 99.55\%$) and provides a stable initialization, while GRPO drives the major accuracy leap ($25.16\% \rightarrow 81.12\%$).

5.3 The Necessity of the Multi-Agent Curriculum

Is reinforcement learning alone sufficient, or is the initialization critical? We evaluate the contribution of our multi-agent system, which serves as a curriculum generator to "warm start" the policy. Table 2 quantifies each stage of the pipeline.

Using only the Questioner-Solver setup (without the Optimizer's prompt refinement), the base model achieves a 99.40% format score but a low 23.93% operation accuracy. This indicates that while the model can follow the JSON schema, it lacks the reasoning depth to solve the spatial puzzle. After introducing the Optimizer and performing two rounds of prompt refinement to distill the *Identify-Reason-Verify* protocol, the operation accuracy improves to 25.16% before any RL training occurs. While the Optimizer's direct accuracy gain is modest (+1.23%), it critically stabilizes the reasoning structure, yielding a policy that consistently produces well-formed reasoning traces suitable for GRPO optimization.

This creates a stable starting policy for GRPO. Without this aligned initialization, RL training is often unstable because the model rarely stumbles upon a correct solution by chance (sparse rewards), leading to mode collapse. The multi-agent system bridges this exploration gap, and GRPO then amplifies spatial precision from 25.16% to 81.12%.

5.4 Ablation: Structured Reasoning vs. Unconstrained Generation

To quantify the value of the "System 2" reasoning protocol, we conduct a component-wise ablation by independently toggling the structured protocol and the minimum character threshold during GRPO training. Results are shown in Table 3.

Two findings emerge. First, the structured protocol alone provides a modest gain (+1.56%), but

Structured Protocol	Threshold	Accuracy (%)
No	No	76.10
Yes	No	77.66
No	Yes	68.83
Yes	Yes	**81.12**

Table 3: Component-wise Reasoning Ablation. Both the structured *Identify-Reason-Verify* protocol and the reasoning budget threshold are necessary; either alone is insufficient, and the threshold without structure actively degrades performance.

combining it with the reasoning budget yields a substantial +5.02% improvement over the unconstrained baseline. Second, naively enforcing a character threshold without structure actually hurts performance ($76.10\% \rightarrow 68.83\%$): without the *Identify-Reason-Verify* scaffold, the model fills tokens with unhelpful or repetitive content rather than meaningful spatial calculations. This confirms that our structured protocol is not merely a formatting constraint; it is a substantive contributor to accuracy by ensuring that reasoning tokens are spent on geometric calculations, coordinate verification, and scene imagination. The structure forces the model to "think before acting", preventing premature commitment to incorrect coordinates.

5.5 Ablation: The Role of Visual Input

Our prompts include both a rasterized screenshot and a serialized JSON representation of whiteboard objects. This raises a natural question: does the visual modality provide meaningful signal, or does the model rely primarily on the textual object list?

To investigate, we trained variants using text-only LLMs (Yang et al., 2024, 2025b,a) on identical data but excluding image inputs. As shown in Table 4, the non-visual `Qwen2.5-3B-Instruct` achieves 78.60% accuracy, which is competitive but noticeably lower than our full pipeline (81.12%). This 2.52% gap suggests that while textual object descriptions carry substantial information, the visual modality provides complementary spatial cues that improve precision.

We also evaluated the impact of few-shot examples. Removing all examples from the prompt (zero-shot) reduces accuracy to 77.39%, confirming that demonstrations help the model ground its spatial reasoning and improve operation accuracy.

Model	Accuracy (%)
Text-only (no image input)	
Qwen2.5-3B-Instruct	78.60
Qwen2.5-0.5B	0.05
Qwen2.5-1.5B	55.18
Qwen3-0.6B	64.62
Qwen3-1.7B	63.23
Qwen3-4B	74.07
Multimodal (image + text)	
Ours (3B + Vision)	**81.12**

Table 4: Impact of Visual Input. Text-only models underperform the full multimodal pipeline, indicating that visual input provides useful spatial grounding.

Model	Format (%)	Accuracy (%)
Qwen2.5-3B (Base)	30.60	1.59
Qwen2.5-3B (+GRPO)	**100.0**	81.12
Qwen3-2B (Base)	65.07	13.57
Qwen3-2B (+GRPO)	99.16	79.11
Qwen3-4B (Base)	90.50	29.47
Qwen3-4B (+GRPO)	**100.0**	**85.40**
GPT-5	98.71	64.37

Table 5: Cross-Architecture Generalization. Our GRPO pipeline yields large gains across two architecture generations (Qwen2.5-VL-Instruct and Qwen3-VL-Instruct) and three parameter scales.

5.6 Generalization Across Model Architectures

A natural concern is whether our gains are specific to the `Qwen2.5-VL-3B-Instruct` backbone. To test generalization, we apply our full GRPO pipeline (identical rewards, curriculum, and structured reasoning protocol) to models from a different architecture generation. As shown in Table 5, the pipeline consistently transforms base models across both the Qwen2.5-VL-Instruct and Qwen3-VL-Instruct (Bai et al., 2025a) families.

The accuracy improvements are consistent and large: +79.53% on Qwen2.5-3B, +65.54% on Qwen3-2B, and +55.93% on Qwen3-4B. Notably, the `Qwen3-VL-4B-Instruct` model achieves 100% format compliance and 85.40% accuracy, surpassing our original 3B agent and outperforming GPT-5 by 21.03%. These results confirm that the performance gains stem from GRPO training and structured reasoning, not from architecture-specific properties of a single backbone. The consistent cross-architecture transfer at the 2B-4B scale provides strong evidence of generalizability.

Model	Inference Time (s)
Qwen2.5-VL-3B-Instruct (Base)	4.87
Qwen2.5-VL-72B-Instruct	44.20
Ours (3B + Reasoning)	**18.15**

Table 6: Inference Time Comparison. Our pipeline trades speed for accuracy. Despite a 3.7× slowdown over the base 3B model, it is 2.4× faster and 2.6× more accurate than the 72B model.

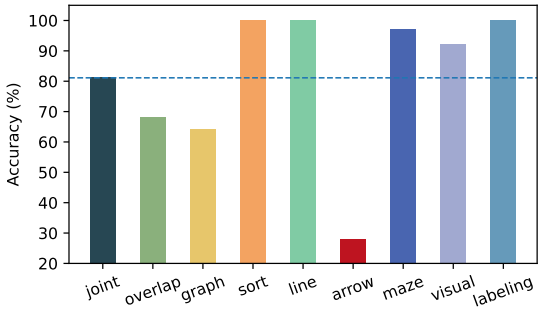

Figure 4: Task-Specific Training Dynamics. While most tasks perform well, the *Arrow Alignment* task (red bar) suffers from negative transfer in joint training, highlighting the challenge of multi-objective reward balancing.

5.7 The Cost of Precision: Inference Latency

A trade-off inherent to our approach is the increased inference time required for reasoning. In Table 6, our pipeline requires an average of 18.15 seconds per task, compared to 4.87 seconds for the base `Qwen2.5-VL-3B-Instruct` model.

While this shows a ∼4× increase in latency, we argue it is a necessary "Cognitive Budget" for reliable agentic behavior. In high-stakes UI manipulation, speed is secondary to validity; a fast but incorrect operation (like deleting the wrong object) incurs a high cost of user correction. Moreover, when viewed as an accuracy-latency Pareto frontier, our pipeline strictly dominates larger models: `Qwen2.5-VL-72B-Instruct` requires 44.20 s yet achieves only 30.78% accuracy, whereas our 3B agent reaches 81.12% in 18.15 s, 2.4× faster and 2.6× more accurate.

A natural question is whether frontier models can close the gap simply by allocating more reasoning tokens. To test this, we prompted `GPT-5` with an extended output budget of 12,288 tokens (up from 4,096), allowing for more extensive chain-of-thought and explicit verification steps. The results show a format score of 98.54% and an accuracy of 67.98%, representing only a marginal improvement of +3.61% over the standard configuration (64.37%), while our 3B agent still leads by 13.14%. This further supports our central argument: the spatial execution gap cannot be closed through increased reasoning budget alone. The reasoning protocol must be internalized through environment-aware optimization (i.e., GRPO training), rather than merely described in the prompt. Frontier models can follow spatial reasoning protocols in form, but without verifiable feedback during training, they do not reliably produce geometrically correct outputs.

5.8 Task Dynamics and Negative Transfer

We analyze the dynamics of multi-task learning. While GRPO generally improves performance across the benchmark, we observe negative transfer in specific tasks. In Figure 4, joint training yields near-perfect accuracy for 5 out of 8 tasks but degrades performance on the *Arrow Alignment* test (dropping to ~30% vs. 48% in isolated training).

We hypothesize this is due to reward dominance. The geometric rewards for tasks like *Line Connection* and *Maze Navigation* are dense and easier to optimize via coordinate shifts, whereas the angular reward for *Arrow Alignment* is more sensitive to small perturbations. The policy likely collapses to a local optimum that prioritizes the easier, high-reward tasks. This suggests that future work should incorporate dynamic reward scaling or curriculum learning to balance task difficulties.

6 Conclusions

We demonstrate that the spatial execution gap in Multimodal LLMs cannot be closed by scale alone, but by verifiable structured reasoning. By combining a multi-agent curriculum with GRPO, we train a specialized 3B agent that outperforms the frontier model `GPT-5` by 16.75% in operation accuracy while maintaining 100% format schema compliance. These results establish that for high-precision user interface tasks, efficient "System 2" reasoning models offer a superior and privacy-preserving alternative to generalist foundation models. Future work will address the inference latency trade-off through knowledge distillation and explore curriculum learning to mitigate negative transfer in multi-objective optimization.

Limitations

Our study focuses on `TLDraw` as a representative testbed for vector-based web interfaces. While the

fundamental challenge of mapping semantic intent to coordinate-based JSON schemas is universal across platforms like Figma, Miro, or Excalidraw, specific schema intricacies (e.g., coordinate systems, nesting depth) vary. Importantly, our reward design is platform-agnostic at the reasoning level: the accuracy rewards (R_{acc}) operate on abstract spatial properties (Euclidean distances, IoU, angular errors) that are independent of any platform's JSON schema. Only the web-alignment reward (R_{web}) is platform-specific, as it validates JSON key structures. As detailed in Appendix A.7, the core spatial primitives between TLDraw and Excalidraw are semantically identical, differing only in key naming conventions, nesting depth, and minor value scaling. Adapting to a new platform therefore requires updating only the R_{web} schema validation and output template, while all spatial reasoning and accuracy rewards transfer unchanged. Nevertheless, our results require further validation on broader cross-platform benchmarks to ensure true ecological validity.

As observed in the *Arrow Alignment* task, joint training can lead to performance degradation in specific sub-tasks due to reward dominance. Dense, high-magnitude rewards (e.g., from Maze navigation) can overshadow sparser signals (e.g., arrow angular alignment), causing the policy to converge to local optima that favor the dominant tasks. While we show that isolated training solves this, building a unified agent requires more sophisticated multi-task optimization strategies, such as dynamic loss weighting or curriculum learning, which are outside the scope of this work.

The "System 2" reasoning protocol comes at a cost: our pipeline requires 18.15 seconds per operation, a $\sim 4\times$ slowdown compared to the base model. While acceptable for asynchronous tasks (e.g., "cleanup this board"), this latency is prohibitive for real-time interactive auto-complete. We view our current model as a *teacher*: future work should explore knowledge distillation, compressing these high-quality reasoning traces into a faster "System 1" model that retains spatial precision without the runtime overhead.

Reinforcement learning on reasoning trajectories introduces specific instabilities. We observe instances of *Format Overfitting*, where the model optimizes for the reasoning structure within the `<think>` tags (maximizing R_{format}) without necessarily improving the quality of the geometric logic. Furthermore, performance is sensitive to the reasoning character threshold (τ_{char}): gains often vanish outside a narrow hyperparameter band, highlighting the need for more robust, adaptive constrained RL algorithms.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2439815.

References

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Mu Cai, Zeyi Huang, Yuheng Li, Haohan Wang, and Yong Jae Lee. 2023. Delving into llms' visual understanding ability using svg to bridge image and text.

Nan Cao, Xin Yan, Yang Shi, and Chaoran Chen. 2019. Ai-sketcher: a deep generative model for producing high-quality sketches. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 2564–2571.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. 2024. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465.

Yamei Chen, Haoquan Zhang, Yangyi Huang, Zeju Qiu, Kaipeng Zhang, Yandong Wen, and Weiyang Liu. 2025a. Symbolic graphics programming with large language models. *arXiv preprint arXiv:2509.05208*.

Zengjue Chen, Runliang Niu, He Kong, and Qi Wang. 2025b. Tgrpo: Fine-tuning vision-language-action model via trajectory-wise group relative policy optimization. *arXiv preprint arXiv:2506.08440*.

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. 2024. Spatialrgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093.

Ayan Das, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Yi-Zhe Song. 2020. Béziersketch: A generative model for scalable vector sketches. In *European conference on computer vision*, pages 632–647. Springer.

excalidraw.com. 2025. Excalidraw. https://github.com/excalidraw/excalidraw. Accessed: 2025-10-6.

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. 2025. Video-r1: Reinforcing video reasoning in mllms. *arXiv preprint arXiv:2503.21776*.

Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. 2024. Scene-llm: Extending language model for 3d visual understanding and reasoning. *arXiv preprint arXiv:2403.11401*.

Google DeepMind. 2025. Gemini-2.5-pro. https://ai.google.dev/gemini-api/docs/models. Accessed: 2025-12-20.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

David Ha and Douglas Eck. 2017. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. 2025. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, and 399 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Maeve Hutchinson, Radu Jianu, Aidan Slingsby, and Pranava Madhyastha. 2024. Llm-assisted visual analytics: Opportunities and challenges. *arXiv preprint arXiv:2409.02691*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Xiao Liang, Zhijiang Guo, and 2 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *Preprint*, arXiv:2502.17419.

Yuan-Hong Liao, Rafid Mahmood, Sanja Fidler, and David Acuna. 2024. Reasoning paths with reference objects elicit quantitative spatial reasoning in large vision-language models. *arXiv preprint arXiv:2409.09788*.

Chang Liu, Loc Hoang, Andrew Stolman, Rene F Kizilcec, and Bo Wu. 2025. Understanding student engagement with large language model-powered course assistants. In *International Conference on Artificial Intelligence in Education*, pages 3–10. Springer.

Chang Liu, Loc Hoang, Andrew Stolman, and Bo Wu. 2024. Hita: A rag-based educational platform that centers educators in the instructional loop. In *International conference on artificial intelligence in education*, pages 405–412. Springer.

Michael Ogezi and Freda Shi. 2025. Spare: Enhancing spatial reasoning in vision-language models with synthetic data. *arXiv preprint arXiv:2504.20648*.

OpenAI. 2025a. Chatgpt. https://chatgpt.com/. Accessed: 2025-12-20.

OpenAI. 2025b. Gpt-5. https://openai.com/index/introducing-gpt-5/. Accessed: 2025-10-6.

Sagi Polaczek, Yuval Alaluf, Elad Richardson, Yael Vinker, and Daniel Cohen-Or. 2025. Neuralsvg: An implicit representation for text-to-vector generation. *arXiv preprint arXiv:2501.03992*.

Reza Pourreza, Apratim Bhattacharyya, Sunny Panchal, Mingu Lee, Pulkit Madan, and Roland Memisevic. 2023. Painter: Teaching auto-regressive language models to draw sketches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 305–314.

Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. 2020. Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14153–14162.

Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. 2025. Starvector: Generating scalable vector graphics code from images and text. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16175–16186.

Prateek Sancheti, Kamalakar Karlapalem, and Kavita Vemuri. 2024. Llm driven web profile extraction for identical names. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1616–1625.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

tldraw.com. 2025. Tldraw. https://github.com/tldraw. Accessed: 2025-10-6.

Yael Vinker, Tamar Rott Shaham, Kristine Zheng, Alex Zhao, Judith E Fan, and Antonio Torralba. 2025. Sketchagent: Language-driven sequential sketch generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23355–23368.

Feiyu Wang, Zhiyuan Zhao, Yuandong Liu, Da Zhang, Junyu Gao, Hao Sun, and Xuelong Li. 2025. Svgen: Interpretable vector graphics generation with large language models. *arXiv preprint arXiv:2508.09168*.

Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Sharon Li, and Neel Joshi. 2024. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. *Advances in Neural Information Processing Systems*, 37:75392–75421.

Sashini Wanigasekara, Dinesh Asanka, Chathura Rajapakse, Dilani Wickramaarachchi, and Abhiru Wijesinghe. 2024. Comparing the adaptability of a genetic algorithm and an llm-based framework for automated software test data generation: In the context of web applications. In *2024 IEEE 3rd International Conference on Data, Decision and Systems (ICDDS)*, pages 1–6. IEEE.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. 2025. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. *arXiv preprint arXiv:2506.09965*.

Ximing Xing, Yandong Guan, Jing Zhang, Dong Xu, and Qian Yu. 2025a. Reason-svg: Hybrid reward rl for aha-moments in vector graphics generation. *arXiv preprint arXiv:2505.24499*.

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. 2025b. Empowering llms to understand and generate complex vector graphics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19487–19497.

Wenrui Xu, Dalin Lyu, Weihang Wang, Jie Feng, Chen Gao, and Yong Li. 2025. Defining and evaluating visual language models' basic spatial abilities: A perspective from psychometrics. *arXiv preprint arXiv:2502.11859*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. 2017. Sketch-a-net: A deep neural network that beats humans. *International journal of computer vision*, 122(3):411–425.

Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Qazim Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, Timothy Baldwin, Zhengzhong Liu, Eric P. Xing, Xiaodan Liang, and Zhiqiang Shen. 2024. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. *Advances in neural information processing systems*, 37:112134–112157.

Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, and Yuwen Xiong. 2025. Easyr1: An efficient, scalable, multi-modality rl training framework.

Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. 2024. Vgbench: Evaluating large language models on vector graphics understanding and generation. *arXiv preprint arXiv:2407.10972*.

A Appendix

A.1 Prompt Design

A typical prompt in this study includes the whiteboard screenshot image, text-based descriptions of the whiteboard objects, the task description, specific instructions to avoid misunderstanding, a template of the standard whiteboard object format, several few-shot examples, and the system instructions. An example looks as follows:

```
[Image Data]
[Whiteboard Objects: "x":0, "y":0, "rotation":0, ...]
[Task: Draw a red star to the north of the red octagon]
[Instructions: Do not deviate from the schema. Validate your output against
```

the schema before returning it. Ensure that your output is enclosed in a JSON code block in markdown format. Do not output additional JSON blocks, and do not include comments or mathematical expressions in the JSON block. All values must be fully computed before submission.]

[Template: Your output should be formatted as a JSON block in tldraw using the following schema...]

[Examples: For instance, for the following tasks...]

[System Instructions: The user and the assistant are having a conversation. The user asks the assistant to solve a task. The assistant provides the solution, including their reasoning process and final answer. The reasoning must be enclosed in <think>...</think> and must follow this exact structure and order: Identify: State which test is being solved and output the test name immediately after "Identify:" using one of the following: shape, maze, visual, line, graph, overlap, sort, point. Reasoning: Use step-by-step reasoning with labels "Step 1", "Step 2", "Step 3", etc. Combine text reasoning and visual reasoning as needed. Intermediate Answer: Produce a concise intermediate answer that describes the action or result that the reasoning leads to. Imagination: Describe the new image after conducting the operations described in the Intermediate Answer. This description should reflect what the modified or final state of the image should look like if the Intermediate Answer is correct. Verify: Verify whether the imagined final image correctly solves the question. If it does not, reconsider and revise the answer. Close the reasoning with </think>. The final answer must be enclosed in <answer>...</answer>, containing only the final action or result...]

A.2 Operation Descriptions

There are multiple types of operations for whiteboard manipulation tasks. An example operation that describes how to create a rectangle on the whiteboard is shown as follows:

```
{"createShapes": [{"id": "shape:new_shape
", "type": "geo", "x": 450, "y": 200, "prop
s": {"geo": "rectangle", "w": 300, "h": 20
0, "color": "blue", "fill": "semi"}}]}
```

A.3 Reward Function Formulations

The overall reward is defined as:

$$R_{\text{total}} = \lambda\, R_{\text{format}} + (1-\lambda)\, R_{\text{acc}}, \quad 0 \le R_{\text{total}} \le 1 \tag{5}$$

where λ is a weighting parameter that controls the trade-off between format correctness and operation accuracy. By default, $\lambda = 0.5$ during GRPO training.

The format reward is defined as a balanced combination of the task-aware structured reasoning reward $R_{\text{task-aware}}$ and the web-alignment reward R_{web}:

$$R_{format} = \rho R_{\text{task-aware}} + (1-\rho) R_{\text{web}}, \tag{6}$$

where ρ is 0.5 by default.

For the task-aware structured reasoning reward, the formulation is:

$$R_{\text{task-aware}} = \alpha \cdot \mathbb{I}[\text{think-answer tags}(T, A)] + \beta \cdot [\text{structured reasoning}(T)] + \gamma \cdot \mathbb{I}[len(T) \ge \tau_{char}] \tag{7}$$

where α, β, and γ are weights assigned to each criterion, and $\mathbb{I}[\cdot]$ denotes the indicator function. By default, $\alpha = 0.1$, $\beta = 0.7$, and $\gamma = 0.2$.

The *think-answer tags* score is defined as an indicator function that checks whether the response contains both <think> and <answer> tags. The score is 1 if the response has the form <think>T</think><answer>A</answer>, and 0 otherwise. Here, T denotes the reasoning content and A denotes the final answer.

To ensure fairness, the task-aware structured reasoning reward $R_{\text{task-aware}}$ is excluded for all baseline models (except our pipeline), as they are not trained to employ it.

The web-alignment reward R_{web} checks the JSON content within the <answer> tags. Let the JSON content contain m key elements, and let $\mathbb{I}[e_i]$ be the indicator function that evaluates to 1 if the i-th element is correctly generated and 0 otherwise. The web-alignment reward is defined as:

$$R_{\text{web}} = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}[e_i]. \tag{8}$$

This ensures that the reward is scaled between 0 and 1. The web-alignment reward enforces JSON

correctness, aligns outputs with digital whiteboard operations, and encourages proportional credit assignment for partially correct generations.

A.4 GRPO Training Objectives

Our method uses GRPO (Guo et al., 2025) to train LLMs to reason about whiteboard tasks and generate responses that instruct the web interface to update its content accordingly. We design eight task types, each associated with a specific pair of reward functions: a *format reward* to ensure the output follows the expected structured format, and an *accuracy reward* to evaluate whether the produced operation description is correct. Thus, for each training step, one question is sampled, and the corresponding format reward and operation accuracy reward are used to guide optimization. Let $\{o_1, \ldots, o_G\}$ be a group of G response outputs sampled from the current policy π_{old}, and let $\mathbf{r} = (r_1, \ldots, r_G)$ denote the reward scores of these samples. Each r_i is the weighted combination of the format and accuracy rewards for the given task. The rewards are normalized within each group to mitigate variance and stabilize training to yield the estimated advantages $\hat{A}_i$:

$$\hat{A}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r}) + \epsilon}$$

The objective then optimizes the policy model π_θ by maximizing:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min \left\{ \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})} \hat{A}_i, \right. \right. \right.$$
$$\left. \left. \left. \text{clip}\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_i \right\} - \beta D_{\text{KL}}[\pi_\theta \parallel \pi_{\text{ref}}] \right) \right]. \quad (9)$$

Here:

- π_θ is the policy model being optimized,
- $\pi_{\theta_{\text{old}}}$ is the policy from the previous iteration,
- ϵ is the clipping parameter,
- β is the KL penalty coefficient that regularizes against the reference policy π_{ref},
- D_{KL} denotes the KL divergence.

This formulation ensures that the learning process is guided by verifiable, task-specific rewards. By combining format and accuracy rewards, our training procedure effectively aligns LLMs to manipulate digital whiteboard objects in diverse tasks.

A.5 Additional Experimental Settings

We construct a smaller dataset for the multi-agent system and a larger dataset for the GRPO training process. The smaller dataset A consists of 600 samples. The larger dataset B consists of 6,000 samples for training, 600 samples for validation, and 600 samples for testing.

In the multi-agent interaction system, we use `Qwen2.5-3B-Instruct` as the questioner, `Qwen2.5-VL-3B-Instruct` as the solver, and `GPT-5` (OpenAI, 2025b) as the optimizer by default. In each global iteration, the questioner and solver iterate over all samples in dataset A, after which the optimizer refines the system instructions based on the current instructions and summarized running statistics such as format rewards and operation accuracy. For each sample, the solver generates three candidates; each candidate is evaluated to obtain reward values, and the one with the highest reward is selected. The questioner then generates feedback based on missing or low-scoring components in the evaluated format compliance, and the selected candidate is included in the prompt as chat history in the next round.

In the GRPO training, we use `EasyR1` (Sheng et al., 2025; Zheng et al., 2025) to build our training pipeline. By default, the learning rate is set to 5×10^{-6} and the data type is `bfloat16`. A typical example of the format reward and operation accuracy on the validation subset of dataset B is shown in Figures 5 and 6.

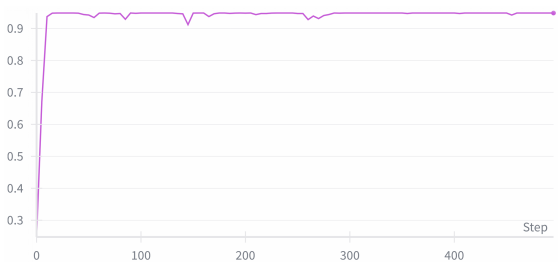

Figure 5: Example Format Reward in GRPO Training.

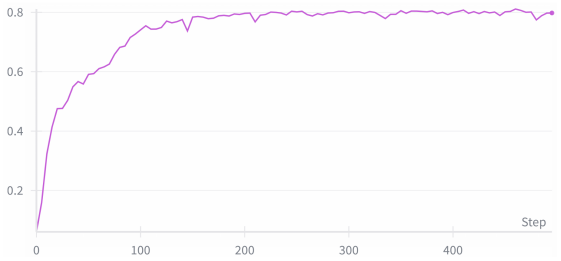

Figure 6: Example Operation Accuracy in GRPO Training.

Model	Inference Time (s)
Qwen2.5-VL-Instruct	
Qwen-3B	4.87
Qwen-7B	10.52
Qwen-32B	36.82
Qwen-72B	44.20
Qwen2.5-Instruct (text-only)	
Qwen2.5-32B	26.83
Qwen2.5-72B	48.59
Qwen3 (thinking mode)	
Qwen3-0.6B	28.61
Qwen3-14B	43.65
Qwen3-32B	67.45
Our pipeline	**18.15**

Table 7: Inference Time (Averaged over eight types of tasks).

A.6 Inference Time and Latency Trade-off Analysis

Our goal is to develop whiteboard agents capable of interacting with users for various purposes such as education (Liu et al., 2024, 2025). The model needs to reason correctly and respond quickly enough to maintain interactive communication with users. We focus on using smaller models rather than larger ones because smaller models may provide advantages in inference time. As shown in Table 7, we conduct experiments to measure the inference time using different models (Yang et al., 2024, 2025b; Bai et al., 2025b; Yang et al., 2025a). The `Qwen2.5-VL-3B-Instruct` model in our pipeline achieves an average inference time of 18.15 seconds.

The structured reasoning protocol increases inference time from 4.87 s (base model) to 18.15 s (our pipeline), a $\sim 4\times$ slowdown. Below we provide a more nuanced analysis of this trade-off.

Model	Time (s)	Accuracy (%)
Qwen2.5-3B (Base)	4.87	1.59
Qwen2.5-7B	10.52	24.74
Ours (3B + Reasoning)	**18.15**	**81.12**
Qwen2.5-72B	44.20	30.78

Table 8: Accuracy-latency Pareto analysis. Our pipeline achieves the best trade-off: $2.6\times$ higher accuracy than the 72B model at $2.4\times$ lower latency.

Accuracy vs. Latency Pareto Frontier. Table 8 contextualizes the latency increase by jointly considering accuracy and inference time for key models. The base `Qwen2.5-VL-3B-Instruct` model is fast (4.87 s) but achieves only 1.59% accuracy, effectively unusable for any practical whiteboard task. Conversely, `Qwen2.5-VL-72B-Instruct` requires 44.20 s but achieves only 30.78% accuracy. Our pipeline occupies a favorable position on the Pareto frontier: it provides 81.12% accuracy at 18.15 s, offering $2.6\times$ the accuracy of the 72B model at $2.4\times$ lower latency.

Cost of Incorrect Actions. In high-stakes UI manipulation scenarios, the cost of an incorrect operation (e.g., deleting the wrong object, placing an element at the wrong position) is typically much higher than the cost of additional thinking time. An incorrect action requires user detection, undo, and re-execution, a cycle that often exceeds the $\sim$13 s latency overhead of our reasoning protocol. We therefore argue that trading speed for reliability is a net positive in agentic whiteboard settings.

Future Mitigation Strategies. We view the current model as a *teacher* for knowledge distillation. The high-quality reasoning traces generated during GRPO training can serve as supervision for training a faster "System 1" student model that retains spatial precision without the runtime overhead. Speculative decoding (Leviathan et al., 2023) and reasoning trace compression are promising complementary directions.

A.7 Cross-Platform Schema Portability

Our framework is designed to be platform-agnostic at the reasoning level. To illustrate the effort required to port our pipeline to a different whiteboard environment, Table 9 compares the JSON schema conventions of TLDraw and Excalidraw for a representative rectangle object.

The core spatial primitives (position, size, rotation, color, and shape type) are semantically iden-

Primitive	TLDraw	Excalidraw
Position	`x`, `y`	`x`, `y`
Size	`props.w`, `props.h`	`width`, `height`
Rotation	`rotation`	`angle`
Color	`props.color`	`strokeColor`
Fill	`props.fill`	`backgroundColor`
Shape type	`props.geo`	`type`
Opacity	`opacity` (0–1)	`opacity` (0–100)

Table 9: Schema Comparison: TLDraw vs. Excalidraw. Core spatial primitives are semantically identical; differences are limited to key naming, nesting depth, and value scaling.

tical across both platforms. The differences are superficial: key naming conventions (e.g., `props.w` vs. `width`), nesting depth (TLDraw uses a `props` sub-object), and minor value scaling (opacity as a $[0, 1]$ float vs. a $[0, 100]$ integer). Adapting our pipeline to Excalidraw would require updating only the web-alignment reward R_{web} (to validate against the Excalidraw schema) and the output template in the prompt. All accuracy rewards (R_{acc}), which operate on abstract spatial properties (Euclidean distances, IoU, angular errors), and the *Identify-Reason-Verify* reasoning protocol transfer without modification.

A.8 Declaration of Generative AI and AI-Assisted Technology Usage

During the preparation of this manuscript, the authors used ChatGPT (OpenAI, 2025a) solely for light grammatical and stylistic refinement of the text. All content was reviewed and edited by the authors as needed, and the authors take full responsibility for the content of the publication.