VLMs Trace Without Tracking: Diagnosing Failures in Visual Path Following

Hyesoo Hong[1] **Min Soo Kim**[1] **Wonje Jeung**[1]
Sangyeon Yoon[1] **Dongjae Jeon**[2] **Albert No**[1]†

[1]Department of Artificial Intelligence, Yonsei University
[2]Department of Computer Science, Yonsei University

Abstract

Vision-language models (VLMs) perform strongly on multimodal benchmarks, yet their ability to reliably follow continuous visual structure remains unclear. We focus on *path following*, a basic operation that requires following a selected path through successive visual continuations. Before considering complex factors such as crossings or overlaps, we first test whether current VLMs can robustly follow a simple continuous line under controlled settings that remove such confounds. Across frontier models, we find that tracing is already fragile, with this vulnerability manifesting as unstable path preservation in the presence of other nearby paths. Analysis of internal representations shows that information needed to determine the correct path continuation becomes weaker, with this degradation already emerging in the visual encoding stage. Common remedies do not resolve this weakness: model scaling provides limited improvement, additional reasoning provides uneven gains at substantial cost, and explicit tracing instructions fail to stabilize performance. The same path-preservation failures and sensitivity to nearby paths persist in visually richer and more realistic settings, suggesting that current VLMs lack a robust mechanism for following a selected path.

1 Introduction

VLMs now achieve strong performance on many multimodal benchmarks [28, 29, 45], yet this progress does not imply mastery of basic visual operations. A growing body of work shows that even frontier VLMs fail on visually simple tasks that humans solve easily [13, 7, 18, 20, 41], often because models lean on language priors, semantic context, or coarse image statistics rather than genuine visual analysis [17, 25, 31, 47, 43]. Elementary capabilities such as individuating objects, sustaining attention across space, and following continuous structure remain underdeveloped [6, 36, 10, 9].

In this work, we investigate whether VLMs can reliably follow a visual path, beginning with its simplest form: a continuous line. This operation matters disproportionately because VLMs are increasingly deployed in settings where they must follow connected visual paths, such as following a route on a transit map [32, 11], tracking a deformable object through a cluttered scene [42], or reading a flow chart [8, 18, 26, 40, 46]. The shared primitive across these tasks is simple: Given a starting point, the model must stay on the selected path as it unfolds through space. As such applications become more common, it is important to ask whether VLMs can reliably perform this basic visual operation rather than merely succeed on high-level task outputs.

Prior works frame VLM failures in path following as limits of nonlocal reasoning under difficult conditions such as long paths, crossings, and visual shortcuts [5, 9]. Before attributing failures to

†Correspondence to: Albert No albertno@yonsei.ac.kr

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

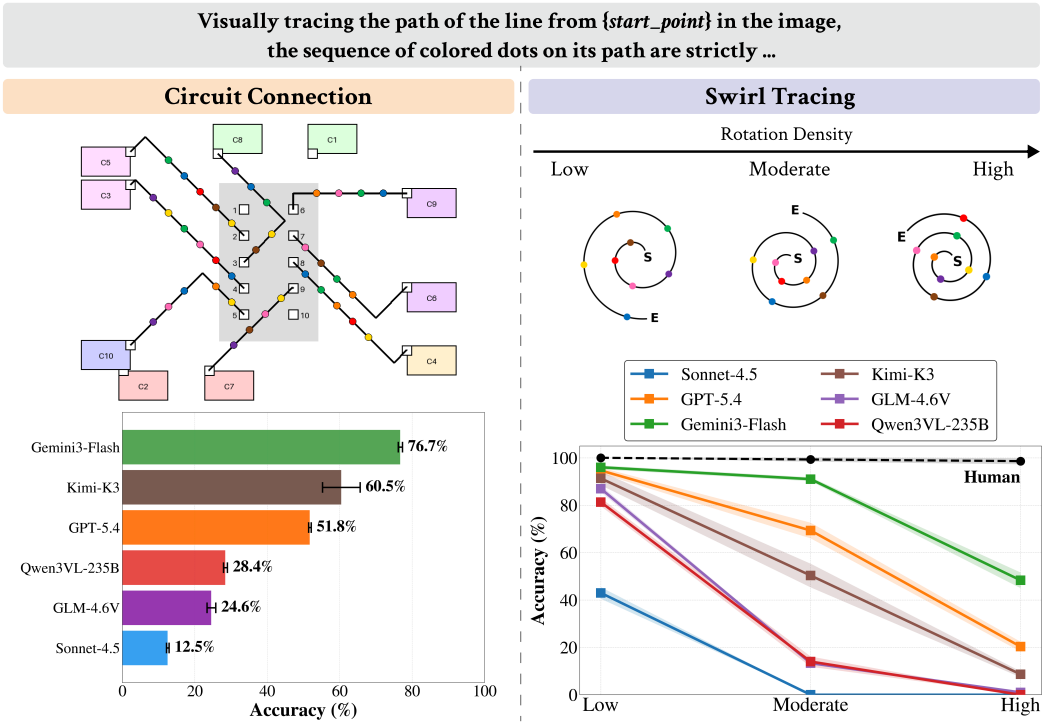


Figure 1: **Two tracing settings and the corresponding model performance.** In both settings, the prompt asks the model to begin from a specified start point and output the sequence of colored dots along the corresponding path in order. Left: Example of the *Circuit Connections* task and model performance across all evaluated frontier models. Right: Examples of *Swirl* tracing at *Low*, *Moderate*, and *High* rotation density and model performance across these three levels.

such complex factors, however, it is necessary to first establish whether VLMs possess the more fundamental ability to follow a simple continuous line. We therefore evaluate six frontier VLMs—Claude Sonnet-4.5, GPT-5.4, Gemini3-Flash, Kimi-K3, GLM-4.6V, and Qwen3-VL-235B—under strictly controlled settings and find that most already struggle to maintain the selected path in the mere presence of nearby paths, even without crossings or overlaps.

Specifically, we first adapt the *Circuit Connections* task from prior path-following work [5], removing crossings, overlaps, and semantic shortcuts to isolate basic path following. Despite this simplification, all models remain below 80% accuracy, with wide variation across models, showing that preserving a selected path is already unreliable. This failure often manifests as an *adjacent-wire jump*, where the model begins on the correct wire but later switches to a nearby wire and continues toward its endpoint. Motivated by this sensitivity to nearby paths, we design the *Swirl* dataset to systematically stress path preservation within one continuous path by varying rotation density. Human accuracy stays near 99% across densities, whereas model accuracy drops by more than 40 percentage points from *Low* to *High* density. This loss appears as increasingly frequent *adjacent-turn jumps*, where models leave the selected spiral path for a neighboring turn at the same angular direction. Together, these results reveal a lack of robust path following, exposed through the systematic destabilization of path preservation when nearby competing paths are introduced.

Our claim is supported by internal analysis and standard-remedy tests. Internal analyses show that next-step information for tacing weakens under controlled variations in nearby-path presence and appearance similarity across the vision encoder and LLM. Standard remedies do not resolve this limitiation: scaling the vision encoder provides limited improvement in connectivity preservation; reasoning shifts from the required tracing procedure toward alternative strategies at substantial computational cost; and explicit tracing instructions fail to stabilize tracing performance. Notably, model-generated tracing instructions closely resemble human tracing procedures, yet yield only limited gains when fed back to the same model, revealing a gap between high-level tracing knowledge and stable visual execution. Case studies on deformable line objects and metro maps show that the same tendency persists in visually richer and more realistic settings.

Overall, our findings from using nearby paths as a stressor provide converging behavioral and representational evidence of a limitation in VLM tracing robustness. Its persistence across controlled perturbations, standard remedies, and realistic settings suggests that overcoming this limitation will require more fundamental solutions than scaling, reasoning, or prompting alone.

2 Related Work

Basic Visual Abilities in VLMs. Recent work shows that strong performance on broad multimodal benchmarks can coexist with failures on visually simple tasks. Prior studies have documented broad shortcomings in the visual abilities of multimodal models [1, 13, 41, 7, 12, 18, 30], as well as failures on basic geometric judgments, visual-only perception, and core visual tasks designed to reduce linguistic shortcuts [36, 23, 9]. Related analyses connect such failures to limitations in binding, correspondence, and spatial reasoning [22, 6, 10, 47, 3]. These findings motivate controlled evaluations that minimize semantic and linguistic shortcuts to probe basic visual abilities. We adopt this approach to study a capability: maintaining a selected path through nearby competing continuations.

Path Following and Nonlocal Visual Reasoning. A related line of work studies path following as a form of nonlocal visual reasoning, where success depends on integrating evidence across distant image regions. Prior work identifies smooth visual search as a weakness, including contour-following and circuit-connection tasks [5]; map-based reasoning tasks also test route planning over transit maps [11]. Contour following has been studied as a long-range dependency in *Pathfinder* [27] and as deformable-object tracing in *HANDLOOM* [42]. Concurrent work, *TraversalBench* [34], studies exact path traversal across factors such as crossings, tortuosity, vertex count, and confounding lines. Our work is complementary, isolating path preservation in a crossing-free, color-only setting and analyzing failures through controlled similarity, internal-state, and reasoning analyses.

Tracing as a Sequential Visual Operation. Classic work on visual cognition provides a precedent for viewing tracing as a sequential operation over connected structure. One account proposes *incremental grouping*: while simple structures can be grouped using local visual features, nearby competing paths introduce clutter and ambiguous continuations, making incremental grouping important for maintaining the selected path through successive local choices [37, 38]. Behavioral curve-tracing studies show that tracing depends on serial attentional processing and becomes harder as paths become longer, more cluttered, or more ambiguous [21, 35, 39, 19]. We use this literature as conceptual grounding: it motivates interpreting VLM tracing errors as failures to remain on the selected path when nearby paths offer competing continuations, rather than as generic spatial mistakes.

3 VLM Tracing Lacks Robust Path Preservation

Before evaluating path following in more complex visual settings, we first ask whether VLMs can follow its simplest form: a single continuous line. We therefore begin with a simplified adaptation of the *Circuit Connections* task [5] to examine basic path following under minimal visual complexity as an initial robustness check. We then introduce *Swirl* to systematically reproduce the failure factor observed in *Circuit Connections* as a controlled stressor and examine how path preservation changes as its strength is progressively increased.

3.1 VLMs Fail Even at Basic Path Following

Path Following Under Minimal Visual Complexity. To isolate basic path following, we adapt the *Circuit Connections* task by removing wire crossings and overlaps, as shown in Figure 1 (left), and place colored dots along each wire to reveal the model's followed path. Given a queried wire, the model must follow it from a port on the breadboard to its endpoint component and report every colored dot encountered in order. A prediction is counted as correct only if it reports the complete ordered sequence and terminates with *"end"*. Detailed task construction is provided in Appendix C.1.

As shown in Figure 1 (left), performance remains limited across all models even in this simplified path-following setting. The best-performing model, Gemini3-Flash, reaches 76.7% accuracy, while GPT-5.4 achieves 51.8% and Sonnet-4.5 performs lowest at 12.5%. Even the strongest model fails

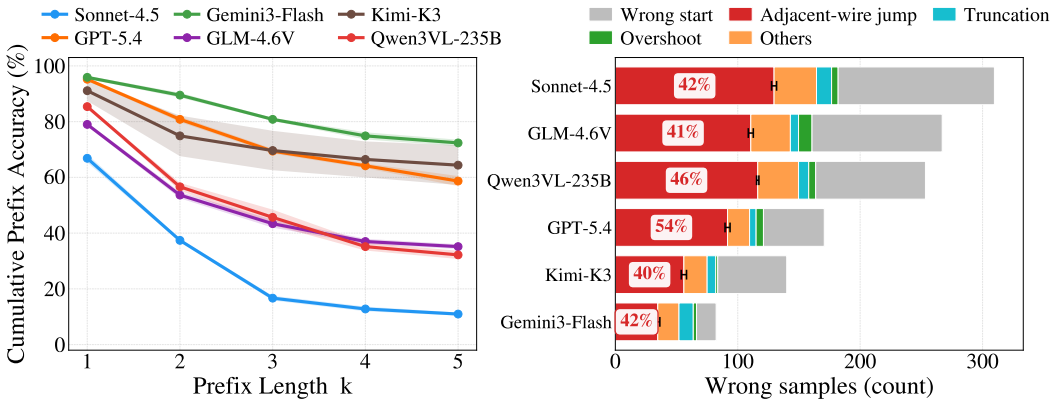


Figure 2: **Path-preservation and error analysis on Circuit Connections.** Left: Cumulative prefix accuracy across tracing steps for wires containing five colored dots across all evaluated models. Right: Error-type distribution for failed traces. Errors are assigned sequentially to the first applicable category for each failed trace: *Wrong start* when the initial dot is incorrect; *Adjacent-wire jump* when the trace switches to a nearby wire within the predefined pixel-distance threshold; *Truncation* when the queried wire is followed correctly but the prediction terminates early; *Overshoot* when the correct sequence is completed but additional dots are generated; and *Others* for all remaining errors.

on nearly one quarter of traces, while the wide performance spread shows that basic path following remains unreliable across frontier VLMs. Understanding what drives these failures is therefore critical for identifying the underlying weakness in path preservation more precisely.

Path Preservation Breaks Down During Tracing. To understand the source of this unstable performance, we analyze cumulative accuracy as tracing progresses and the types of errors that occur. Cumulative prefix accuracy at step k is defined as the proportion of traces whose first k predicted dots exactly match the ground-truth sequence. Therefore, an error at any earlier step also makes all subsequent prefix steps incorrect. As shown in Figure 2 (left), models often recover the initial segment correctly, but cumulative prefix accuracy consistently declines as tracing proceeds, even under this simplified setting. This pattern shows that extending the correct sequence becomes progressively more difficult as tracing proceeds. Full cumulative prefix accuracy results for wires containing different numbers of dots are provided in Appendix C.2.

We next analyze the error types underlying this cumulative decline. As shown in Figure 2 (right), *Wrong start* errors are common, indicating that some failures occur before tracing is correctly initialized. Our main focus, however, is the *Adjacent-wire jump*, where a model starts on the correct wire but subsequently switches to a nearby wire and continues tracing from there. These jumps account for 40–54% of all errors across models, making them one of the dominant failure modes. Thus, even when the correct path is initially identified, models frequently fail to preserve it by switching to nearby paths. Details of the error-type classification are provided in Appendix C.3.

3.2 Swirl: A Controlled Stress Test of Path Preservation

Systematically Varying Nearby-Path Proximity. Previous recurring failure makes nearby paths a natural controlled stressor for testing whether VLM path following remains stable. Therefore, we design *Swirl* as a controlled stress test of path preservation. Using a single continuous spiral and keeping path length, stroke width, and dot count fixed, we vary the spacing between neighboring turns by progressively increasing the number of turns. We use this inter-turn spacing to define *Low*, *Moderate*, and *High* rotation-density levels (Figure 1, right), with smaller spacing corresponding to higher density and a stronger nearby-path stressor. The design also avoids introducing multiple independent paths and removes factors unrelated to path following, such as domain-specific objects, labels, ports, and background structure. Each image contains the same eight uniquely colored, non-repeating dots as trace verifiers. Detailed dataset construction is provided in Appendix D.1.

Tracing Degrades as the Nearby-Path Stressor Strengthens. If models possessed robust path-following ability, increasing rotation density should not systematically disrupt path preservation. As

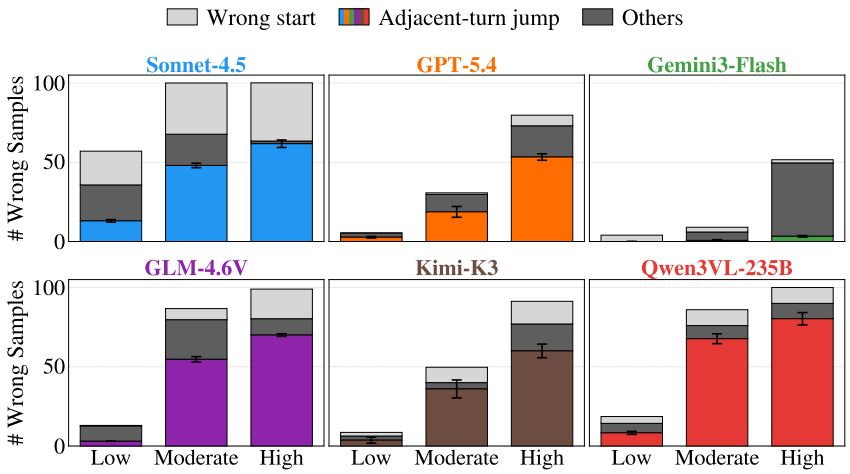


Figure 3: **Error analysis on Swirl.** Shown errors are assigned sequentially to the first applicable category: *Wrong start* when the initial dot is incorrect; *Adjacent-turn jump* when the trace moves to a dot located at the same angular direction on the immediately adjacent outer turn of the spiral; and *Others* for the remaining errors. *Truncation* and *Overshoot* were rarely observed and are omitted.

shown in Figure 1 (right), the task remains trivial for humans across density levels, with accuracy changing only from 100.0% to 98.6%. Against this baseline, the sharp degradation of all evaluated VLMs is striking: full-sequence accuracy falls from 94.7% to 20.3% for GPT-5.4, from 96.0% to 48.3% for Gemini3-Flash, and from 91.3% to 8.7% for Kimi-K3 between *Low* and *High* density. This contrast highlights the lack of path-following robustness in current VLMs. Human-baseline details, full prefix-accuracy, and error-type classification are provided in Appendices D.2 to D.4.

As rotation density increases, a growing fraction of failures in most models, except Gemini3-Flash, take the form of an *adjacent-turn jump* (Figure 3). In this failure, instead of following the next dot along the current spiral turn, the model jumps outward to a dot at the same angular direction on the immediately adjacent outer turn and continues tracing from there. The systematic increase in adjacent-path jumps as nearby-path competition grows demonstrates its effectiveness as a controlled stressor for revealing the fragility of path preservation in current VLMs. We further verify that these results are not primarily driven by insufficient input resolution: no patch contains multiple colored dots, and doubling the input resolution yields no substantial recovery, as described in Appendix D.6.

4 Diagnosing Path Preservation Under Nearby-Path Competition

We next examine how nearby, non-overlapping paths affect tracing and its internal representations, focusing on Qwen3-VL-8B [4]. We additionally conduct the same analysis with InternVL3.5-8B [44], with the additional results and full experimental details provided in Appendix E.

4.1 Effect of Nearby-Path Presence on Path Preservation

We treat the presence and number of nearby competing paths as the controlled variable, comparing *Baseline*, *Near-1*, and *Near-2* (Figure 4) while keeping the queried path fixed. Tracing accuracy decreases from 100% in *Baseline* to 98% in *Near-1* and 28% in *Near-2*, revealing that tracing is highly sensitive to nearby paths. To examine how tracing-related information evolves through the model, we apply simple linear probes at each evaluated vision-encoder block and LLM layer using representations of the current dot's image patch. *Probe A* predicts the color of the dot within the current patch, testing whether the model retains information about its current position on the queried path, while *Probe B* predicts the color of the next dot along the queried path, testing whether the representation contains information needed to follow the correct continuation.

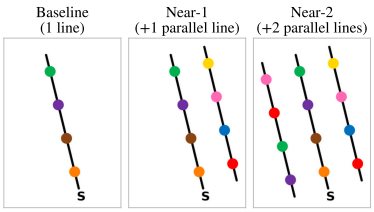
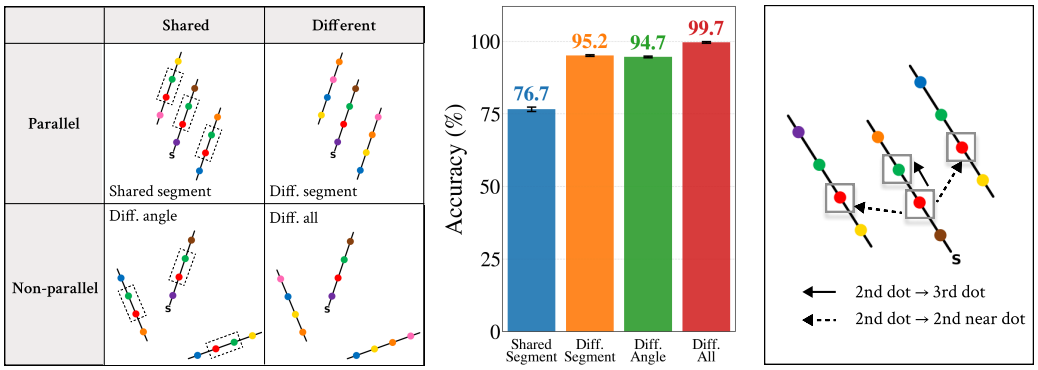


Figure 4: Examples from the Baseline, Near-1, and Near-2.



(a) Four controlled similarity conditions (b) Tracing accuracy for each condition (c) Attention comparison setup

Figure 5: **Effect of nearby-path similarity on tracing performance and internal selectivity.** (a) Four controlled similarity conditions varying whether nearby paths share local structure and orientation with the queried path. (b) Tracing accuracy across conditions increases as local similarity between the queried and nearby paths decreases. (c) Attention comparison setup, measuring attention from the second dot on the queried path to the correct third dot versus the nearby dot.

Probe A remains at 100% across all three conditions, whereas Probe B accuracy at the strongest vision-encoder block decreases from 57.0% to 48.4% and 36.6% as nearby-path competition increases. These results suggest that the mere presence of nearby paths can weaken continuation information while leaving local dot identity intact, with further degradation as more paths are introduced, already evident in the vision encoder. Detailed task construction, experimental settings, and probing analyses are provided in Appendix E.3. We further confirm that replacing nearby paths with non-line controls has little effect on tracing performance, suggesting that nearby visual content alone does not account for the observed failures, as detailed in Appendix F.

4.2 Effect of Nearby-Path Similarity on Path Preservation

Beyond the presence of nearby paths, we further examine how their visual and structural similarity to the queried path affects path preservation. We construct four conditions (Figure 5a) in which each path contains four colored dots: the first verifies whether the model starts on the correct path, the second and third manipulate local continuation similarity, and the final dot has a path-specific color that reveals which path the model follows. In *Shared segment*, nearby paths contain matching color steps at the second and third positions and run parallel to the queried path, creating the strongest local similarity. In *Different segment*, matching color steps are removed while the paths remain locally parallel and retain directional similarity. In *Different angle*, the matching color steps are retained but the nearby paths differ in orientation from the queried path. Finally, *Different all* removes matching color steps and shared orientation, resulting in the weakest resemblance to the queried path.

As shown in Figure 5b, accuracy drops to 76.7% in the *Shared segment* condition, compared to 99.7% in *Different all*. This suggests that the model's tracing becomes less stable not only in the presence of nearby paths but also as their structural similarity to the queried path increases.

Nearby-Path Similarity Weakens True-Continuation Preference. To examine the internal signals underlying the behavioral accuracy changes, we analyze two complementary measures. *Attention preference* in the vision encoder measures whether attention from the current queried-path position favors the true next dot over a competing dot on the nearby path, while *representational preference* across the vision encoder and LLM decoder measures whether the hidden state at the current queried-path position is closer to the true continuation than to the nearby-path alternative. Together, these diagnostics characterize how increasing nearby-path similarity is reflected in the model's internal signals for path preservation at both the attention and representation levels. The compared regions are illustrated in Figure 5c; implementation details are provided in Appendix G.

Let q_2 and q_3 denote the second and third dots on the queried path, and $d_{1,2}$ and $d_{2,2}$ the corresponding second dots on the two nearby paths.

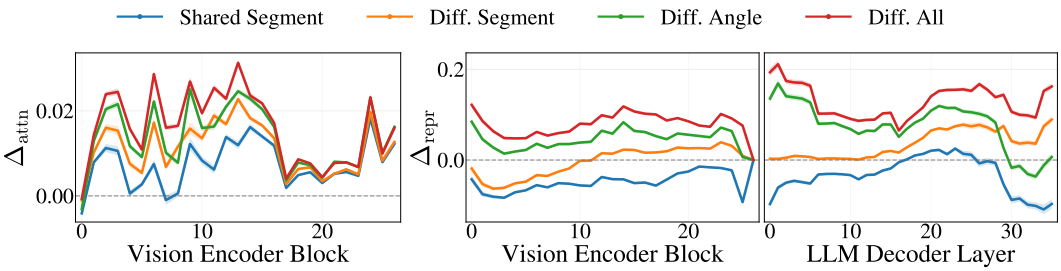


Figure 6: **Internal selectivity under increasing shared local structure.** Left: block-wise vision attention margin (Δ_{attn}) from *Red* on the queried path to the correct next dot *Green* versus the nearby competing distractor *Dist-Red*. Middle and right: cosine similarity margin (Δ_{repr}) between the current queried-path state *Red* and the true next state *Green* versus *Dist-Red*, shown separately for the vision encoder and LLM decoder, respectively.

For attention preference, we compute

$$\Delta^{(\ell)}_{\text{attn}} = a^{(\ell)}_{q_2 \to q_3} - \frac{1}{2}\left(a^{(\ell)}_{q_2 \to d_{1,2}} + a^{(\ell)}_{q_2 \to d_{2,2}}\right).$$

For representational preference, we compute

$$\Delta^{(\ell)}_{\text{repr}} = \cos\big(\tilde{h}^{(\ell)}_{q_2}, \tilde{h}^{(\ell)}_{q_3}\big) - \max\Big\{\cos\big(\tilde{h}^{(\ell)}_{q_2}, \tilde{h}^{(\ell)}_{d_{1,2}}\big), \cos\big(\tilde{h}^{(\ell)}_{q_2}, \tilde{h}^{(\ell)}_{d_{2,2}}\big)\Big\},$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity and $\tilde{h}^{(\ell)}$ denotes the normalized hidden representation at layer ℓ. In both preferences, positive values indicate preference for the true continuation, values near zero indicate weak discrimination, and negative values indicate association with the nearby path.

If path following were robust, the model should maintain a consistent preference for the true continuation across all four conditions, regardless of nearby-path appearance. However, Figure 6 (left) shows that attention preference varies systematically across early-to-middle vision-encoder blocks despite the queried path remaining unchanged. The preference is lowest in the *Shared segment* condition and highest in *Different all*. A similar pattern appears in representational preference across both the vision encoder and LLM decoder (Figure 6, middle and right), suggesting that nearby-path similarity can affect how tracing-relevant information is processed at both stages. These internal patterns align with the behavioral trend observed above (Figure 5b), providing further experimental evidence consistent with limited robustness of path preservation in current VLMs.

5 Standard Remedies Do Not Resolve Fragile Visual Tracing

We test three commonly used approaches for improving VLM performance: scaling up the model, increasing reasoning effort, and prompt engineering. We therefore ask whether these standard remedies can compensate for the lack of robustness in visual tracing under nearby visual competition.

5.1 Vision-Encoder Scaling Does Not Improve Connectivity Preservation

Our preceding probing results suggest that tracing-relevant information is already weakened in the vision encoder, motivating a scaling analysis at the encoder level. We therefore apply Probe B (in Section 4.1), which predicts the next dot's color from the image patch containing the current dot, across models with different vision-encoder scales from the Qwen3-VL (302M and 411M) and Gemma 4 (151M and 545M) families (Figure 7). Despite these increases in vision-encoder size, larger and smaller models show little difference in Probe B accuracy, and the same degradation persists when nearby paths are introduced. Thus, although the vision encoder becomes larger, the improvement in connectivity representation remains limited. We further examine full-model scaling across the Qwen3-VL family, jointly scaling the vision encoder and LLM, and observe similarly limited gains across scales on the Swirl task; detailed results are provided in Appendix D.7.

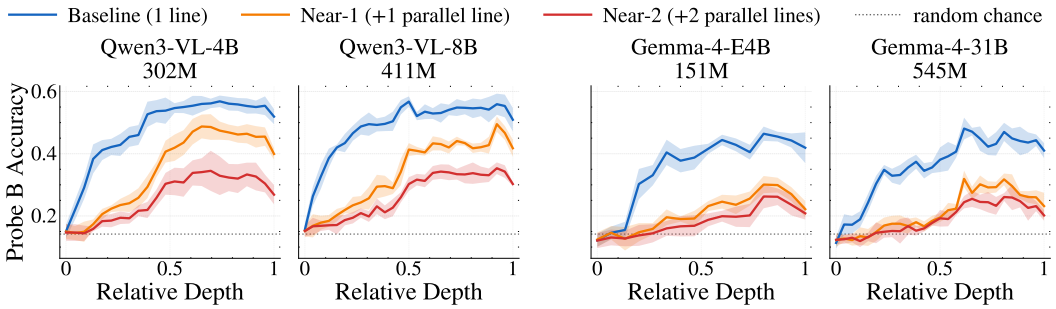


Figure 7: **Probe B across vision-encoder scales.** Probe B accuracy across vision-encoder blocks for model sizes within the Gemma 4 and Qwen3-VL families under *Baseline*, *Near-1*, and *Near-2*, which vary the number of nearby paths. Probe B measures how well the representation of the queried-path dot encodes information about the next dot's color needed to continue tracing the path.

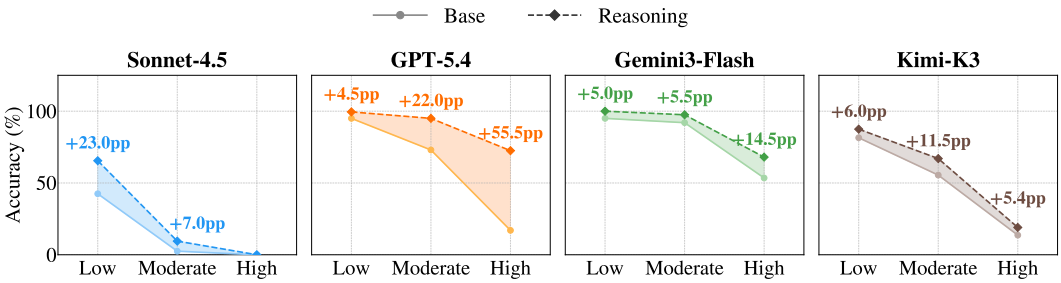


Figure 8: **Effect of reasoning on Swirl performance.** Accuracy of *Base* (non-reasoning) and *Reasoning* models across Swirl difficulty levels, highlighting reasoning-induced performance changes.

5.2 Reasoning Does Not Reliably Compensate for Tracing Failures

We investigate whether longer and deeper reasoning can support tracing and compensate for its lack of robustness. We evaluate reasoning-enabled VLMs on the *Swirl* task (Figure 1, right). Results for additional models, including GLM-4.6V and Qwen3-VL-235B, are provided in Appendix H.

Reasoning Provides Uneven Benefits at Substantial Cost. As shown in Figure 8, most models show only limited gains from reasoning, with GPT-5.4 as a notable exception, improving by 55.5 pp at *High* density. However, these gains come at substantial computational cost. As rotation density rises, all models consume increasingly more tokens and exhibit more explicit self-corrections, such as *"wait,"* or *"let me check"* (Table 1), despite the task requiring the same tracing procedure and output format. Even GPT-5.4, which achieves the largest accuracy gain, increases its average token usage from 501 at *Low* density to 3,443 at *High*, nearly a sevenfold increase. This sharp growth in reasoning length and repeated reconsideration warrants closer examination of how the reasoning process itself changes. Further details on self-correction frequency are provided in Appendix H.1.

Reasoning Strategies Shift with Density. The tracing task requires the same sequential procedure at every density level: starting from S, the model must identify the next connected dot and continue along the same path until reaching E. At *Low* density, reasoning resembles this procedure through direct order inference or step-by-step spatial descriptions, whereas higher density increasingly induces qualitatively different, non-sequential strategies (Table 2). For example, GPT-5.4 directly lists the path order at *Low* density but shifts at *High* density toward geometric reconstruction, such as decomposing the spiral into multiple rings. Kimi-K3 uses spatial information at *Low* density to describe how one point connects to the next, but at *High* density instead attempts to reconstruct the spiral using inferred centers, radial coordinates, and radius-based ordering, relying on geometric criteria that do not necessarily preserve line connectivity. The sharp increase in token usage and self-correction is therefore largely driven by these substitute strategies, as additional reasoning is spent on geometric reconstruction rather than sequential tracing. Thus, changing only rotation density substantially

Table 1: **Reasoning length and self-correction frequency on Swirl.** Average reasoning-token usage and self-correction markers per sample are reported across density levels for all evaluated reasoning models. Darker shading indicates a larger increase from each model's *Low*-density baseline.

	Reasoning tokens			Self-corrections / sample		
Reasoning Model	**Low**	**Moderate**	**High**	**Low**	**Moderate**	**High**
Sonnet-4.5	390	510	537	4.20	6.14	6.38
GPT-5.4	501	1,373	3,443	0.12	0.53	1.45
Gemini3-Flash	557	800	1,337	0.47	0.88	1.66
GLM-4.6V	477	706	854	9.25	15.64	19.49
Kimi-K3	2,565	2,984	4,631	9.44	10.07	16.48
Qwen3-VL-235B	657	998	1,311	10.23	16.70	24.23

Table 2: **Representative reasoning patterns across Swirl density levels.** Reasoning traces from GPT-5.4 and Kimi-K3 illustrate a shift from brief sequential tracing at *Low* density, where models follow the visible path from one continuation to the next (blue), toward geometric reconstruction at *High* density, where models infer global structure using loops, radii, or spatial layout (orange).

Model	**Density**	**Representative reasoning summary**
GPT-5.4	Low	"**The line goes right/upward to** purple **, curves to** blue **, then left to** orange **, moving down-left to** brown **, down-right to** yellow **, up-right to** pink **, up-left to** green **, left to** red **, and finally down-left to** E."
	High	"**I see three loops**: the **inner ring** has red and pink, the **middle** goes through yellow and blue, and ... ends at E. I need to figure out **how the spiral transitions from the first to the second loop**..."
Kimi-K3	Low	"Actually after blue at bottom, **line continues to the right and up along outer right side to** purple (453,355) **, then up-left to** pink (429,211), . . . **Let's determine exact order maybe** S **->** yellow **->** orange ... **->** red **->** brown **->** E."
	High	"Black spiral roughly centered at (330,320), radii: **inner ∼42, middle ∼75, outer ∼110**. **Let's reconstruct spiral mathematically** from endpoints and dots. . . . path could go outward in spiral, **encountering dots by radius order:** red (**r27**), orange (**r32**), brown (**r56**), ..., purple (**r87**), yellow (**r96**), green (**r100**)."

alters how models approach the same tracing task, a shift that is difficult to reconcile with a stable and robust tracing mechanism. Additional reasoning examples are provided in Appendix H.4.

5.3 Explicit Tracing Instructions Do Not Improve Tracing Robustness

We next test whether explicitly specifying how to trace the path can stabilize the reasoning shifts observed above and improve tracing robustness. We compare three prompting conditions. *Standard* uses the original tracing prompt without procedural guidance. *Human-instructed* provides a tracing procedure explicitly specified by a human. For *Model-generated*, each model is shown a Swirl example and asked to describe a general tracing procedure that could be applied consistently across different Swirl instances and across rotation densities; this procedure is then provided back to the same model during evaluation. The full prompts and accuracy results are provided in Appendix I.

As shown in Figure 9, neither human- nor model-generated instructions provide substantial performance gains or stabilize performance as density increases. Token usage continues to rise sharply with density, and reasoning still shifts from sequential tracing toward global reconstruction strategies (Appendix I.1). Thus, explicitly specifying the tracing procedure does not recover robust tracing in practice or stabilize the reasoning process. Notably, the model-generated instructions themselves largely recover the core human-designed tracing procedure: starting from S, following visible connectivity, and proceeding along the selected path. Models therefore appear to possess high-level procedural knowledge of tracing, yet fail to execute that procedure robustly as rotation density changes, revealing a gap between *knowing what to do* and *being able to execute it visually*.

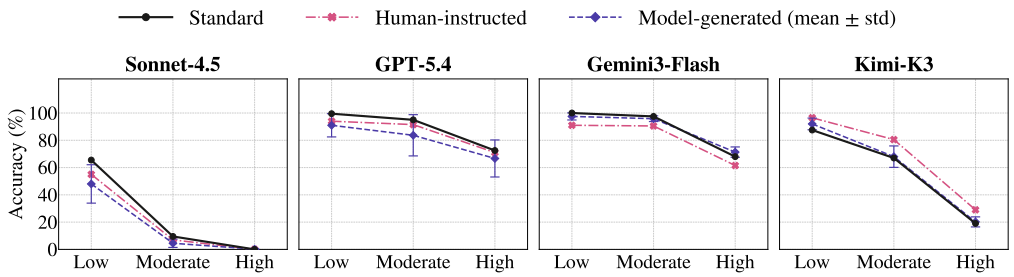


Figure 9: **Effect of explicit tracing instructions on model accuracy.** *Standard* uses the original prompt without procedural guidance; *Human-instructed* provides a human-designed explicit sequential tracing procedure; and *Model-generated* uses model-specific instructions, reporting mean accuracy across three generated versions with standard-deviation error bars per model.

6 Case Study: Tracing Failures in Real-World Settings

In previous sections, we stress-tested diverse VLMs and found that they fail to robustly preserve the selected path as tracing progresses. We examine whether the same path-preservation failure appears in more realistic tracing settings, where multiple structures cross, overlap, or run in close proximity. We evaluate two such settings: tangled cables from the HANDLOOM dataset [42], and metro maps with realistic diagrams (See Figure 10).

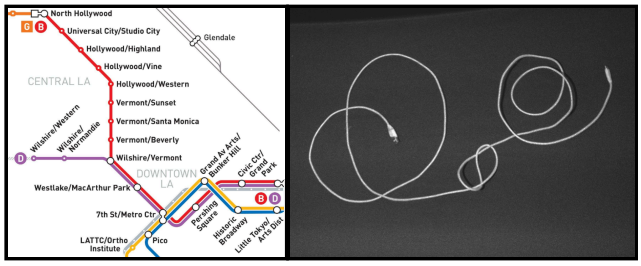


Figure 10: **Realistic tracing examples.** Metro maps (left) and tangled HANDLOOM cables (right), requiring tracing through crossings, overlaps, and nearby competing paths.

As in our controlled experiments, performance remains unreliable in these realistic settings under richer visual conditions, with models exhibiting switches from the selected path to nearby visually competing paths. In the tangled-cable setting, we observe internal signal leakage toward an adjacent segment rather than remaining concentrated on the next dot along the selected path. In the metro map case, reasoning again falls back on substitute strategies, such as pruning candidate station names rather than tracing the line itself. The consistency of this behavior across both types of scenes indicates that the failures we identify are not artifacts of synthetic tasks, but reflect fragile path preservation in VLMs. Full experimental results for both datasets are provided in Appendix J.

7 Conclusion

We studied *path following* to test whether current VLMs can robustly maintain a selected path during tracing. Across controlled settings, tracing performance degrades and path preservation becomes increasingly unstable in the presence of nearby paths and as their influence grows, while internal probing shows weakened information about the correct continuation despite preserved current-path information. Standard remedies, including vision-encoder scaling, reasoning, and explicit tracing instructions, do not reliably resolve this limitation. More broadly, as VLMs are increasingly deployed in settings that demand reliable path following, addressing this failure becomes a practical necessity. We hope this work provides a concrete target for future research on robust visual tracing.

Acknowledgment

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00457882, AI Research Hub Project), IITP grant funded by the Korean Government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University)), and KCHIPS (Korea Collaborative & High-tech Initiative for Prospective Semiconductor Research) (RS2024-00405946, 24052- 15TC) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea).

Competing interests

The authors declare that, during the 36 months preceding submission, they had no financial relationships outside the submitted work that could reasonably be perceived as influencing this research.

References

[1] Haider Al-Tahan, Quentin Garrido, Randall Balestriero, Diane Bouchacourt, Caner Hazirbas, and Mark Ibrahim. Unibench: Visual reasoning requires rethinking vision-language beyond scaling. In *NeurIPS*, 2024.

[2] Anthropic. Introducing claude sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5, 2025. Accessed: 2026-04-07.

[3] Rim Assouel, Declan Campbell, Yoshua Bengio, and Taylor Webb. Visual symbolic mechanisms: Emergent symbol processing in vision language models. In *ICLR*, 2026.

[4] 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, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

[5] Shmuel Berman and Jia Deng. Vlms have tunnel vision: Evaluating nonlocal visual reasoning in leading vlms. In *NeurIPS*, 2025.

[6] Declan Campbell, Sunayana Rane, Tyler Giallanza, Nicolò De Sabbata, Kia Ghods, Amogh Joshi, Alexander Ku, Steven M Frankland, Thomas L Griffiths, Jonathan D Cohen, et al. Understanding the limits of vision language models through the lens of the binding problem. In *NeurIPS*, 2024.

[7] Shivam Chandhok, Wan-Cyuan Fan, Vered Shwartz, Vineeth N Balasubramanian, and Leonid Sigal. Response wide shut? surprising observations in basic vision language model capabilities. In *ACL*, 2025.

[8] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024.

[9] Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Yiping Bao, et al. Babyvision: Visual reasoning beyond language. In *ICML*, 2026.

[10] Shiqi Chen, Tongyao Zhu, Ruochen Zhou, Jinghan Zhang, Siyang Gao, Juan Carlos Niebles, Mor Geva, Junxian He, Jiajun Wu, and Manling Li. Why is spatial reasoning hard for vlms? an attention mechanism perspective on focus areas. In *ICML*, 2025.

[11] Sicheng Feng, Kaiwen Tuo, Song Wang, Lingdong Kong, Jianke Zhu, and Huan Wang. Rewardmap: Tackling sparse rewards in fine-grained visual reasoning via multi-stage reinforcement learning. In *ICLR*, 2026.

[12] Stephanie Fu, Tyler Bonnen, Devin Guillory, and Trevor Darrell. Hidden in plain sight: Vlms overlook their visual representations. In *COLM*, 2025.

[13] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *ECCV*, 2024.

[14] Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

[15] GLM-V Team. GLM-4.5V and GLM-4.1V-Thinking: Towards Versatile Multimodal Reasoning with Scalable Reinforcement Learning, 2025. URL https://arxiv.org/abs/2507.01006.

[16] Google. Gemini 3 flash. https://deepmind.google/models/gemini/flash/, 2025. Accessed: 2026-04-27.

[17] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *CVPR*, 2024.

[18] Yifan Hou, Buse Giledereli, Yilei Tu, and Mrinmaya Sachan. Do vision-language models really understand visual language? In *ICML*, 2025.

[19] Roos Houtkamp, Henk Spekreijse, and Peter R Roelfsema. A gradual spread of attention during mental curve tracing. *Perception & Psychophysics*, 2003.

[20] Kung-Hsiang Huang, Can Qin, Haoyi Qiu, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. Why vision language models struggle with visual arithmetic? towards enhanced chart and geometry understanding. In *Findings of ACL*, 2025.

[21] Pierre Jolicoeur, Shimon Ullman, and Marilynn Mackay. Curve tracing: A possible basic operation in the perception of spatial relations. *Memory & Cognition*, 1986.

[22] Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? investigating their struggle with spatial reasoning. In *EMNLP*, 2023.

[23] Ryo Kamoi, Yusen Zhang, Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, and Rui Zhang. Visonlyqa: Large vision language models still struggle with visual perception of geometric information. In *COLM*, 2025.

[24] Kimi Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

[25] Kang-il Lee, Minbeom Kim, Seunghyun Yoon, Minsung Kim, Dongryeol Lee, Hyukhun Koh, and Kyomin Jung. Vlind-bench: Measuring language priors in large vision-language models. In *Findings of NAACL*, 2025.

[26] Yunxin Li, Baotian Hu, Haoyuan Shi, Wei Wang, Longyue Wang, and Min Zhang. Visiongraph: Leveraging large multimodal models for graph theory problems in visual context. In *ICML*, 2024.

[27] Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated recurrent units. In *NeurIPS*, 2018.

[28] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, 2024.

[29] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *ICLR*, 2024.

[30] Dezhi Luo, Yijiang Li, Maijunxian Wang, Tianwei Zhao, Bingyang Wang, Siheng Wang, Pinyuan Feng, Pooyan Rahmanzadehgervi, Ziqiao Ma, and Hokin Deng. Vision language models cannot reason about physical transformation. In *ICML*, 2026.

[31] Tiange Luo, Ang Cao, Gunhee Lee, Justin Johnson, and Honglak Lee. Probing visual language priors in vlms. In *ICML*, 2025.

[32] Srija Mukhopadhyay, Adnan Qidwai, Aparna Garimella, Pritika Ramu, Vivek Gupta, and Dan Roth. Unraveling the truth: Do vlms really understand charts? a deep dive into consistency and robustness. In *Findings of EMNLP*, 2024.

[33] OpenAI. Introducing gpt-5.4. https://openai.com/index/introducing-gpt-5-4/, 2026. Accessed: 2026-04-07.

[34] Clara Petrova, Zhuo Chen, and Marin Soljačić. Traversalbench: Challenging paths to follow for vision language models. *arXiv preprint arXiv:2604.10999*, 2026.

[35] Richard Pringle and Howard E Egeth. Mental curve tracing with elementary stimuli. *Journal of Experimental Psychology: Human Perception and Performance*, 1988.

[36] Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *ACCV*, 2024.

[37] Pieter R Roelfsema. Cortical algorithms for perceptual grouping. *Annual Review of Neuroscience*, 2006.

[38] Pieter R Roelfsema and Roos Houtkamp. Incremental grouping of image elements in vision. *Attention, Perception, & Psychophysics*, 2011.

[39] H. Steven Scholte, Henk Spekreijse, and Pieter R. Roelfsema. The spatial profile of visual attention in mental curve tracing. *Vision Research*, 2001.

[40] Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Manya Wadhwa, Zeyu Liu, Zayne Sprague, et al. Chartmuseum: Testing visual reasoning capabilities of large vision-language models. In *NeurIPS*, 2025.

[41] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *CVPR*, 2024.

[42] Vainavi Viswanath, Kaushik Shivakumar, Mallika Parulekar, Jainil Ajmera, Justin Kerr, Jeffrey Ichnowski, Richard Cheng, Thomas Kollar, and Ken Goldberg. Handloom: Learned tracing of one-dimensional objects for inspection and manipulation. In *CoRL*, 2023.

[43] An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. Vision language models are biased. In *ICLR*, 2026.

[44] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025. URL https://arxiv.org/abs/2508.18265.

[45] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *CVPR*, 2024.

[46] Xiao Zhang, Dongyuan Li, Liuyu Xiang, Yao Zhang, Cheng Zhong, and Zhaofeng He. Do mllms really understand the charts? *arXiv preprint arXiv:2509.04457*, 2025.

[47] Yikang Zhou, Tao Zhang, Shilin Xu, Shihao Chen, Qianyu Zhou, Yunhai Tong, Shunping Ji, Jiangning Zhang, Lu Qi, and Xiangtai Li. Are they the same? exploring visual correspondence shortcomings of multimodal llms. In *ICCV*, 2025.

A Limitations

Our study focuses on a specific visual primitive: preserving the identity of a selected path under nearby local competition. The controlled circuit and Swirl tasks are designed to isolate this operation, rather than to span the full diversity of visual reasoning. The color-dot sequence interface similarly operationalizes step-by-step tracing in a measurable form. While this abstracts away many properties of natural visual tasks, our HANDLOOM and metro-map evaluations provide complementary evidence that the same failure pattern appears in richer tracing settings. Our proprietary-model results should be interpreted with the usual caveat that closed-source systems may change across API versions and inference settings. Hidden reasoning configurations may also affect both accuracy and generated reasoning traces. We therefore emphasize qualitative trends and recurring error modes in addition to exact performance numbers. Finally, our failure-mode interpretations are based on interventions and systematic error patterns rather than full mechanistic identification. For open-source models, internal diagnostics provide supporting evidence that nearby distractors compete with the true continuation. For closed-source models, where internal states are unavailable, we treat reasoning traces and errors as behavioral evidence rather than direct evidence of internal mechanisms.

B Additional Implementation Details

B.1 Evaluated Models

We evaluate a mixture of proprietary and open-source vision-language models. Depending on the model and experiment, we access models through provider APIs or OpenRouter, or run publicly available Hugging Face checkpoints locally. Local checkpoints are used where access to internal model representations is required. All models are used in accordance with the corresponding public release licenses, model cards, API terms, or service terms.

The evaluated models are as follows:

- **GPT-5.4** [33]
- **Claude Sonnet 4.5** [2]
- **Gemini3-Flash** [16]
- **Kimi-K3** [24]
- **GLM-4.6V** [15]
- **Qwen3-VL-4B-Instruct** [4]
- **Qwen3-VL-8B-Instruct** [4]
- **Qwen3-VL-8B-Thinking** [4]
- **Qwen3-VL-32B-Instruct** [4]
- **Qwen3-VL-235B-A22B-Instruct** [4]
- **Qwen3-VL-235B-A22B-Thinking** [4]
- **InternVL3.5-8B-Instruct** [44]
- **Gemma4-E4B** [14]
- **Gemma4-31B** [14]

B.2 Inference Details

Unless otherwise specified, we use the default generation settings for each model. GPT-5.4 is accessed through the official OpenAI API, while other API-based evaluations use OpenRouter. For models that support a *medium* reasoning-effort setting, we use *medium*; otherwise, we use the model's native reasoning configuration. We run each Circuit Connections and Swirl evaluation three times and report the mean accuracy and standard deviation across runs (Appendix D.7).

Compute Resources. Proprietary-model experiments were run through external model APIs. Open-source model evaluations and internal analyses were run on NVIDIA L40S GPUs. The internal attention and representation analyses used a single L40S GPU per run. We did not train or fine-tune any VLM; only the diagnostic linear probes were trained.

C Circuit Connections

C.1 Circuit Connections Task Construction

We adapt the *Circuit Connections* task from prior work on smooth visual search [5]. The original task asks the model to identify the component connected to a queried breadboard port by following a wire through a circuit diagram. In that setting, wires are visually distinguished by their colors, ports are shown as circular terminals, and the answer only requires the final connected component label. The original generator also allows a broader range of visual configurations, including crossing wires.

Our version modifies this task to focus more directly on basic path following. First, we render all wires in the same color, so the model cannot solve the task by matching wire color across the image. Second, we remove wire crossings and overlaps, and reject candidate wires that collide with existing wires, pass through unrelated ports, overlap labels, cross component interiors, or run along the outline of the breadboard or components. This reduces explicit topological ambiguity and makes the task depend primarily on following the queried wire itself. Third, we replace circular ports with square white ports to make port locations visually explicit and consistent with the prompt. Finally, we place colored dots along each wire and ask the model to report the ordered dot sequence before giving the connected component label. This changes the task from endpoint identification to directly verifiable sequential tracing. Representative examples from our modified task are shown in Figure 11.

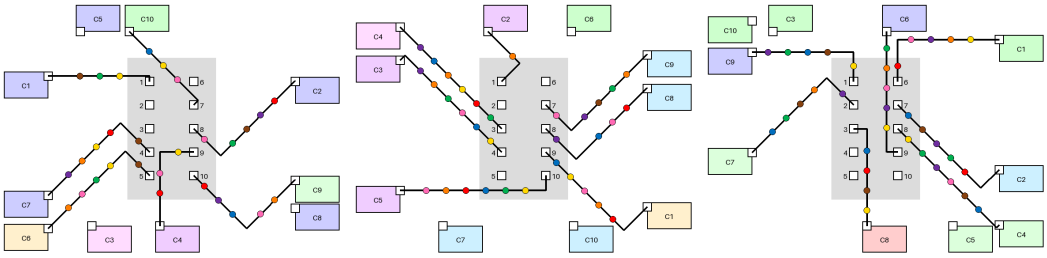


Figure 11: **Examples from the modified Circuit Connections task.** Each example contains a central breadboard with numbered square ports, external component labels, single-color wires, and colored dots placed along each wire. The model is queried with a breadboard port and must report the colored dots on the corresponding wire in order, followed by the connected component label.

Each image contains a central breadboard, shown as a gray rectangle with numbered square ports, and several external components labeled with identifiers such as C1, C2, and so on. A queried wire is defined as a connected line that links one breadboard port to one component port. Each queried wire connects exactly two endpoints: the queried breadboard port and the component at the other end.

Colored dots are placed along each wire at regular intervals, with margins near endpoints and corners to avoid ambiguity from labels, ports, and sharp turns. The dot colors are sampled from a fixed color set: red, orange, yellow, green, blue, violet, pink, and brown. For each queried wire, the ground-truth answer is the ordered sequence of dot colors along that wire, followed by the connected component label and the token "end."

The final dataset contains 50 circuit images and 354 queried wires in total. Each wire connected to a breadboard port is queried once. A response is counted as correct only when it exactly recovers the full ordered color sequence on the queried wire, the correct component label, and the final "end" token. Responses with missing colors, extra colors, incorrect order, an incorrect component label, or missing termination are counted as incorrect.

Prompt. The prompt is designed to force an explicit trace rather than a component-only answer.

Prompt for Circuit Connections

Task: A wire is a series of connected, same colored lines that go from the center of a port, represented on the screen as a white square, to another port. Each wire only connects two ports, one at either end. You must complete the sentence by listing all the colors of the dots on that wire in sequential order, and then give the label of the component connected at the other end. Append 'end' at the very end. No explanations are allowed.

Output format: color1, color2, ..., colorN, {component_label}, end

Visually tracing the wire from port {port_num} on the breadboard, which is the gray rectangle with numbered ports, the sequence of colored dots on its wire is strictly

C.2 Prefix Accuracy Across Circuit Wires

Because the *Circuit Connections* dataset contains wires with different numbers of colored dots, we report cumulative prefix accuracy separately for each group. At each tracing step, a prediction is counted as correct only when all preceding dots have also been recovered correctly. As shown in Figure 12, cumulative accuracy generally declines as tracing proceeds across the different wire groups. This pattern shows that failures accumulate as models attempt to extend the correct sequence to subsequent steps.

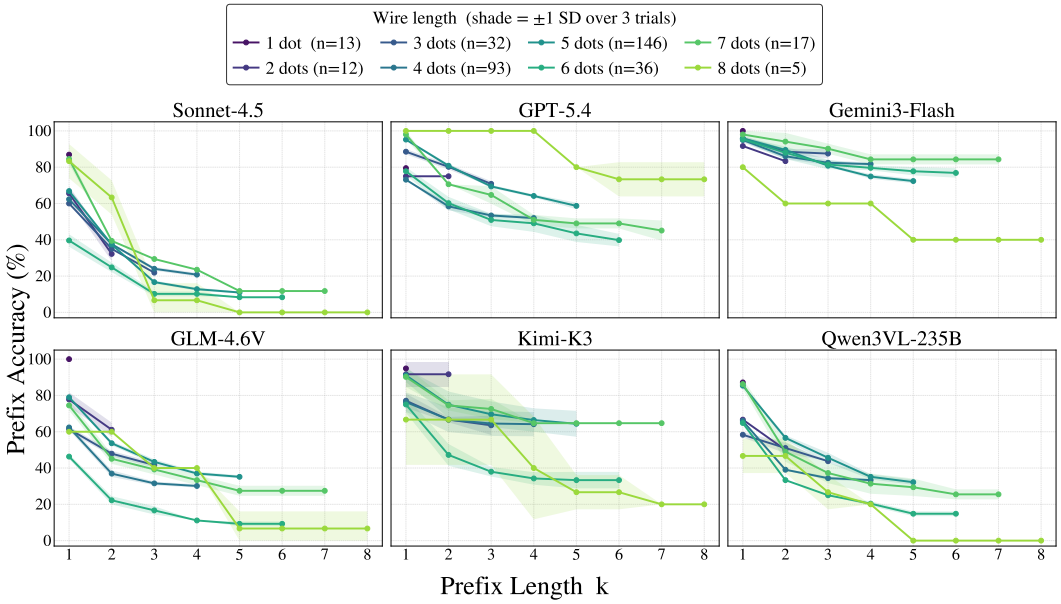


Figure 12: **Cumulative prefix accuracy across wire lengths.** Results are grouped by the number of colored dots on each wire. At each step, accuracy requires the model to have correctly recovered every preceding dot in the sequence. Across wire lengths, accuracy decreases as tracing proceeds, showing the accumulation of path-preservation errors.

C.3 Error-Type Classification for Circuit Connections

To classify incorrect responses, we first compute the longest common prefix between the predicted and ground-truth color sequences. The first position at which the two sequences differ is defined as the divergence point. Errors are then assigned to mutually exclusive categories using a fixed priority order.

We first classify *wrong-start* errors, in which the first predicted dot does not match the ground-truth first dot. Because these responses fail to initialize tracing on the queried wire, their subsequent trajectories cannot be meaningfully interpreted as path-following failures. We therefore identify wrong-

start errors using only the first-dot prediction and apply the coordinate-based divergence analysis only to the remaining responses.

The next category is an *adjacent-wire jump*. At the divergence point, we identify dots with the predicted color that belong to wires other than the queried wire. A response is classified as an adjacent-wire jump when the nearest candidate dot within a Euclidean distance of $\tau = 135$ pixels from the last correctly traced position is selected, and the prediction subsequently follows that wire for at least two consecutive dots in either direction. The threshold corresponds to three grid intervals, where each interval is 45 pixels. We also assign this category when the minimum continuation length is not satisfied but the model predicts the endpoint component of the candidate wire, indicating that it followed the wire to its destination.

The remaining errors are classified as *truncation*, where the response terminates before reporting all dots in the ground-truth sequence; *overshoot*, where additional dots are produced after the full correct sequence; or *others*, which includes all other deviations that do not exhibit a consistent transition pattern.

Table 3: **Error-type distribution on Circuit Connections.** Mean $\pm$ standard deviation of raw error counts over three trials is reported for each mutually exclusive error category. Each trial contains 354 evaluated wires.

Model	Adjacent-wire jump	Wrong start	Truncation	Overshoot	Others
Sonnet-4.5	129.67 ± 2.05	127.67 ± 1.25	12.33 ± 1.25	5.33 ± 0.47	34.67 ± 1.25
GPT-5.4	91.67 ± 2.05	49.67 ± 1.89	5.33 ± 2.49	6.00 ± 0.82	18.00 ± 2.83
Gemini3-Flash	34.67 ± 1.70	16.00 ± 3.27	11.67 ± 3.09	2.67 ± 0.47	17.33 ± 1.25
Qwen3-VL-235B	116.33 ± 0.94	89.67 ± 0.94	8.33 ± 0.47	5.67 ± 0.47	33.33 ± 3.40
Kimi-K3	56.00 ± 2.16	56.33 ± 11.26	7.00 ± 0.82	1.67 ± 0.47	19.00 ± 4.55
GLM-4.6V	110.67 ± 2.05	106.33 ± 2.05	6.67 ± 0.94	11.00 ± 2.16	32.33 ± 3.40

D Swirl

D.1 Swirl Task Construction

We design the *Swirl* task as an abstract tracing benchmark with no semantic objects, background clutter, or path crossings. Each image contains a single spiral path marked with "S" at the start and "E" near the end. The model is instructed to trace the path from "S" and report the colored dots encountered along the path in order. All images in the main evaluation are rendered at a resolution of 640×640 pixels.

Each image contains exactly eight colored dots. To avoid ambiguity from repeated color identities, we use a fixed set of eight colors and assign each color only once within an image: red, orange, yellow, green, blue, purple, pink, and brown. Thus, errors reflect incorrect ordering or loss of the traced path rather than confusion caused by duplicate color labels.

Prompt for Swirl

Task: You must complete the sentence by listing the colors of the dots that satisfy the sentence's logic, in sequential order, and append 'end' at the very end. No explanations are allowed.

Visually tracing the path of the line from 'S' in the image, the sequence of colored dots on its path is strictly

We control density while keeping the path length, line width, number of dots, and image resolution fixed. Rotation density is defined by the spacing between neighboring turns, which is varied by changing the number of turns while preserving the overall path length. Specifically, the number of turns ranges from 1.4–1.8 for *Low*, 1.8–2.2 for *Moderate*, and 2.2–2.7 for *High*. Increasing the number of turns compresses the spacing between neighboring tracks, allowing us to evaluate path preservation as nearby continuations become more closely spaced. For a spiral with T turns, the dots are distributed across $8/T$ angular directions, separated by $360^\circ/(8/T)$, and the entire pattern is randomly rotated for each sample.

For the main evaluation in Section 3.2, we use 100 images per level. For the reasoning analysis in Section 5.2, we use 200 images per level to examine a wider range of generated reasoning traces and failure patterns.

D.2 Human Baseline

We additionally measure a human baseline on the Swirl dataset. Ten human participants each completed 50 images randomly sampled from each rotation density set. Participants traced the spiral from the starting point and entered the eight encountered colors in order by selecting the corresponding color buttons. Each sample was limited to 20 seconds, while the average completion time was 11.3 seconds per sample in *High* density level. Human participants maintained near-perfect prefix accuracy throughout the trace (Table 4), providing a reference for the difficulty of the task under the same sequential tracing requirement.

D.3 Prefix Accuracy

As shown in Table 4, the same cumulative degradation is observed across density levels, but becomes substantially stronger as density increases. At *Low* density, several models maintain high prefix accuracy throughout the trace, whereas *Moderate* and especially *High* density produce much steeper declines. This further supports that path preservation becomes less robust as nearby competing turns are brought closer together.

Table 4: **Cumulative prefix accuracy on Swirl across density levels.** Mean $\pm$ standard deviation (%) is reported at each tracing step. Model results are computed over three trials, while the human baseline is computed over 10 participants. Prefix-k accuracy requires all dots from the first through the k-th dot to be predicted correctly.

Density	Model	Prefix-1	Prefix-2	Prefix-3	Prefix-4
Low	Sonnet-4.5	78.67 $\pm$ 0.94	65.67 $\pm$ 1.70	62.33 $\pm$ 1.25	55.00 $\pm$ 1.63
	GPT-5.4	100.00 $\pm$ 0.00	98.67 $\pm$ 1.25	97.67 $\pm$ 0.94	97.67 $\pm$ 0.94
	Gemini3-Flash	96.00 $\pm$ 0.82	96.00 $\pm$ 0.82	96.00 $\pm$ 0.82	96.00 $\pm$ 0.82
	Qwen3-VL-235B	95.67 $\pm$ 1.89	95.67 $\pm$ 1.89	95.67 $\pm$ 1.89	93.33 $\pm$ 0.94
	Kimi-K3	97.33 $\pm$ 2.05	96.67 $\pm$ 2.36	96.67 $\pm$ 2.36	95.00 $\pm$ 2.45
	GLM-4.6V	99.67 $\pm$ 0.47	96.00 $\pm$ 1.41	96.00 $\pm$ 1.41	96.00 $\pm$ 1.41
	Human	**100.00 $\pm$ 0.00**	**100.00 $\pm$ 0.00**	**100.00 $\pm$ 0.00**	**100.00 $\pm$ 0.00**
Moderate	Sonnet-4.5	67.67 $\pm$ 1.89	52.00 $\pm$ 1.41	35.00 $\pm$ 2.45	18.67 $\pm$ 3.40
	GPT-5.4	99.00 $\pm$ 0.00	88.67 $\pm$ 1.70	87.00 $\pm$ 1.63	85.67 $\pm$ 1.70
	Gemini3-Flash	97.00 $\pm$ 1.63	97.00 $\pm$ 1.63	97.00 $\pm$ 1.63	97.00 $\pm$ 1.63
	Qwen3-VL-235B	90.00 $\pm$ 0.00	84.67 $\pm$ 1.25	77.67 $\pm$ 0.94	67.33 $\pm$ 1.25
	Kimi-K3	90.33 $\pm$ 1.70	88.67 $\pm$ 1.25	83.67 $\pm$ 0.94	77.00 $\pm$ 0.82
	GLM-4.6V	93.00 $\pm$ 1.63	81.67 $\pm$ 1.25	80.00 $\pm$ 1.63	69.00 $\pm$ 0.82
	Human	**100.00 $\pm$ 0.00**	**99.80 $\pm$ 0.60**	**99.80 $\pm$ 0.60**	**99.60 $\pm$ 0.80**
High	Sonnet-4.5	63.33 $\pm$ 1.25	41.33 $\pm$ 3.40	4.00 $\pm$ 0.82	1.00 $\pm$ 0.00
	GPT-5.4	93.33 $\pm$ 0.47	75.00 $\pm$ 0.82	53.67 $\pm$ 3.30	48.00 $\pm$ 2.94
	Gemini3-Flash	98.00 $\pm$ 0.82	97.33 $\pm$ 1.25	92.33 $\pm$ 1.25	90.67 $\pm$ 1.70
	Qwen3-VL-235B	90.00 $\pm$ 2.83	82.33 $\pm$ 3.86	44.33 $\pm$ 3.86	20.33 $\pm$ 3.86
	Kimi-K3	85.67 $\pm$ 1.25	73.33 $\pm$ 0.94	35.33 $\pm$ 2.49	26.67 $\pm$ 2.05
	GLM-4.6V	81.33 $\pm$ 0.94	71.33 $\pm$ 1.70	29.33 $\pm$ 2.05	9.33 $\pm$ 1.25
	Human	**99.80 $\pm$ 0.60**	**99.60 $\pm$ 0.80**	**99.60 $\pm$ 0.80**	**99.20 $\pm$ 1.33**

Table 4: **Cumulative prefix accuracy on Swirl across density levels (continued).**

Density	Model	Prefix-5	Prefix-6	Prefix-7	Prefix-8
Low	Sonnet-4.5	44.00 ± 2.16	43.67 ± 1.70	43.67 ± 1.70	43.67 ± 1.70
	GPT-5.4	94.67 ± 1.25	94.67 ± 1.25	94.67 ± 1.25	94.67 ± 1.25
	Gemini3-Flash	96.00 ± 0.82	96.00 ± 0.82	96.00 ± 0.82	96.00 ± 0.82
	Qwen3-VL-235B	81.33 ± 1.70	81.33 ± 1.70	81.33 ± 1.70	81.33 ± 1.70
	Kimi-K3	91.67 ± 2.87	91.33 ± 2.87	91.33 ± 2.87	91.33 ± 2.87
	GLM-4.6V	87.00 ± 1.41	87.00 ± 1.41	87.00 ± 1.41	87.00 ± 1.41
	Human	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
Moderate	Sonnet-4.5	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	GPT-5.4	69.67 ± 3.40	69.33 ± 3.30	69.33 ± 3.30	69.33 ± 3.30
	Gemini3-Flash	94.33 ± 1.25	92.67 ± 1.25	91.00 ± 0.82	91.00 ± 0.82
	Qwen3-VL-235B	15.33 ± 1.70	14.33 ± 2.36	14.00 ± 2.16	14.00 ± 2.16
	Kimi-K3	50.67 ± 4.64	50.33 ± 4.99	50.33 ± 4.99	50.33 ± 4.99
	GLM-4.6V	17.00 ± 0.82	14.33 ± 0.47	13.33 ± 0.47	13.33 ± 0.47
	Human	**99.40 ± 0.92**	**99.40 ± 0.90**	**99.40 ± 0.90**	**99.40 ± 0.90**
High	Sonnet-4.5	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	GPT-5.4	27.67 ± 2.05	22.00 ± 2.16	20.33 ± 2.05	20.33 ± 2.05
	Gemini3-Flash	70.67 ± 4.64	61.67 ± 5.91	48.33 ± 3.30	48.33 ± 3.30
	Qwen3-VL-235B	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Kimi-K3	8.67 ± 1.25	8.67 ± 1.25	8.67 ± 1.25	8.67 ± 1.25
	GLM-4.6V	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	Human	**99.00 ± 1.34**	**99.00 ± 1.34**	**98.60 ± 1.28**	**98.60 ± 1.28**

D.4 Error-Type Classification for Swirl

For incorrect predictions, we classify each sample by the first point at which the prediction diverges from the ground-truth sequence, so that every error is assigned to exactly one mutually exclusive category. Predictions are first normalized using the same procedure as for accuracy evaluation: text after the final "end" token is removed, color names are mapped to a canonical vocabulary (purple→violet, magenta→pink, cyan→blue, teal→green), only palette-color tokens are retained, and the trailing termination token is removed. A prediction is counted as correct only when the resulting color sequence exactly matches the ground-truth traversal.

We first separate *wrong-start* errors, where the first predicted color already differs from the first ground-truth color. Because no correctly traced point precedes this divergence, an adjacent-turn transition is not defined for these cases. For the remaining errors, we identify the first divergence position $i \geq 1$ and classify the error according to the prediction at that point.

An *adjacent-turn jump* occurs when the substituted dot is the ring-adjacent neighbor of the last correctly traced dot, i.e., a point located one neighboring turn away along the same radial direction. Using the spiral geometry stored in the dataset metadata, this is operationally defined by a Euclidean distance of at most $\tau \cdot \texttt{ring_spacing}$, with $\tau = 1.05$.

The remaining categories are *Truncation*, where the prediction is a strict prefix of the ground-truth sequence without any substitution, and *Overshoot*, where the complete ground-truth sequence is produced before additional colors are appended. After these categories are checked, all remaining failures are assigned to *Others*. Since every image contains all eight palette colors exactly once, every predicted palette color necessarily corresponds to a dot in the image. Counts are aggregated over three independent runs of 100 images per density level and reported either as the average number of errors per 100 samples or as a proportion of all errors at that density level.

Table 5: **First-error type distribution on Swirl across density levels.** Mean ± standard deviation denotes the number of samples per 100 evaluations assigned to each mutually exclusive first-error category over three trials. Each incorrect prediction is classified according to the first point of divergence from the ground-truth trace.

Density	Model	Wrong start	Adjacent-turn jump	Truncation	Overshoot	Others
Low	Sonnet-4.5	21.3 ± 0.94	13.0 ± 0.82	0.0 ± 0.00	0.7 ± 0.94	22.0 ± 1.41
	GPT-5.4	0.0 ± 0.00	2.7 ± 0.47	0.0 ± 0.00	0.0 ± 0.00	2.7 ± 1.25
	Gemini3-Flash	4.0 ± 0.82	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
	Qwen3-VL-235B	4.3 ± 1.89	8.3 ± 0.94	0.0 ± 0.00	0.0 ± 0.00	6.0 ± 0.82
	Kimi-K3	2.3 ± 1.70	3.7 ± 1.89	0.3 ± 0.47	0.0 ± 0.00	2.3 ± 2.05
	GLM-4.6V	0.3 ± 0.47	3.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	9.7 ± 0.94
Moderate	Sonnet-4.5	32.3 ± 1.70	48.0 ± 1.41	0.0 ± 0.00	0.0 ± 0.00	19.7 ± 2.49
	GPT-5.4	1.0 ± 0.00	18.7 ± 3.40	0.0 ± 0.00	0.0 ± 0.00	11.0 ± 3.74
	Gemini3-Flash	3.0 ± 1.63	0.7 ± 0.47	0.0 ± 0.00	0.0 ± 0.00	5.3 ± 1.89
	Qwen3-VL-235B	10.0 ± 0.00	67.7 ± 3.09	0.0 ± 0.00	0.0 ± 0.00	8.3 ± 0.94
	Kimi-K3	9.7 ± 1.70	36.0 ± 5.66	0.0 ± 0.00	0.0 ± 0.00	4.0 ± 1.41
	GLM-4.6V	7.0 ± 1.63	54.7 ± 1.70	0.0 ± 0.00	0.0 ± 0.00	25.0 ± 1.63
High	Sonnet-4.5	36.7 ± 1.25	61.7 ± 2.36	0.0 ± 0.00	0.0 ± 0.00	1.7 ± 1.25
	GPT-5.4	6.7 ± 0.47	53.3 ± 2.05	0.0 ± 0.00	0.0 ± 0.00	19.7 ± 3.30
	Gemini3-Flash	2.0 ± 0.82	3.3 ± 0.47	0.0 ± 0.00	0.0 ± 0.00	46.3 ± 3.86
	Qwen3-VL-235B	10.0 ± 2.83	80.3 ± 3.86	0.7 ± 0.47	0.0 ± 0.00	9.0 ± 1.63
	Kimi-K3	14.3 ± 1.25	60.0 ± 4.32	0.0 ± 0.00	0.0 ± 0.00	17.0 ± 2.16
	GLM-4.6V	18.7 ± 0.94	70.0 ± 0.82	0.0 ± 0.00	0.0 ± 0.00	10.3 ± 0.47

D.5 Run-to-Run Consistency on Swirl

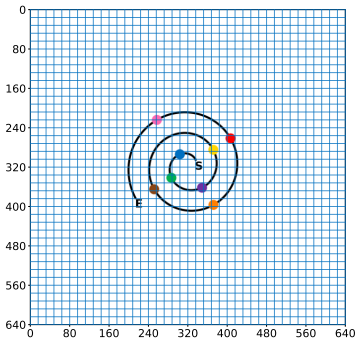
To verify that the observed errors are not merely a product of stochastic sampling, we measured how consistently each model behaves across its three runs, aligning the runs image by image (Table 6). Mean pairwise Cohen's κ on per-image correctness ranges from 0.64 to 0.96, indicating substantial run-to-run agreement. Given the large combinatorial space of possible eight-color output sequences, the repeated occurrence of identical errors is particularly meaningful: when two runs both fail on an image, 41.7–74.3% of the failures produce the same predicted sequence. These results indicate that

the observed failures are largely systematic rather than arising primarily from stochastic variation across runs.

Table 6: **Run-to-run consistency on Swirl.** Statistics are pooled across the Low, Moderate, and High density levels over three runs (300 images per model). κ denotes mean pairwise Cohen's κ on per-image correctness, and *Same wrong sequence* measures how often two jointly incorrect runs produce the same prediction.

Model	κ	Same wrong sequence (%)
GPT-5.4	0.78	46.5
Gemini3-Flash	0.65	56.6
Sonnet-4.5	0.87	59.7
Qwen3-VL-235B	0.89	58.3
GLM-4.6V	0.96	74.3
Kimi-K3	0.64	41.7

D.6 Input Resolution Control



Model	1× Res.	2× Res.	Δ
Sonnet-4.5	0.0%	0.0%	0.0 pp
GPT-5.4	20.3%	22.0%	+1.7 pp
Gemini3-Flash	48.3%	47.0%	−1.3 pp
Kimi-K3	8.7%	16.0%	+7.3 pp
GLM-4.6V	1.0%	4.0%	+3.0 pp
Qwen3-VL-235B	0.0%	0.0%	0.0 pp

Figure 13: **Input-resolution analysis on High-density Swirl. Left:** Patch-grid visualization of a 640×640 Swirl image using the 40×40 patch grid of the Qwen3-VL vision tokenizer. Each colored dot and its surrounding line span approximately a 3×3-patch region, and no patch contains more than one dot. **Right:** Tracing accuracy when increasing the input resolution from 640×640 ($1\times$) to 1280×1280 ($2\times$).

We examine whether insufficient input resolution could explain the tracing failures. As shown in Figure 13 (left) for the *High*-density setting, the original 640×640 images are divided by the Qwen3-VL vision tokenizer into a 40×40 grid of 16×16-pixel patches, yielding 1,600 visual patches. No patch contains more than one dot, ruling out unresolved dot merging at the patch level.

Because the patch sizes and visual tokenization schemes of closed-source frontier models are not publicly available, we additionally conduct a model-agnostic resolution control on the *High*-density setting by doubling both spatial dimensions of the input. Specifically, we repeat the evaluation using 1280×1280 images while preserving the same underlying path, dot placement, and task instructions.

As shown in Figure 13 (right), doubling the input resolution produces only limited and inconsistent changes across models. Sonnet-4.5 and Qwen3-VL-235B remain unchanged at 0.0%, while GPT-5.4 increases from 20.3% to 22.0% (+1.7 pp) and GLM-4.6V from 1.0% to 4.0% (+3.0 pp). Gemini3-Flash slightly decreases from 48.3% to 47.0% (-1.3 pp). Kimi-K3 shows the largest improvement, from 8.7% to 16.0% (+7.3 pp), but its absolute accuracy remains low even at the doubled resolution. Overall, these results suggest that increasing input resolution provides at most limited recovery and that insufficient input resolution is unlikely to be the primary cause of the observed tracing failures.

D.7 Full-Model Scaling on Swirl

We further examine whether increasing overall model size improves visual tracing across the Qwen3-VL family, comparing the 2B, 4B, 8B, 32B, and 235B variants. The vision encoder scales from 24 layers with a 1024-dimensional hidden size in the 2B and 4B models to 27 layers with a 1152-dimensional hidden size in the 8B, 32B, and 235B models. The language-model component also scales substantially: the 2B, 4B, 8B, and 32B variants use 28/2048, 36/2560, 36/4096, and 64/5120 layers/hidden dimensions, respectively, while the 235B-A22B variant uses a 94-layer mixture-of-experts decoder with 128 experts and 8 activated per token. As shown in Figure 14, larger models provide only modest gains, and these advantages diminish as rotation density increases. In particular, performance remains substantially degraded under the more difficult density conditions even at the largest model scale. This indicates that increasing overall model capacity alone does not resolve the underlying path-preservation failure.

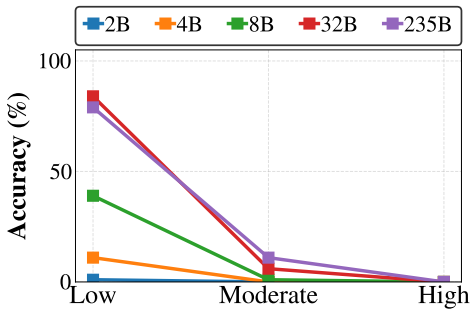


Figure 14: **Qwen3-VL model-size scaling on Swirl task.** Accuracy across rotation-density levels for models from 2B to 235B parameters.

E Path Preservation Under Local Competition Details

E.1 Task Construction

We construct three conditions under otherwise identical visual settings, keeping the queried path fixed for each sample while varying only the number of nearby competing paths (Figure 4). The *Baseline* condition contains a single straight line with four colored dots and an "S" marking the start point. In *Near-1*, one parallel nearby path is added on one side of the queried path. *Near-2* adds a second parallel path on the opposite side at the same distance from the queried path. Across conditions, dot colors are sampled from a fixed eight-color palette, with four non-repeating colors assigned to each path. When colors are reused across paths, they are arranged such that no two paths share the same color at the same tracing step. The first and fourth dot colors are also distinct across paths, allowing the corresponding path to be identified at both the beginning and end of the trace.

E.2 Prefix Accuracy Under nearby-path Perturbations

Across both models, cumulative prefix accuracy declines as nearby paths are introduced (Table 7). The *Baseline* remains near-perfect throughout all four tracing steps, while *Near-1* shows only a modest decline. In contrast, *Near-2* exhibits a sharp cumulative drop: Qwen3-VL-8B decreases from 64.3% at the first dot to 27.7% at the fourth, while InternVL3.5-8B falls from 94.3% to 33.0%. This consistent pattern across two model families shows that path following becomes substantially less reliable when the queried path is surrounded by nearby paths.

Table 7: **Cumulative prefix accuracy under controlled nearby-path perturbations.** Mean ± standard deviation (%) over three runs is reported for Qwen3-VL-8B and InternVL3.5-8B.

Model	Condition	Prefix-1	Prefix-2	Prefix-3	Prefix-4
Qwen3-VL-8B	Baseline	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Near-1	98.00 ± 0.00	97.67 ± 0.47	97.67 ± 0.47	97.67 ± 0.47
	Near-2	64.33 ± 0.47	47.67 ± 0.47	36.67 ± 0.47	27.67 ± 0.47
InternVL3.5-8B	Baseline	99.00 ± 1.00	99.00 ± 1.00	99.00 ± 1.00	99.00 ± 1.00
	Near-1	99.33 ± 0.58	96.67 ± 0.58	91.67 ± 1.53	86.67 ± 2.31
	Near-2	94.33 ± 0.58	77.67 ± 6.03	48.00 ± 3.61	33.00 ± 2.65

E.3 Linear Probing Details

To examine the corresponding internal representations, we extract queried-path dot representations from both the vision encoder and LLM image tokens and train single-layer linear probes for two targets. *Probe A* predicts the color of the probed dot itself, measuring preservation of local visual identity. *Probe B* predicts the color of the next dot along the queried path, measuring whether the representation contains information required for path continuation. The same probing procedure is applied across all three conditions.

Probe training. For each condition and layer independently, we train a multinomial linear probe using

$$\texttt{StandardScaler} \rightarrow \texttt{PCA} \rightarrow \texttt{LogisticRegression},$$

with $C = 1.0$, L2 regularization, the L-BFGS optimizer, and a maximum of 2,000 iterations. Each queried-path dot contributes one training example: the averaged 3×3 patch embedding for the vision encoder or the corresponding merged image token for the LLM. The scaler and PCA are fitted only on the training fold.

Probe Input Construction and Cross-Validation. We conduct the probing analysis using Qwen3-VL-8B [4] and InternVL3.5-8B [44] with 400 images per condition, probing only representations associated with dots on the queried path. For the vision encoder, we extract a 3×3 pre-merge patch region centered on each queried-path dot (Figure 15) at blocks $\{0, 4, 8, 12, 16, 20, 24, 26\}$ for Qwen3-VL-8B (27 blocks) and $\{0, 4, 8, 12, 16, 20, 23\}$ for InternVL3.5-8B (24 blocks), sampling every fourth block and including the final one. Each region covers the full colored dot without overlapping regions assigned to other dots. For the LLM, we probe layers $\{0, 6, 12, 18, 24, 30, 35\}$; both models use a 36-layer Qwen3 language model.

For a queried-path dot q_j at vision block ℓ, the corresponding patch features are

$$H_{i,j}^{(\ell)} \in \mathbb{R}^{9 \times d_v},$$

where d_v is the vision-encoder hidden dimension. We average the nine patch vectors to obtain a single dot representation,

$$x_{i,j}^{(\ell)} = \frac{1}{9} \sum_{p=1}^{9} H_{i,j,p}^{(\ell)} \in \mathbb{R}^{d_v}.$$

For the LLM, each dot is represented by its corresponding merged image token,

$$z_{i,j}^{(k)} \in \mathbb{R}^{d_l},$$

where d_l is the LLM hidden dimension at layer k.

Thus, for each layer and condition, the full probe inputs are

$$X_A \in \mathbb{R}^{1600 \times d}, \qquad X_B \in \mathbb{R}^{1200 \times d},$$

where $d = d_v$ for vision representations and $d = d_l$ for LLM representations. *Probe A* trains all four queried-path dots ($400 \times 4 = 1{,}600$ examples), whereas *Probe B* uses the first three dots because the fourth has no successor ($400 \times 3 = 1{,}200$ examples).

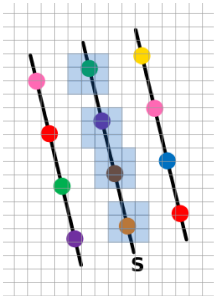


Figure 15: **Patch assignment for linear probing.** Example of the 3×3 patch region used to extract each queried-path dot representation.

We evaluate the probes using five-fold grouped cross-validation with images as groups. The 400 images in each condition are divided into five disjoint folds of 80 images. In each split, one fold is used for testing and the remaining four for training, ensuring that representations from the same image never appear in both sets. This yields 1,280/320 train/test examples for *Probe A* and 960/240 for *Probe B* in each split. The number of examples is matched across conditions, and probe accuracy is averaged over the five test folds.

PCA dimension sensitivity. PCA is applied to reduce the high-dimensional vision and LLM representations before fitting the linear probe, limiting overfitting while retaining information relevant for decoding. We repeat the analysis across PCA dimensions to verify that the observed Probe A and Probe B results are not specific to the dimensionality choice. The main experiments use $n_{\text{PCA}} = 512$.

For *Probe A*, Table 8 (A) reports results at vision block 24 and LLM layer 6, the respective peak-performing locations. Accuracy saturates early and remains similarly high across *Baseline*, *Near-1*, and *Near-2*, indicating that local color identity is preserved regardless of nearby paths. We nevertheless use $n_{\text{PCA}} = 512$ to keep the setup consistent with Probe B.

For *Probe B*, Table 8 (B) reports vision block 24, where vision accuracy peaks, and LLM layer 0, the earliest probed LLM representation. Vision accuracy saturates at 512 dimensions, while LLM accuracy increases only slightly beyond 512. Across informative PCA dimensions, the ordering *Baseline* > *Near-1* > *Near-2* remains stable, showing that the observed degradation is not driven by the dimensionality choice.

Table 8: **Probe accuracy across PCA dimensions.** Held-out linear-probe accuracy across PCA dimensions for **(A)** Probe A and **(B)** Probe B. The main setting, $n_{\text{PCA}} = 512$, is highlighted in bold. Explained variance in Panel B refers to the vision features.

(A) Probe A: Current-dot color prediction

	Vision block 24			LLM layer 6		
n_{PCA}	Baseline	Near-1	Near-2	Baseline	Near-1	Near-2
8	0.930	0.988	0.998	0.151	0.741	0.913
16	1.000	1.000	1.000	0.922	0.994	0.989
32	1.000	1.000	1.000	0.983	0.992	0.989
64	1.000	1.000	1.000	0.984	0.989	0.989
128	1.000	1.000	1.000	0.986	0.989	0.989
256	1.000	1.000	1.000	0.990	0.991	0.990
512	**1.000**	**1.000**	**1.000**	**0.991**	**0.993**	**0.991**
768	1.000	1.000	1.000	0.991	0.993	0.991
Raw (960)	1.000	1.000	1.000	0.991	0.993	0.991

(B) Probe B: Next-dot color prediction

		Vision block 24			LLM layer 0		
n_{PCA}	Explained var.	Baseline	Near-1	Near-2	Baseline	Near-1	Near-2
8	65.4%	0.163	0.142	0.156	0.156	0.156	0.163
16	83.2%	0.302	0.205	0.163	0.214	0.173	0.166
32	91.1%	0.347	0.220	0.183	0.336	0.234	0.175
64	95.5%	0.452	0.368	0.260	0.356	0.245	0.193
128	98.0%	0.485	0.386	0.318	0.370	0.293	0.218
256	99.3%	0.543	0.460	0.355	0.395	0.329	0.240
512	**99.9%**	**0.570**	**0.484**	**0.366**	**0.411**	**0.340**	**0.253**
768	100.0%	0.569	0.493	0.374	0.423	0.345	0.260
Raw (960)	100.0%	0.570	0.493	0.373	0.427	0.343	0.267

Optimization and training loss. We additionally track the training cross-entropy throughout L-BFGS optimization to ensure that differences between conditions are not caused by incomplete probe training. We use an optimization tolerance of 10^{-4}, and allow up to 2,000 iterations for convergence. For each condition, layer, and cross-validation fold, we record the cross-entropy after each accepted optimization step. The plotted loss excludes the L2 penalty and therefore reflects how well the probe fits the training labels directly. The individual fold trajectories and their mean $\pm$ SEM are shown in Figures 17a, 17b, 18a and 18b. Across Probe A and Probe B, the optimization curves reach stable plateaus under this convergence criterion for all three conditions. We therefore compare held-out probe accuracy only after the probes are sufficiently optimized, reducing the possibility that the lower accuracy under *Near-1* or *Near-2* is simply due to slower or incomplete optimization.

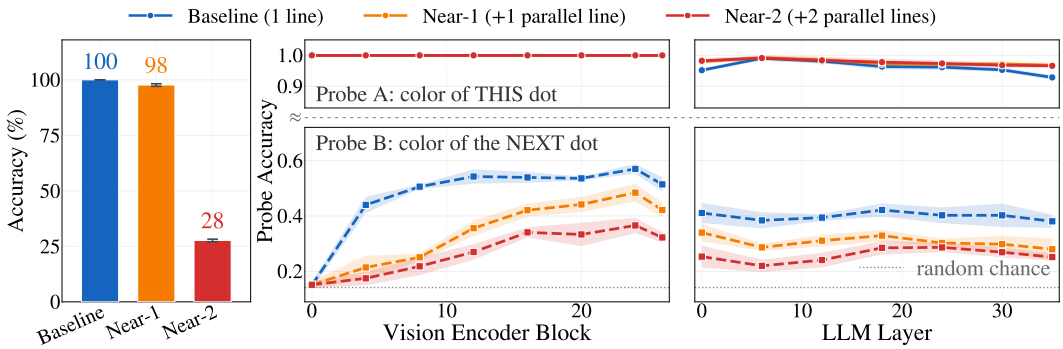
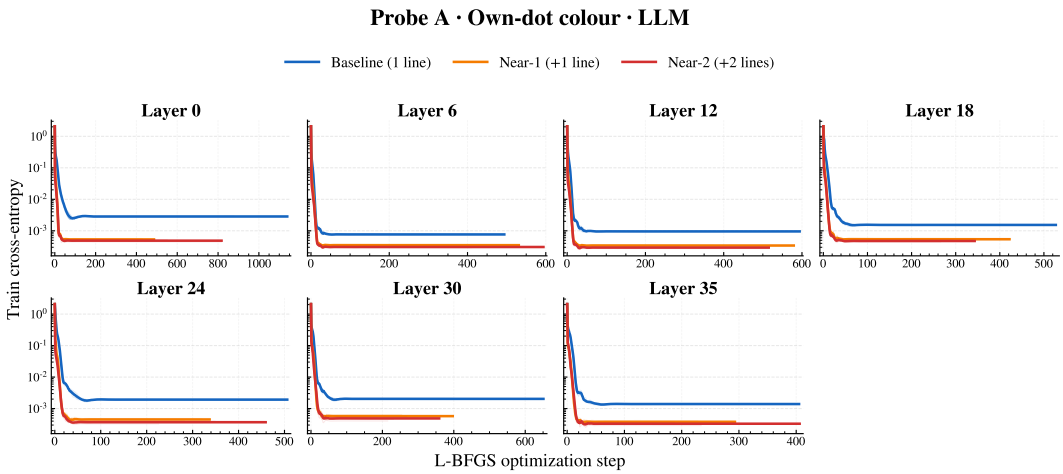


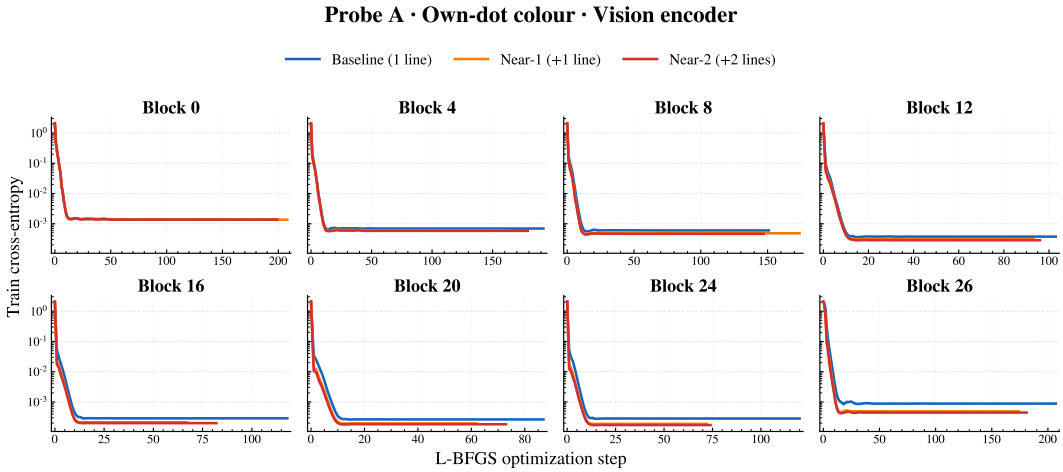
Figure 16: **Task performance and probing accuracy of Qwen3-VL-8B under nearby-path competition.** Left: Task accuracy of Qwen3-VL-8B across the *Baseline*, *Near-1*, and *Near-2* conditions, showing the effect of nearby paths. Right: Linear probing accuracy of Qwen3-VL-8B vision-encoder and LLM representations. *Probe A* measures whether the representation of a dot patch preserves its own color identity, while *Probe B* measures whether it contains information about the next dot along the queried path. The dashed line marks random-choice accuracy $(1/7)$ among the seven remaining palette colors for next-dot prediction.

E.4 Replication with InternVL3.5-8B.

InternVL3.5-8B shows a similar overall trend (Figure 19). Tracing accuracy decreases from 99% in *Baseline* to 87% in *Near-1* and 33% in *Near-2*. Probe A remains high across conditions, while Probe B weakens as nearby paths are introduced, indicating reduced information about which dot comes next along the queried path. This replication suggests that the sensitivity of continuation information to nearby paths is not specific to Qwen3-VL-8B.

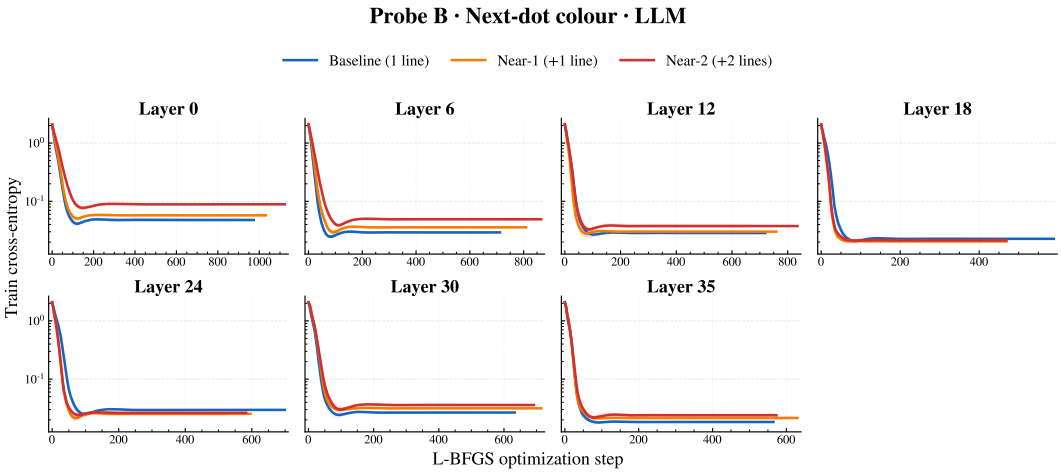


(a) LLM layers.

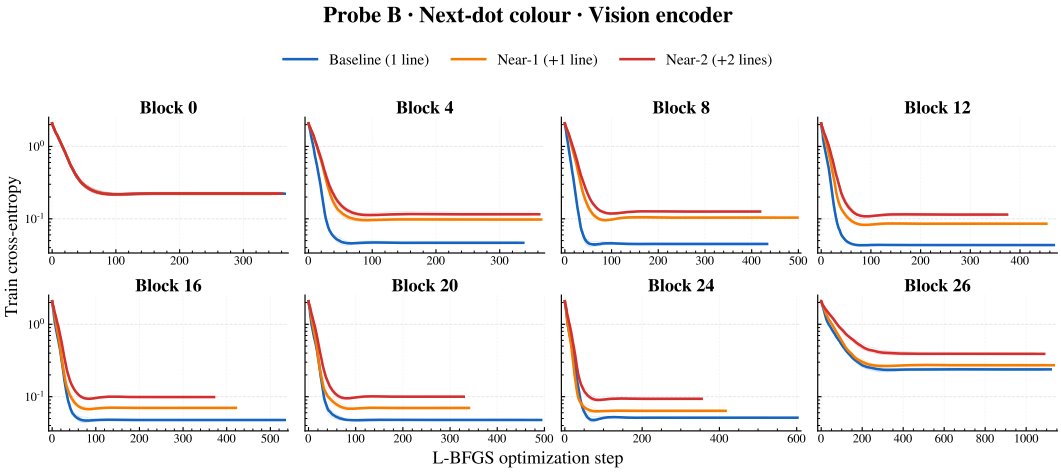


(b) Vision-encoder blocks.

Figure 17: **Probe A optimization across model representations.** Training cross-entropy across L-BFGS optimization steps for *Baseline*, *Near-1*, and *Near-2* in the LLM (top) and vision encoder (bottom).



(a) LLM layers.



(b) Vision-encoder blocks.

Figure 18: **Probe B optimization across model representations.** Training cross-entropy across L-BFGS optimization steps for *Baseline*, *Near-1*, and *Near-2* in the LLM (top) and vision encoder (bottom).

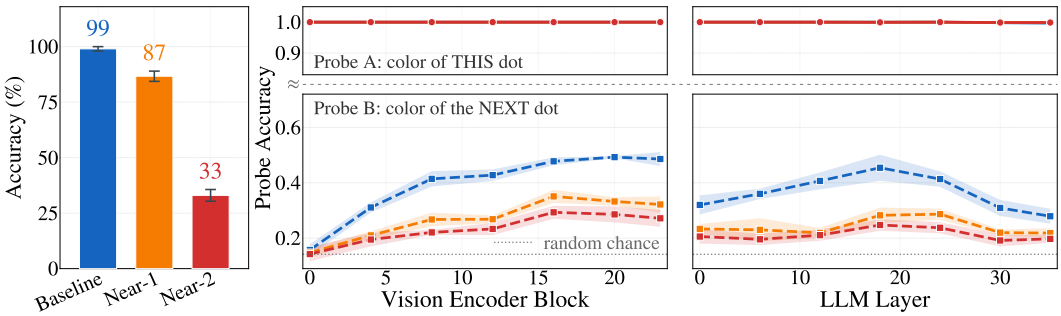


Figure 19: **Task performance and probing results for InternVL3.5-8B under nearby-path competition. (a)** Tracing accuracy across the *Baseline*, *Near-1*, and *Near-2* conditions. **(b)** Linear probing of vision-encoder and LLM representations. Probe A predicts the probed dot's own color, while Probe B predicts the next dot color along the queried path. The dashed baseline indicates random-choice accuracy ($1/7$).

F Control Experiments with Non-Line Visual Elements

To test whether tracing failure is caused simply by nearby visual content or specifically by competing traceable structure, we conduct this control analysis with Qwen3-VL-8B [4], replacing the two parallel lines in *Near-2* with several non-line controls while keeping the queried path unchanged. Representative examples are shown in Figure 20.

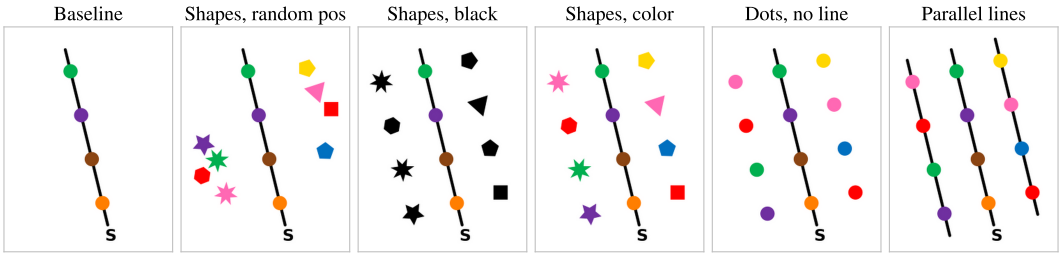


Figure 20: **Representative examples of non-line controls.** The controls vary shape, color, position, and connectivity while keeping the queried path fixed.

Control Construction. We compare six conditions while keeping the queried path unchanged. *Baseline* contains only the queried path with no nearby visual elements. *Shapes, Random Position* places colored shapes at random nearby positions, breaking their aligned arrangement while ensuring that they are not closer to the queried path than the original *Near-2* elements. *Shapes, Black* places black shapes at the aligned *Near-2* positions, while *Shapes, Color* uses randomly selected colored shapes at the same positions. *Dots, No Line* preserves the colors and positions of all eight nearby dots from *Near-2* but removes the connecting lines. Finally, *Parallel Lines* corresponds to the original *Near-2* condition, where the aligned colored dots are connected into two parallel paths. For the shape-based controls, the rendered shape area is matched to that of a colored dot. Across conditions, the queried path, its rotation, and its color sequence remain fixed.

Competing Traceable Structure Drives the Behavioral Failure. As shown in Table 9, nearby non-line visual elements have almost no effect on tracing. Randomly positioned colored shapes preserve 100% prefix accuracy across all four steps, while aligned black shapes, colored shapes, and colored dots without connecting lines remain between 98–100%. In contrast, connecting the aligned colored dots into two continuous parallel lines causes cumulative accuracy to fall from 64.3% at the first dot to 27.7% after all four dots. The non-line controls produce at most a 0–2 pp decline across the trace, whereas *Near-2* already loses 35.7 pp at the first dot and continues to accumulate errors at subsequent steps. These results indicate that nearby visual content, color, or aligned placement alone does not reproduce the failure; substantial path-preservation failure emerges when the nearby structure forms an alternative continuous path that can itself be traced.

Table 9: **Cumulative prefix accuracy across control types.** Accuracy (%) after correctly tracing the first k dots. Values report mean $\pm$ standard deviation across repeated evaluations.

Control Type	$\mathbf{k=1}$	$\mathbf{k=2}$	$\mathbf{k=3}$	$\mathbf{k=4}$
None	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Shapes, Random Position	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Shapes, Black	100.0 ± 0.0	100.0 ± 0.0	99.0 ± 1.0	99.0 ± 1.0
Shapes, Color	99.0 ± 0.0	99.0 ± 0.0	98.0 ± 0.0	98.0 ± 0.0
Dots, No Line	100.0 ± 0.0	99.3 ± 0.6	99.3 ± 0.6	99.3 ± 0.6
Parallel Lines (Near-2)	$\mathbf{64.3 \pm 0.6}$	$\mathbf{47.7 \pm 0.6}$	$\mathbf{36.7 \pm 0.6}$	$\mathbf{27.7 \pm 0.6}$

Path-Continuation Information Is More Sensitive Than Local Dot Identity. The two probes show different sensitivity to nearby visual elements (Figure 21). Probe A remains perfect throughout the vision encoder across all control conditions and above 0.93 in the LLM representations, indicating that the color identity of individual dots remains well represented. Probe B is more sensitive: at vision block 24, accuracy decreases from 0.570 with no nearby element to 0.556 with aligned black

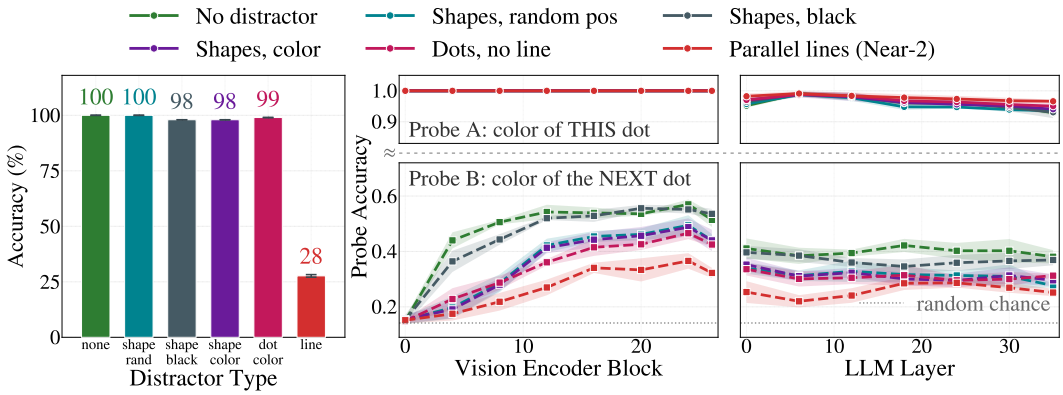


Figure 21: **Behavioral performance and internal signals across control types.** Task accuracy is compared with Probe A, which measures information about each dot's own color, and Probe B, which measures information about the next dot along the queried path. Non-line visual elements largely preserve task performance, whereas the connected parallel-line condition produces substantial behavioral degradation.

shapes, 0.494 with randomly positioned colored shapes, 0.487 with aligned colored shapes, 0.466 with colored dots without connecting lines, and 0.366 with parallel lines. Because Probe B explicitly predicts the next-dot color, some reduction in the presence of additional colored elements is expected and may reflect interference from the verifier colors themselves. Consistent with this interpretation, these colored non-line controls retain near-perfect tracing accuracy despite lower Probe B accuracy.

The key difference appears when the same nearby colored elements are connected into continuous parallel paths: Probe B drops further to 0.366, accompanied by a sharp decline in tracing performance. Thus, the main degradation emerges not from nearby colors alone, but when they form competing traceable paths. Together with our appearance-similarity analysis (Appendix G), these results further show that path preservation becomes increasingly fragile as nearby structures more closely resemble competing continuations.

G Effect of Appearance Similarity on Path Preservation

We further examine how similarity between the paths themselves affects path following under local competition. Specifically, we vary the visual and structural similarity of nearby paths to the queried path and measure how tracing performance changes. We conduct the additional analysis on InternVL3.5-8B [44].

G.1 Similar Nearby Paths Break Path Preservation

As shown in Table 10, accuracy drops sharply in the *Shared segment* condition and is highest in the *Different all* condition. If the model were failing because of a generic inability to read colored dots or follow the task, performance should be similarly poor across conditions. Instead, the substantial recovery as similarity is reduced indicates that path preservation becomes increasingly fragile when nearby paths more closely resemble the queried path.

G.2 Internal States Become Less Selective for the True Continuation

We examine how *attention preference* in the vision encoder and *representational preference* across the vision encoder and LLM decoder change with the appearance similarity of nearby paths. Because the four-dot paths yield fewer valid next-step samples and make the main probing setup in the previous section less directly comparable, we use these preference-based measures instead. If tracing is robust, both attention and representations should remain selective for the true continuation across similarity conditions.

Table 10: **Accuracy across local similarity conditions.** Tracing accuracy of Qwen3-VL-8B and InternVL3.5-8B under varying appearance similarity between nearby paths and the queried path. Results are reported as mean ± standard deviation over three runs.

Condition	Qwen3-VL-8B	InternVL3.5-8B
Shared segment	76.67 ± 0.76	50.50 ± 2.50
Different segment	95.17 ± 0.29	80.50 ± 1.32
Different angle	94.67 ± 0.29	65.50 ± 2.29
Different all	99.67 ± 0.29	86.83 ± 2.08

Attention Preference Extraction. We compute attention preference using Qwen3-VL-8B and InternVL3.5-8B by mapping the queried-path and nearby-path dot coordinates to their corresponding vision-token regions. Each dot is represented by a 3×3 pre-merge patch neighborhood centered on its coordinate to reduce sensitivity to patch-alignment errors. Specifically, we examine the transition from q_2 to q_3 on the queried path, where local appearance similarity is controlled across conditions. We compare attention from q_2 to the correct continuation q_3 against attention from q_2 to the corresponding second-position dots on the two nearby paths, $d_{1,2}$ and $d_{2,2}$. At each vision-encoder block, we average attention over heads and source tokens.

Representational Preference Extraction. For the representation analysis, we register hooks on the output of each vision-encoder block and each LLM decoder layer. For each dot region, we average the hidden states over its 3×3 image-token neighborhood to obtain one region representation. Before computing cosine similarity, hidden states are mean-centered within the layer. We then compute the cosine similarity between the queried-path representation at q_2 and its true continuation q_3, and compare it with the larger similarity between q_2 and the corresponding second dots on the two nearby paths, $d_{1,2}$ and $d_{2,2}$.

InternVL3.5-8B Result. We observe the same overall pattern in InternVL3.5-8B. Tracing accuracy decreases as nearby paths become more similar to the queried path, with the lowest performance in the *Shared segment* condition and higher accuracy as local similarity is reduced (Table 10). The internal analysis shows a corresponding trend: attention preference for the true continuation is lowest when nearby paths share the strongest local similarity with the queried path and increases as this similarity is reduced, while representational preference likewise decreases under greater nearby-path similarity across the vision encoder and LLM decoder (Figure 22). These results replicate the same sensitivity to nearby-path appearance observed in Qwen3-VL-8B, indicating that the effect is not specific to a single model.

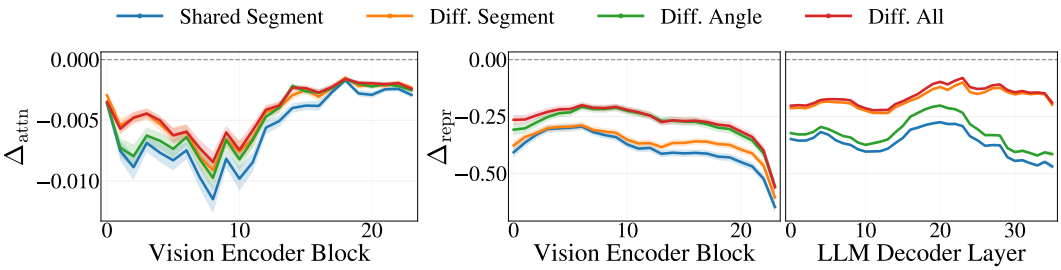


Figure 22: **Internal preference in InternVL3.5-8B under varying appearance similarity.** Left: Block-wise attention preference (Δ_{attn}) in the vision encoder, comparing attention from the second queried-path dot to the correct third dot against attention to the corresponding second-position dots on nearby paths. Right: Representational preference (Δ_{repr}) across the vision encoder and LLM decoder, comparing similarity to the true continuation against similarity to the nearby-path alternatives. Positive values indicate greater preference for the correct continuation.

G.3 Additional Masking Intervention Results

We perform a masking intervention to test whether locally similar nearby-path cues directly contribute to tracing failures across all similarity conditions. For each intervention, we mask a 3×3 image-token patch centered on the specified target: the corresponding dot on the nearby path, the queried-path dot, or a randomly selected location. Random masks are sampled only from object-free regions, excluding a 5×5 area around every dot center to avoid overlap with task-relevant elements. We use the same prompt as in the Swirl task construction (Appendix D.1) and evaluate performance using exact-match scoring.

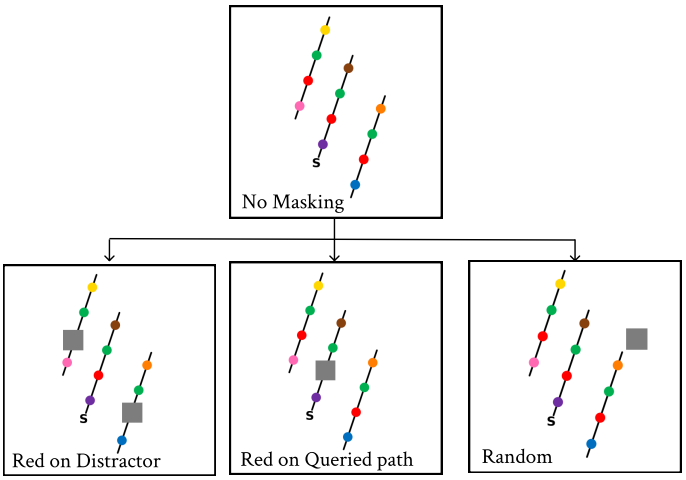


Figure 23: **Causal masking intervention.** Example shown for the *Shared segment* condition. Masking is applied around the local transition where the model must preserve the queried path against a nearby competing continuation. We compare masking the corresponding dot on the nearby path, the queried-path dot, and a random image region.

Causal Masking Confirms Nearby-Path Interference. The intervention setup is illustrated in Figure 23, and the full results across all conditions are reported in Table 11. Across all conditions, masking a random patch has little effect, while masking the queried-path region sharply reduces accuracy, confirming that this region is necessary for correct tracing. In contrast, masking the corresponding nearby-path region improves performance, with the largest recovery in the *Shared segment* condition, where accuracy increases from 76.5% to 90.0% (+13.5 points). The gain becomes smaller as local similarity is reduced and is negligible in *Different all*, providing causal evidence that locally similar nearby-path cues interfere most strongly with preservation of the queried path.

Table 11: **Patch masking accuracy across all path conditions.** All results use a 3×3 masking patch. Values in parentheses indicate the change relative to the *No Masking* baseline within the same condition.

Masking Setup	Shared Segment	Different Segment	Different Angle	Different All
No Masking	76.5	95.0	94.5	99.5
Nearby-Path Dot	90.0 (**+13.5**)	98.0 (+3.0)	99.0 (+4.5)	99.5 (+0.0)
Queried-Path Dot	2.0 (-74.5)	0.5 (-94.5)	0.5 (-94.0)	0.0 (-99.5)
Random	76.5 (+0.0)	94.5 (-0.5)	94.0 (-0.5)	99.0 (-0.5)

H Additional Analysis of Reasoning Traces

H.1 Self-Correction Frequency Estimation

We estimate self-correction frequency from the generated reasoning traces rather than from the final answers alone. Because the reasoning outputs differ substantially across models, we use model-specific textual markers that indicate revision, rechecking, or reconsideration. For Sonnet-4.5, GPT-5.4, and Gemini3-Flash, which do not expose the full reasoning trace, we apply the same analysis to their provided reasoning summaries. The goal is to quantify self-correction frequency by detecting explicit markers that the model revises or re-evaluates its current solution.

Table 12: **Self-correction markers used to estimate self-correction frequency.**

Model	Self-correction markers
GPT-5.4	*wait*, *actually*, *revising*, *rechecking*, *reconsidering*
Sonnet-4.5	*wait*, *actually*, *let me check*, *let me recheck*, *re-examine*
Gemini3-Flash	Bold revision headers containing correction-related terms, including *revise/revising*, *revisit*, *re-examine*, *reconsider*, *recheck*, *correct*, *fix*, *adjust*, *amend*, *backtrack*, *wrong*, *error*, or *mistake*
Qwen3-VL-235B	*wait*, *actually*, *let me check*, *let me recheck*, *re-examine*, and *reconsider*
GLM-4.6V	*wait*, *let's check*, *let me confirm*, and *let's look again*
Kimi-K3	*wait*, *actually*, *hmm*, *need verify*, *need check*, and *need re-check*

For model m at rotation density level ℓ, we define *self-correction frequency* as the average number of detected self-correction markers per reasoning trace:

$$F(m,\ell) = \frac{1}{\left|\mathcal{D}_\ell^{(m)}\right|} \sum_{i \in \mathcal{D}_\ell^{(m)}} c\left(t_i^{(m)}\right), \tag{1}$$

where $\mathcal{D}_\ell^{(m)}$ denotes the set of evaluated samples for model m at density level ℓ, $t_i^{(m)}$ is the corresponding reasoning trace for sample i, and $c(t_i^{(m)})$ denotes the number of detected self-correction markers in that trace. Higher self-correction frequency therefore indicates more frequent revisions or reconsiderations during reasoning.

H.2 Keyword-Based Analysis of Reasoning-Style Shifts

To quantify how visible reasoning patterns change with rotation density, we use the brief sequential tracing observed at *Low* density as a reference and identify lexical cues associated with alternative, non-tracing strategies. Each reasoning trace is matched against a fixed set of keyword and regular-expression patterns grouped into three categories: angular matching, ring decomposition, and radial estimation. *Angular matching* refers to reasoning that orders dots using their angular positions rather than visible line connectivity. *Ring decomposition* refers to partitioning the spiral into inner, middle, or outer loops and reasoning over these groups separately. *Radial estimation* refers to using estimated coordinates or distances from the spiral center to determine the dot sequence. These categories were manually defined by the authors as a coarse quantitative proxy for changes in visible reasoning style, rather than an exhaustive or absolute classification of model reasoning. The complete mapping is provided in Table 13. We count all matched occurrences within each trace and report their average frequency per sample at *Low*, *Moderate*, and *High* density.

As shown in Figure 24, non-tracing reasoning patterns generally become more prominent as rotation density increases. With the exception of Sonnet-4.5, all models show higher keyword-match frequency at higher density, indicating more frequent use of global geometric constructs such as rings, angles, radii, and coordinates. Gemini3-Flash exhibits substantially lower overall intensity

Table 13: **Keyword mappings for non-tracing reasoning patterns.** Exact lexical and regular-expression patterns used to count each reasoning category.

Category	Keywords / patterns used for counting
Angular matching	Numerical angle expressions ending in `°`, `degree`, or `degrees`; `top-right`, `top-left`, `bottom-right`, `bottom-left` (and their space-separated forms).
Ring decomposition	`inner ring`, `inner loop`, `inner circle`, `inner spiral`, `outer ring`, `outer loop`, `outer circle`, `outer spiral`, `innermost`, `outermost`.
Radial estimation	Pixel-coordinate expressions of the form `(x, y)`; `radius`, `radii`, and expressions of the form `r = [number]`.

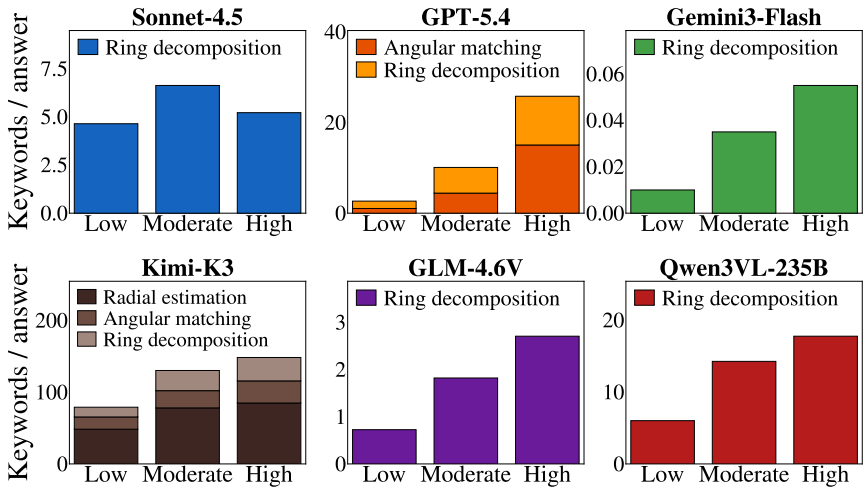


Figure 24: **Non-tracing reasoning-pattern intensity across Swirl density.** Average keyword-match frequency for non-tracing reasoning patterns at *Low*, *Moderate*, and *High* rotation density. Higher values indicate more frequent lexical cues associated with global geometric reasoning.

than the other models, likely because it provides only a limited reasoning summary rather than the full reasoning trace, reducing the amount of detectable strategy-related vocabulary. Because this analysis relies on keyword-based matching, it should be interpreted only as an indirect indicator of trends in visible reasoning style rather than an absolute measure of strategy use or internal reasoning.

H.3 Additional Reasoning Results for Qwen3-VL-235B and GLM-4.6V

We further examine Qwen3-VL-235B and GLM-4.6V, which are omitted from the main-text reasoning analysis for brevity. As shown in Figure 25, neither model exhibits a substantial improvement in tracing performance when reasoning is enabled. These results provide additional evidence that extended reasoning does not consistently compensate for the lack of robustness in visual tracing.

H.4 Reasoning Trace Examples

We provide qualitative examples of how reasoning changes with rotation density across models. At *Low* density, models generally describe the path through brief sequential tracing, proceeding from one connected dot to the next; representative examples for each model are shown in Table 14. At *Moderate* and *High* density, however, reasoning increasingly includes alternative strategies that depart from this sequential procedure, such as ring decomposition, angular or directional matching, coordinate estimation, and repeated direction reconsideration. Representative examples of these density-dependent reasoning patterns are provided in Table 15.

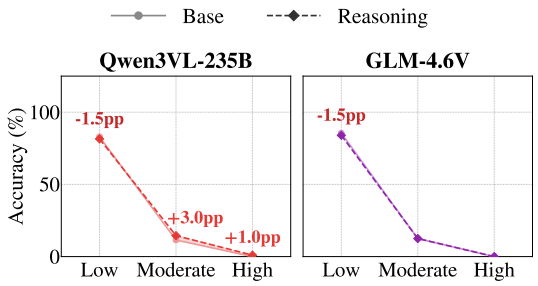


Figure 25: **Additional reasoning results for Qwen3-VL-235B and GLM-4.6V.** Accuracy of *Base* and *Reasoning* models on Swirl across rotation-density levels.

Table 14: **Representative reasoning traces at the Low difficulty level.**

Model	**Reasoning Type**	**Generated Reasoning**
Sonnet-4.5	Brief spatial tracing	"Let me **trace the path from 'S'** in the image. Starting at 'S', I need to **follow the line** and identify the colored dots **in sequence**. Looking at the image: S is at a purple dot; **following the line from S**, it goes to a yellow dot, then to an orange dot, then to a green dot, then to a red dot, then to a blue dot, then to a brown dot, then to a pink dot, **then to E**."
GPT-5.4	Brief spatial tracing	"**Listing colors in order.** I'll note the colors **directly as I trace** the spiral path, **starting from 'S' and going outward**. The colors I find in order are: red, green, pink, purple, orange, yellow, brown, blue, and then 'E' to wrap it up with "end" at the finish."
Gemini3-Flash	Brief spatial tracing	"**Mapping the Sequence.** I've pinpointed the 'S' and, as anticipated, it's **directly adjacent to an orange dot**. Now, the tracing begins! I'm ready to **follow the line, dot by dot**, from the orange starting point. **Defining the Path.** I have finalized the color sequence: orange, red, yellow, blue, purple, green, brown, pink, and end. The spiral from 'S' to 'E' is now **completely mapped out**."
Qwen3-VL-235B	Brief spatial tracing	"The path starts at S and **follows the line**. **Let's trace it.** First, **from S**, the line goes to green, then brown, then blue, then purple, then orange, then red, then pink, then yellow, **and then to E**. ... Starting at S, the first dot is green, then brown, blue, purple, orange, red, pink, yellow."
Kimi-K3	Brief spatial tracing	"**Need inspect image.** ... Image shows black line forming spiral from S at inner near right of small loop. Dots colored along path: near S **inner** yellow at top of **inner arc**, orange left **inner**, green lower-left **inner**, blue bottom **outer**, purple right **outer** ..."
GLM-4.6V	Brief spatial tracing	"The path starts at S, then **we trace the line**. Let's list the dots **in order**. **Starting from S**, the first dot is orange, then yellow, then pink, then green, then red, then brown, then blue, then purple. ... So the sequence is orange, yellow, pink, green, red, brown, blue, purple."

Table 15: **Representative reasoning traces exhibiting global-heuristic substitution.** Examples illustrate global-heuristic reasoning patterns observed across models. Gemini3-Flash does not expose full reasoning traces, making such patterns less directly observable; its excerpt should therefore be viewed as illustrative rather than fully representative.

Model	Substitution Type	Generated Reasoning
Sonnet-4.5	Ring decomposition	"There are **two circles**: an **outer one** and an **inner one**. S is marked on the inner circle near the purple dot. Tracing from S along the line: **purple** → **red** → **pink** → **blue** → **back to purple**. Since it is a **circular path**, I should list the sequence of dots until returning to the starting point, then append 'end'."
GPT-5.4	Ring decomposition	"**I need to document the order of the colored dots:** red, purple, orange, and yellow **on the inner loop**; then brown, pink, blue, and green **on the outer loop**." The two per-ring lists are concatenated into the final answer.
	Angular matching	"I am considering loops that start with the inner section at purple (**southeast**) and progress to red (**northeast**), then pink (**northwest**), yellow (**southwest**), orange (**southeast**), and blue. … This arrangement follows a counterclockwise spiral and increases in radius, with points at every **quadrant** roughly every 90 degrees as it turns."
Gemini3-Flash	Ring decomposition	"There are **two loops**. The **inner loop** holds orange, brown, purple, and blue, while the **outer loop** holds pink, yellow, green, and red. … So the sequence is orange, brown, purple, blue, then pink, yellow, green, red."
Qwen3-VL-235B	Ring decomposition	"**Inner circle (smallest loop)**: purple (top-left inner), yellow (bottom inner), brown (right inner) — **Middle circle (next loop)**: red (left middle), blue (bottom middle), green (right middle) — **Outer circle (E loop)**: orange (top-right outer), pink (top outer)."
Kimi-K3	Coordinate estimation	"Dots along it at **increasing radii and angles**. **If we list by radius from center (~325,330)**: yellow **(301,338) r~25**; green **(347,356) r~34**; purple **(365,295) r~52**; orange **(289,270) r~68**; red **(375,395) r~82**; pink **(408,266) r~105**; blue **(263,230) r~117**; brown **(220,391) r~122**. **Radii not monotonic perfectly due center estimate**. Sequence by angle maybe."
	Angular matching	"This order (yellow, red, orange, pink, green, brown, blue, purple) is **not monotonic angle around center** maybe. Let's **compute angles relative to center S (~300,320)**: yellow angle **~90**° (down), red **~135**° (down-right) … orange (39,–67) angle **–60**°, pink (–46,–42) angle **–137**° or 223, green (–50,67) angle **127**°, brown (67,105) angle **57**°, blue (55,–132) angle **–67**°, purple (–107,–74) angle **–145**°. **Not simple**."
	Ring decomposition	"Two black curved paths: **inner small nearly circular arc** around S, and **outer larger arc/circle** with E at upper left. Colored dots placed on paths: **Inner circle dots**: brown at upper-left, purple at upper-right, red at right-lower, orange at bottom near S. **Outer large circle**: yellow at top-right, pink at right, blue at bottom, green at left-lower? Hmm green dot at lower-left **may lie on outer circle**? … It might be on outer circle, **not inner**."
GLM-4.6V	Ring decomposition	"Wait, the spiral: the **innermost** is S, then the **first loop** has orange, blue, pink, yellow? So the first dot after S is orange, then blue, then pink, then yellow (that's **the inner loop**), then **the next loop**: purple, green, brown, red?"

I Prompting Intervention Details

Prompt Construction. We compare three prompting conditions. *Standard* uses the original tracing prompt without procedural guidance. *Human-instructed* adds a tracing procedure explicitly specified by a human, instructing the model to follow visible line connectivity from S while rejecting choices based on proximity or visual similarity. For *Model-generated*, we provide each model with the same representative *Low*-density Swirl image and ask it not to solve the image, but instead to generate three reusable and meaningfully distinct tracing procedures that could be applied to other instances of the task. Each procedure must consist of explicit numbered steps describing how to trace from S, without assuming any particular color sequence, path shape, number of dots, or answer from the example image. The task statement and final answer prefix are held fixed across all versions, while only the generated instruction and hypothetical example may vary. Table 17 reports only the instruction component of all three versions generated by each model.

Prompt for Generating Model-Specific Tracing Instructions

You are asked to write explicit instructions for solving a visual line-tracing task.
Your goal is **NOT** to solve the image shown here. Instead, generate three reusable tracing instructions that you yourself would find effective for reliably solving this type of task.
For each version, the instruction should explicitly describe, in sequential order, the visual tracing procedure that should be followed from the starting point "S" along the target line. Each numbered step should describe one concrete action or decision that can be directly followed when solving a new image.
Do not assume any particular color sequence, path shape, number of dots, or answer from the current image. The instructions must generalize to other images of the same task.
Each version must include a simple hypothetical example demonstrating how its tracing procedure should be applied. The example must not use or reveal the solution to the current image.
Generate three meaningfully distinct tracing procedures. Each version should represent a different approach to reliably tracing the target line rather than a superficial rewording of the same procedure.
Keep each instruction concise, but include every procedural step that you consider necessary for reliable tracing.

For all three versions:

- Copy the Task sentence exactly as provided below.
- Copy the final sentence beginning with "Visually tracing..." exactly as provided below.
- Do not modify, paraphrase, or omit either of these two fixed components.
- Only the Instruction and Example should differ across versions.
- The instruction must use numbered sequential steps.
- Use as many numbered steps as you consider necessary; the number of steps is not fixed.
- Do not include any text outside the three specified blocks.
- Do not use code fences.

Return exactly three blocks in the following structure:

<version_1>
Task: Complete the sentence by listing the colors of the dots on the target line in sequential order, followed by "end". No explanations are allowed.
Instruction:
1. [First procedural step]
2. [Next procedural step]
3. [Continue with additional numbered steps as necessary]
[· · ·]
Example:
[Give one simple hypothetical example demonstrating how this tracing procedure should be applied. The example must not use or reveal the solution to the current image.]
Visually tracing the path of the line from "S" in the image, the sequence of colored dots on its path is strictly
</version_1>
...
</version_3>

Table 16: **Human-instructed tracing procedure.** The tracing procedure was explicitly specified by a human and provided identically to all evaluated models.

Step	Instruction
1	Start from the point labeled "S" and treat it as the current position.
2	Select the dot that is visibly connected to the current position by a continuous line; use this line connection as the sole criterion for selection.
3	Do not select a dot based on proximity or visual similarity to the path. For example, do not select a dot simply because it is close to the current position, or because it lies on a nearby parallel segment.
4	Move to the selected dot, treat it as the new current position, and repeat until the path ends.

Table 17: **Model-generated tracing instructions.** Three reusable tracing procedures generated independently by each model. Only the generated *Instruction* component is shown; the fixed task statement, hypothetical example, and final answer prefix are omitted.

Model	Ver.	Generated Instruction
Sonnet-4.5	V1	1. Locate the starting point labeled 'S' in the image. 2. Identify the line segment that begins directly at or connects to 'S'. 3. Follow this line continuously with your eyes in the direction it extends from 'S'. 4. Each time the line passes through or touches a colored dot, record that dot's color in sequence. 5. Continue tracing the line without deviation until it terminates (reaches an endpoint, a label like 'E', or closes back on itself). 6. List all recorded colors in the order encountered, followed by 'end'.
	V2	1. Locate the starting point 'S' and identify which line segment is connected to it. 2. From 'S', move along the connected line to the first colored dot you encounter on that line. 3. Record the color of this dot. 4. At each dot, verify that the line continues beyond it by identifying the line segment exiting the dot. 5. Follow the exiting line segment to the next colored dot on the same continuous path. 6. Repeat steps 3–5, ensuring you stay on the same continuous line (ignore any other lines that may be nearby but not connected). 7. Stop when the line reaches a termination point or label (such as 'E') with no further continuation. 8. Output the list of colors in the sequence recorded, followed by 'end'.
	V3	1. Find the point marked 'S' in the image. 2. Observe the direction of the line emanating from 'S' and identify the next dot that lies directly on this line path. 3. Record the color of this next dot. 4. Position yourself mentally at this dot and determine which direction the line continues from here. 5. Look ahead along the continuing line to find the next dot on the path. 6. Record this new dot's color and repeat steps 4–5. 7. Continue this forward progression until no more dots are encountered on the line or the line reaches a terminal point. 8. State the complete sequence of colors in order, concluding with 'end'.

Table 17: **Model-generated tracing instructions (continued).**

Model	Ver.	Generated Instruction
GPT-5.4	V1	1. Locate the letter 'S' and identify the single line that touches it. 2. Start at 'S' and trace that exact line continuously in the direction it extends, without switching to any nearby path. 3. Each time the traced line passes through a colored dot, write down that dot's color immediately. 4. At curves, loops, or close approaches, stay on the same unbroken stroke; ignore dots on lines that are adjacent but not connected to your traced path. 5. Continue tracing until the line clearly terminates. 6. Output the recorded colors in the order encountered, then add end.
	V2	1. Find 'S' and note the first visible direction the target line leaves it. 2. Follow the line only a short segment at a time, always asking which stroke is the direct continuation of the one you were just on. 3. When you reach a colored dot that lies on that stroke, add its color to your list. 4. After each dot, resume tracing from just beyond that dot along the same line. 5. If the path passes near another part of the drawing, ignore proximity and follow continuity of the stroke itself. 6. Repeat until no more line remains. 7. Give only the color sequence you collected, followed by end.
	V3	1. Begin at 'S' and mentally mark your current position on the line. 2. Trace forward until you hit the next colored dot on that same line; record its color as the next checkpoint. 3. From that dot, continue to the next dot on the same connected path and record it. 4. Keep using each found dot as the new checkpoint so you never skip ahead or jump across gaps. 5. If the line bends around or passes close to earlier sections, confirm that the path remains physically connected before recording any dot. 6. Stop when the checkpoint-to-checkpoint tracing reaches the line's endpoint. 7. Return the recorded colors in order, followed by end.

Table 17: **Model-generated tracing instructions (continued).**

Model	Ver.	Generated Instruction
Gemini3-Flash	V1	1. Locate the 'S' marker which indicates the start of the line. 2. Identify the specific line segment that is physically connected to the 'S' marker. 3. Move your focus along this line away from the start point. 4. When you encounter a colored dot directly on the line, note its color. 5. Continue following the same line path, ensuring you do not cross over to adjacent or intersecting lines. 6. Record the color of every dot in the exact order you pass them until you reach the 'E' marker.
	V2	1. Position your gaze at the 'S' symbol at the center of the image. 2. Follow the continuous path of the line as it spirals outward from the center. 3. Identify every colored circle that sits directly on the line you are tracing. 4. Keep track of the colors in a list according to their distance from the 'S' marker along the path. 5. Be careful at points where the line loops close to itself; only record dots that are on the specific path originating from 'S'. 6. Stop the process once the path reaches the 'E' terminal marker.
	V3	1. Identify the 'S' start point and the 'E' end point on the diagram. 2. Trace the single, unbroken line that connects these two points, starting your journey at 'S'. 3. Scan forward along the line and list the color of the first dot you pass. 4. Continue scanning forward along the line's specific trajectory, listing each new dot you encounter. 5. Ensure you do not skip any dots or include dots from other parts of the spiral. 6. Once the path ends at the 'E' marker, finalize your list.

Table 17: **Model-generated tracing instructions (continued).**

Model	Ver.	Generated Instruction
Kimi-K3	V1	1. Locate 'S' and identify the single visible line segment that leaves it; ignore all other labels and nearby dots. 2. Place an imaginary tracer at 'S' and move only along the center of that stroke in the direction the stroke leaves 'S'. 3. Continue smoothly along curves, keeping the tracer on the same continuous stroke and never jumping across blank space. 4. Whenever the stroke passes through or touches a dot, append that dot's color to the sequence in the order encountered. 5. At any crossing or junction, choose the branch with the smallest change in direction that keeps the same stroke width/style; do not turn onto a merely touching or crossing line. 6. If a chosen branch dead-ends, backtrack to the last junction and take the next smoothest untried continuation. 7. Stop when the stroke ends, reaches a labeled endpoint such as 'E', returns to the start with no new dots, or no untraced continuation remains. 8. Output the recorded colors separated by spaces, then 'end'.
	V2	1. Mark 'S' as the current position and estimate the outgoing tangent of the target line at 'S'. 2. Advance a short fixed step along that tangent; after each step, re-estimate the local tangent from the most recent stroke direction. 3. Use a look-ahead window: before crossing any dot or junction, confirm the next piece of stroke continues the same tangent, thickness, and spacing pattern. 4. Treat a dot as on-path only if the stroke center intersects the dot or clearly connects to its edge along the current direction; record its color once. 5. At a junction, rank candidate exits by tangent alignment and continuity, try the best-ranked exit, and mentally mark it as used. 6. If the trail becomes ambiguous, compare the immediate backward direction and reject any candidate that requires a sharp reversal unless the visible stroke truly makes that U-turn. 7. Repeat step-advance, dot recording, and junction ranking until an endpoint label, open end, completed loop, or exhausted stroke is reached. 8. Read the recorded list from first to last and finish with 'end'.
	V3	1. Convert the picture mentally into a graph: nodes are 'S', endpoints/labels, junctions, and dots; edges are stroke pieces between adjacent nodes. 2. Keep only edges belonging to the target stroke connected to 'S'; discard isolated decorations, labels, and crossing strokes that do not share a continuous junction. 3. Starting at the 'S' node, choose the incident edge that is the actual outgoing stroke and mark it visited. 4. Traverse from node to node across unvisited connected edges; when a node is a dot, append its color to the sequence. 5. At a junction node with multiple unvisited edges, select the edge that continues the same stroke with the least curvature change; mark rejected edges so they are not revisited unless backtracking is required. 6. If traversal reaches a dead end with unvisited edges elsewhere, backtrack along visited edges to the nearest junction with an untried plausible edge and resume. 7. For loops, traverse the cycle once from the entry direction and do not recount dots when the path closes or repeats a node. 8. Stop after all reachable target-stroke edges from 'S' have been considered or an explicit end label is reached; then report colors in visit order followed by 'end'.

Table 17: **Model-generated tracing instructions (continued).**

Model	Ver.	Generated Instruction
GLM-4.6V	V1	1. Locate the starting point marked 'S' in the image. 2. Begin tracing the line starting from 'S', moving in the direction the line extends from this point. 3. As you trace, identify and record the color of each dot you encounter in the exact order they appear along the line. 4. Continue tracing until you reach the end of the visible line path. 5. After tracing the complete path, list the recorded colors in sequence, followed by 'end'.
	V2	1. Identify the starting point 'S' and the initial direction of the line extending from it. 2. Trace the first segment of the line from 'S' until you reach the first colored dot. 3. Record the color of this first dot. 4. Continue tracing from this dot to the next colored dot along the line, recording its color. 5. Repeat step 4 for each subsequent dot encountered along the line path. 6. Stop when no more dots appear on the line and list all recorded colors in order, followed by 'end'.
	V3	1. Find the starting point 'S' and note the immediate direction the line takes from this point. 2. Follow the line continuously, mentally marking each time you pass a colored dot. 3. Create a list in your mind of the colors of dots as you encounter them along the tracing path. 4. After completing the full trace from 'S' to the end of the line, verify your mental list by retracing quickly. 5. Convert your verified list of colors into the required format, ending with 'end'.

Table 17: **Model-generated tracing instructions (continued).**

Model	Ver.	Generated Instruction
Qwen3-VL-235B	V1	1. Locate the label 'S' marking the starting point on the line. 2. Begin moving along the line in the direction it initially curves away from 'S'. 3. Note the color of each dot encountered in the order they appear along the path. 4. Continue moving forward along the line without reversing direction until all dots on the path are recorded. 5. List the colors in the recorded order, followed by 'end'.
	V2	1. Identify 'S' as the starting position on the line. 2. Determine the overall direction of the line's path (e.g., clockwise spiral or counterclockwise spiral) from 'S'. 3. Follow the line consistently in that direction, noting each colored dot in sequence. 4. If the line loops inward or outward, maintain the established direction to avoid missing dots. 5. Stop when the line terminates or no more dots remain on the path.
	V3	1. Start at 'S' and visually follow the line to the nearest adjacent dot. 2. Record the color of this first dot. 3. From the current dot, proceed along the line to the next nearest dot, ensuring the line segment connects directly without crossing other segments. 4. Repeat step 3 for each subsequent dot until the end of the line is reached. 5. Combine all recorded colors in order, appending 'end' at the conclusion.

I.1 Prompt Engineering Does Not Stabilize Reasoning

We examine whether explicit tracing instructions stabilize both tracing performance and the reasoning process as density increases. If prompt engineering successfully induced a more robust tracing procedure, we would expect not only improved accuracy but also more stable performance and reasoning cost across density levels.

Explicit Instructions Do Not Stabilize Performance or Reasoning. As shown in Table 18, neither human- nor model-generated instructions consistently improve tracing accuracy or stabilize performance from *Low* to *High* density. This instability is also reflected in the reasoning process: reasoning length and self-correction frequency both continue to increase with density across models and prompt conditions (Table 19). For example, GPT-5.4 increases from 617 to 4,178 tokens under the *Human-instructed* prompt and from 716 to 4,031 under *Model-generated* prompts, while Qwen3-VL-235B rises from 13.30 to 37.55 self-corrections per sample under the *Human-instructed* prompt. In several cases, prompting therefore fails to suppress, and can even amplify, the increase in reasoning cost and repeated reconsideration as density rises.

Table 18: **Full accuracy results for the prompting intervention.** Tracing accuracy (%) under the *Standard*, *Human-instructed*, and three *Model-generated* prompting conditions across Swirl density levels. Each condition contains 200 samples per density level.

Model	Prompt	Low	Moderate	High
Sonnet-4.5	Standard	65.5	9.5	0.0
	Human-instructed	55.0	7.0	0.0
	Model-generated V1	51.5	3.0	0.5
	Model-generated V2	60.0	8.0	0.0
	Model-generated V3	32.5	2.5	0.0
GPT-5.4	Standard	99.5	95.0	72.5
	Human-instructed	94.0	91.5	71.0
	Model-generated V1	81.5	66.5	51.0
	Model-generated V2	98.0	95.0	75.0
	Model-generated V3	93.5	89.5	74.0
Gemini3-Flash	Standard	100.0	97.5	68.0
	Human-instructed	91.0	90.5	61.5
	Model-generated V1	99.0	97.5	67.0
	Model-generated V2	99.0	96.5	74.0
	Model-generated V3	94.5	93.5	73.0
Kimi-K3	Standard	87.5	67.0	19.5
	Human-instructed	96.5	80.5	29.0
	Model-generated V1	91.5	66.5	23.0
	Model-generated V2	96.5	76.5	21.5
	Model-generated V3	88.0	61.0	16.0
Qwen3-VL-235B	Standard	81.5	14.5	1.0
	Human-instructed	75.5	11.0	1.0
	Model-generated V1	84.0	11.5	1.5
	Model-generated V2	74.5	14.5	1.0
	Model-generated V3	78.5	12.0	1.5
GLM-4.6V	Standard	84.0	12.5	0.0
	Human-instructed	77.5	8.5	0.0
	Model-generated V1	81.5	7.5	0.0
	Model-generated V2	77.0	10.0	0.5
	Model-generated V3	82.0	10.5	0.0

Table 19: **Reasoning length and self-correction frequency across Swirl difficulty levels and prompt conditions.** Average reasoning-token usage and number of self-correction markers per sample are reported for *Low*, *Moderate*, and *High* density conditions. *Standard* uses no tracing instruction; *Human-instructed* uses a single hand-written instruction; *Model-generated* averages the three instructions each model wrote for itself.

		Reasoning tokens			**Self-corrections / sample**		
Reasoning Model	**Prompt**	**Low**	**Moderate**	**High**	**Low**	**Moderate**	**High**
Sonnet-4.5	Standard	390	510	537	4.20	6.14	6.39
	Human-instructed	448	521	534	4.44	6.15	6.58
	Model-generated	477	544	525	4.99	6.17	6.16
GPT-5.4	Standard	521	1,392	3,460	0.12	0.54	1.50
	Human-instructed	617	1,659	4,178	0.27	0.73	1.84
	Model-generated	716	1,812	4,031	0.33	0.84	1.81
Gemini3-Flash	Standard	557	800	1,337	0.47	0.88	1.66
	Human-instructed	878	1,238	2,463	0.56	1.14	3.00
	Model-generated	452	567	1,087	0.05	0.18	0.88
GLM-4.6V	Standard	477	706	854	9.25	15.64	19.49
	Human-instructed	690	1,111	1,434	11.20	22.79	30.98
	Model-generated	607	909	1,091	11.64	20.55	25.89
Kimi-K3	Standard	2,565	2,984	4,631	9.44	10.07	16.48
	Human-instructed	1,486	2,612	4,512	3.81	6.55	10.29
	Model-generated	1,830	3,093	4,498	6.01	8.80	10.70
Qwen3-VL-235B	Standard	657	997	1,310	10.23	16.70	24.23
	Human-instructed	867	1,612	2,119	13.30	29.34	37.55
	Model-generated	959	1,610	2,034	15.37	27.76	36.90

Explicit Instructions Do Not Stabilize Reasoning Strategies. Reasoning strategies remain sensitive to density across prompting conditions. As shown in Figure 26, the intensity of non-tracing strategies generally increases from *Low* to *High* density under *Standard*, *Human-instructed*, and *Model-generated* prompting, indicating that explicit tracing instructions do not consistently suppress the shift away from sequential tracing. Strategy categories are identified through keyword-based matching using the mappings in Table 13; accordingly, this intensity should be interpreted as an indirect indicator of reasoning-pattern change rather than an absolute measure of strategy use.

The qualitative reasoning traces show the same pattern more directly. As shown in Table 20, even when models are given their own tracing procedures, *Low*-density reasoning typically advances sequentially from S along connected dots, whereas *High*-density reasoning again shifts toward global reconstruction using rings, angles, radii, or other global criteria rather than line connectivity. Thus, even self-generated tracing instructions do not prevent the strategy shift observed as density increases.

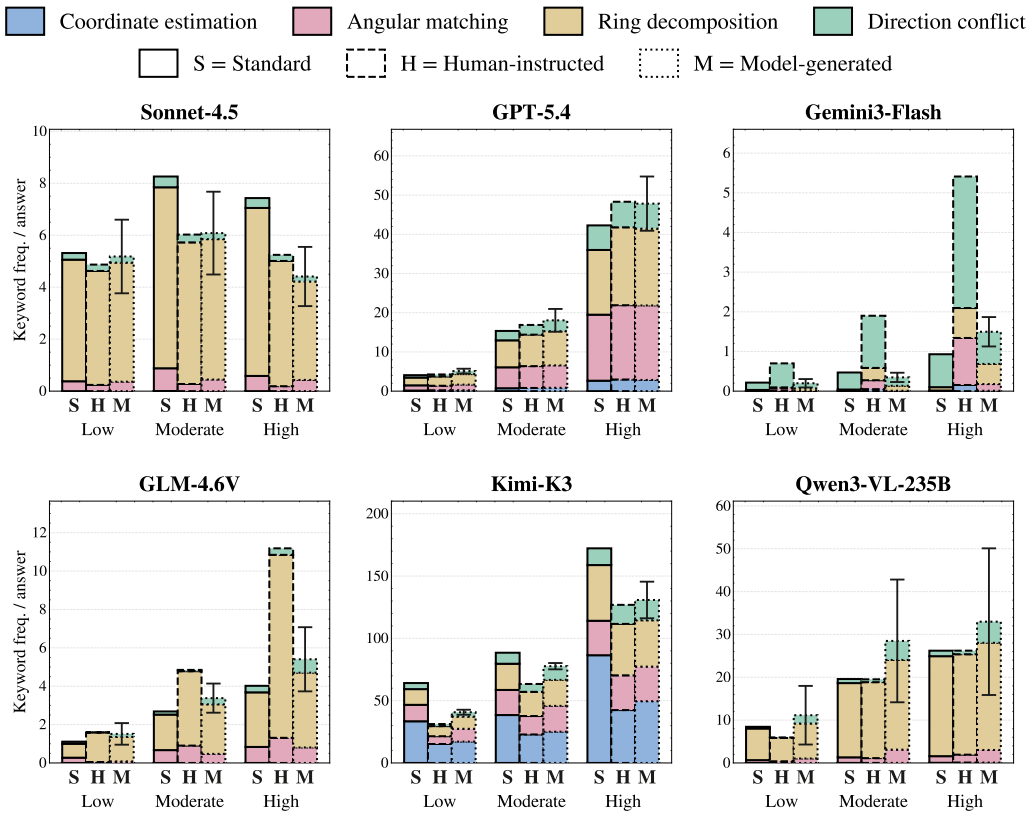


Figure 26: **Non-tracing strategy intensity across prompting conditions.** Keyword-based strategy intensity across *Low*, *Moderate*, and *High* density for *Standard*, *Human-instructed*, and *Model-generated* prompting. *Model-generated* reports the mean across three generated instructions, with error bars showing the standard deviation. The measure is intended as an indirect indicator of reasoning-pattern shifts rather than an absolute measure of strategy use.

Table 20: **Reasoning pattern examples under *model-generated* instructions, at *Low* vs. *High* rotation density.** Excerpts are verbatim from the *Model-generated* condition. Even when the model is given its own tracing instruction, *Low* density is handled by advancing from S to the next connected dot, whereas *High* density again induces ordering by a global criterion rather than by line connectivity.

Model	Density	Reasoning excerpt
Sonnet-4.5	Low	"From S, there appears to be a line that goes to a purple dot, then **follows the spiral outward** … The line goes to purple dot – Then to red dot – Then to green dot…"
	High	"Red sits near the **9–10 o'clock position on the inner ring**, while yellow is around **5–6 o'clock on that same inner ring**. Moving outward, green appears at roughly **3 o'clock on the middle ring**, blue at 6–7 o'clock…"
GPT-5.4	Low	"if I start from yellow, I can **follow along line one** to reach red, and then orange. After orange, I wonder if the line continues … how these colors **connect** and what follows next."
	High	"I need to **assign each dot an angle along the spiral corresponding to their increasing radius** and determine their sequence. … purple is roughly at $330°$ and brown around $390°$, both at inner radii; green at about $410°$…"
Gemini3-Flash	Low	"**Finding Start Point**: I've successfully identified the starting point 'S' on the spiral. My next step is to **trace the path** from there." (137 characters, single block)
	High	"**radial and angular positions** of the outermost points … Orange is at **6 o'clock** and Pink at **9 o'clock** on the outer loop. … I'm now **working backwards from E**, tracing counter-clockwise from E ($\sim$2 o'clock) past Yellow ($\sim$12 o'clock)…"
GLM-4.6V	Low	"locate the 'S' point. Then, **follow the line** from S. The first dot after S is green … then brown, then blue, then purple, then orange, then red, then pink, then yellow."
	High	"the **innermost loop** has S, green … then orange (**on the middle loop**), then yellow (**middle loop**), then purple (**outer loop**), then red (**outer loop**), then pink (**middle loop**), then blue (middle loop)?"
Kimi-K3	Low	"The line starts at S, **goes to** blue dot, **then curves down to** green dot, **then to** yellow dot, then to brown dot, then up to red dot, then orange dot at top…"
	High	"pink **(287,330)** lower-left inner; green **(309,255)** … **radius $\sim$70**; blue (301,214) top outer **radius $\sim$108**. **Colors positions by angle around center (320,320)**: red angle -80 r23; yellow r60 angle -5; green r65 angle -105…"
Qwen3-VL-235B	Low	"The **first adjacent dot from S** is the green one. Then **from green, the next dot along the line** is purple. Then from purple, the next is brown. Then blue, then pink, then yellow…"
	High	"**Inner circle**: purple at **3 o'clock**, orange at **9 o'clock**. **Counterclockwise order**: purple (3) $\rightarrow$ orange (9). **Middle circle**: yellow (4) $\rightarrow$ pink (8). **Outer circle**: brown (2), blue (10), green (8), red (4)."

J Full Results for Richer Visual Ambiguity Settings

To assess whether the path-preservation failures observed in our controlled experiments extend to visually richer settings, we evaluate tracing in tangled cables from HANDLOOM [42] and metro maps. These settings introduce crossings, overlaps, clutter, and realistic diagram structure, allowing us to test whether VLMs remain sensitive to nearby competing paths and exhibit the same path-preservation failures beyond simplified synthetic tasks.

J.1 HANDLOOM Cable Tracing

We evaluate reasoning-enabled models on tangled cable images from HANDLOOM [42]. In this setting, the target cable must be traced through scenes containing self-overlap, crossings, and nearby cable segments. We place 30 colored dots along each target cable, and models are asked to output these dots in endpoint-to-endpoint order.

As shown in Figure 27, cumulative prefix accuracy drops progressively across sequence steps. Models often begin on the correct cable, but their predictions become less reliable as the traced sequence grows longer and local ambiguities accumulate. This pattern is consistent with a gradual loss of the selected cable rather than a purely formatting-related error.

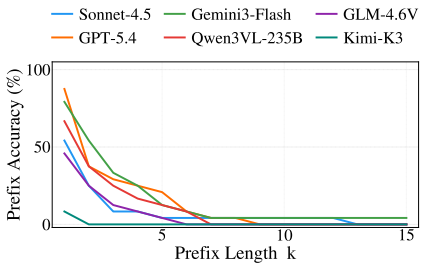
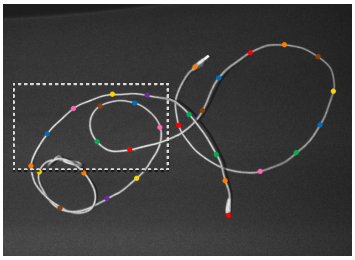


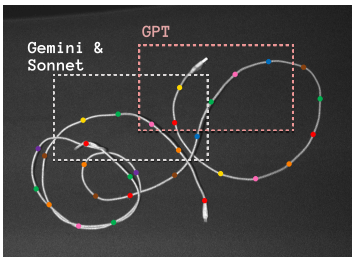
Figure 27: **Progressive sequence breakdown on HANDLOOM cable tracing.** Prefix accuracy on 24 cable images, plotted over the first 15 of 30 colored-dot steps.

To separate nearby-cable distraction from explicit intersection handling, we inspect local regions that are away from crossings. For each region, we align the model prediction with the local ground-truth colored-dot sequence. We treat a prediction as corresponding to that region only when it contains at least three consecutive colors that match the ground-truth sequence in either forward or reverse order. We then examine whether the model continues along the target cable or jumps to a nearby cable segment. Red text in Table 21 marks deviations from the ground truth. Across the aligned examples, models often produce a short correct subsequence before leaving the target cable and continuing along an adjacent segment, consistent with the path-switching pattern observed in the controlled tasks. We do not include additional Qwen3-VL-235B examples because no further outputs satisfied this matching criterion in either direction; Kimi-K3 is also excluded because it exceeded the 5,000-token generation limit without producing a final answer.

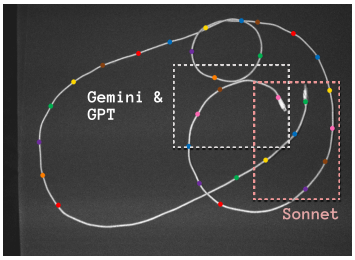
Table 21: **HANDLOOM cable-tracing failure examples.**



Source	Colored dot sequence
Ground truth	... green brown blue pink yellow ...
Gemini3-Flash	... green brown blue yellow purple ...
Ground truth	... orange blue pink yellow purple green ...
Sonnet-4.5	... orange blue pink yellow blue purple ...
Ground truth	... red green brown pink yellow ...
GPT-5.4	... red green brown blue yellow pink blue ...
Ground truth	... yellow pink blue brown green red ...
GLM-4.6V	... yellow pink blue red yellow purple green



Source	Colored dot sequence
Ground truth	... yellow green pink orange red ...
Gemini3-Flash	... yellow green pink red blue green ...
Ground truth	... yellow green pink orange red ...
Sonnet-4.5	... yellow green pink red blue ...
Ground truth	yellow red yellow pink ...
GPT-5.4	yellow green pink blue ...



Source	Colored dot sequence
Ground truth	... blue pink brown pink blue ...
Gemini3-Flash	... blue pink brown orange purple ...
Ground truth	... green yellow blue green ...
Sonnet-4.5	... green yellow blue pink yellow blue ...
Ground truth	... yellow purple orange green blue ...
GPT-5.4	... yellow purple orange brown pink blue ...

J.1.1 Cable attention analysis.

We apply the same internal-attention analysis described in Appendix G.2 to the tangled-cable setting using Qwen3-VL-8B-Thinking [4]. To focus on local competition rather than explicit crossing resolution, we sample regions away from crossings and self-overlap points. For each source dot s on the target cable, we manually label nearby dots on adjacent cables based on their spatial proximity to s, excluding the immediately preceding and following dots in the ground-truth sequence. Let t denote the true sequential next dot and $\{c_j\}_{j=1}^{N_C}$ the labeled nearby competitors. At each vision-encoder layer ℓ, we select the three nearby competitors receiving the highest attention from s and measure the attention preference as

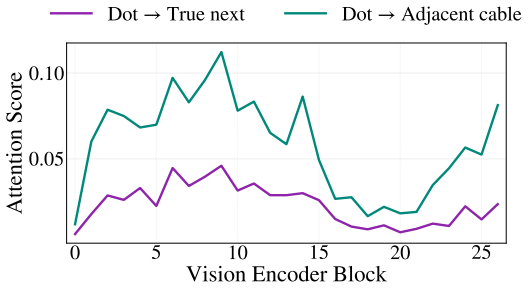


Figure 28: **Attention to true next dots and nearby cable competitors.** Qwen3-VL-8B-Thinking vision-encoder attention from the source dot to the true next dot versus a nearby cable competitor, averaged over HANDLOOM regions away from crossings and self-overlaps.

$$\Delta^{(\ell)}_{\text{attn,cable}} = a^{(\ell)}_{s\to t} - \frac{1}{3}\sum_{j\in\mathcal{T}_3^{(\ell)}(s)} a^{(\ell)}_{s\to c_j},$$

where $\mathcal{T}_3^{(\ell)}(s)$ denotes the three adjacent-cable competitors with the highest attention from s at layer ℓ. Thus, the second term represents the mean attention assigned to the three strongest nearby competing dots.

As shown in Figure 28, attention is frequently distributed toward nearby competitors rather than cleanly favoring the true next dot. This pattern is consistent with the behavioral failures above: even when the local region does not require resolving an explicit crossing, nearby cable structure can compete with the target continuation and weaken path preservation.

Cropped Cable Attention Examples. To visualize this effect at the example level, we further inspect cropped HANDLOOM regions. Each crop contains a source dot on the currently traced cable, the true sequential next dot, and nearby dots on spatially adjacent cable segments. For each selected source dot, we compare attention from the source region to the true next dot with attention from the same source region to the adjacent cable competitor. The visualizations in Figure 29 show the selected crop at early, middle, and late vision-encoder blocks, together with the corresponding block-wise attention scores.

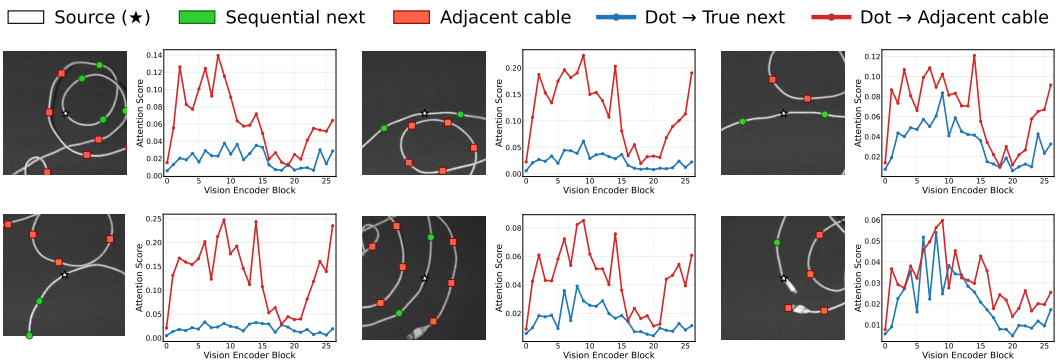


Figure 29: **Cropped HANDLOOM cable attention examples.** Each panel shows a selected crop and vision-encoder attention from the source dot to the true next dot versus a nearby adjacent-cable competitor.

Together, the prefix-accuracy trend, qualitative sequence deviations, average attention analysis, and cropped attention examples show that the HANDLOOM failures are consistent with the path-switching pattern observed in the controlled tasks.

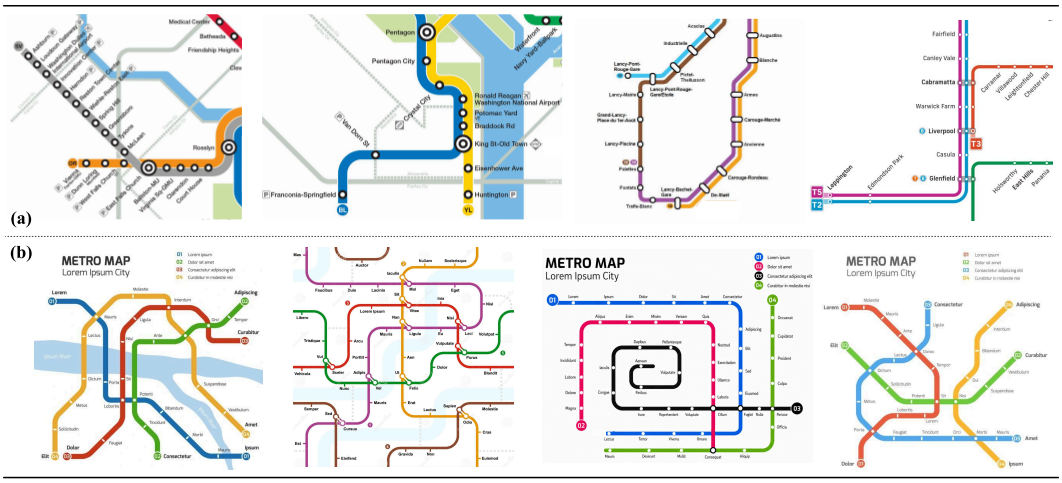


Figure 30: **Examples from the metro-map set.** The set contains (a) cropped regions from real transit maps and (b) synthetic images rendered in a metro-map style. Crops are selected to preserve local route ambiguity while avoiding regions dominated by extreme clutter or unreadable text.

J.2 Metro Maps

Following the HANDLOOM cable probe, we also evaluate reasoning-enabled models in a visually richer practical diagrammatic setting: metro maps. Metro-map reading depends directly on the tracing operation studied in this paper: a reader must maintain the identity of a selected route as it bends, runs near other colored lines, shares stations or segments, and competes with dense station labels. We use this setting as an additional practical probe of whether extended reasoning traces maintain stable local continuation under realistic diagram conventions.

J.2.1 Dataset and Prompt

To cover both real map conventions and controlled metro-like geometries, we construct a small metro-map set from cropped public-transit maps and synthetic images. Full city transit maps are often too large and visually dense for a diagnostic tracing evaluation, so we crop local regions with moderate complexity: enough nearby routes, junctions, and labels to make line identity nontrivial, while avoiding regions dominated by extreme clutter or unreadable text. The final set contains 20 images and 50 line-tracing queries. Examples are shown in Figure 30. In each query, the model is given a line color and a starting station, and must output the station names on that line exactly as written in the image, in traversal order.

We use the following prompt template for each metro-map query. The final answer is constrained to a station-name list; for reasoning-enabled models, we separately inspect the model-emitted reasoning trace.

Prompt for Metro Maps

Task: You must complete the sentence by listing only station names as written in the image, in sequential order, and append 'end' at the very end. Do not include any non-station text.

Visually tracing the path of the `{line_color}` line from `{start_point}` in the image, the sequence of stations on its path is strictly

J.2.2 Quantitative Results

As shown in Table 22, reasoning-enabled models remain unreliable on this practical tracing task. Gemini3-Flash, GPT-5.4, and Kimi-K3 show relatively stronger performance, solving 30/50, 29/50, and 28/50 queries, respectively. Performance then drops substantially: Sonnet-4.5 solves 16/50, Qwen3-VL-235B solves 9/50, and GLM-4.6V solves only 6/50. No model exceeds 60% exact-

Table 22: **Reasoning accuracy and reasoning length on the metro-map set.** Only reasoning-enabled variants are shown. Exact-match accuracy is reported over 50 queries, with raw counts in parentheses. Token counts report the average length of model-emitted reasoning traces, both overall and separately for correct and incorrect responses.

Model	Accuracy (%)	Avg. Tokens	Correct Avg.	Wrong Avg.
Sonnet-4.5	32.0 (16/50)	1533	998	1784
GPT-5.4	58.0 (29/50)	1522	793	2528
Gemini3-Flash	60.0 (30/50)	1518	673	2785
Qwen3-VL-235B	18.0 (9/50)	1807	917	2002
GLM-4.6V	12.0 (6/50)	2372	621	2610
Kimi-K3	56.0 (28/50)	1709	1219	2333

Table 23: **Self-correction behavior on the metro-map set.** Self-correction rate is reported over all 50 queries and separately over correct and incorrect responses. We use the model-specific lexical signals defined in Table 12. For Gemini3-Flash, we additionally count revision-style headers containing terms such as *Revisiting*, *Revising*, or *Re-examining*.

Model	Overall Self-Corr. (%)	Correct Self-Corr. (%)	Wrong Self-Corr. (%)
Sonnet-4.5	86.0 (43/50)	81.2 (13/16)	88.2 (30/34)
GPT-5.4	28.0 (14/50)	10.3 (3/29)	52.4 (11/21)
Gemini3-Flash	18.0 (9/50)	6.7 (2/30)	35.0 (7/20)
Qwen3-VL-235B	100.0 (50/50)	100.0 (9/9)	100.0 (41/41)
GLM-4.6V	100.0 (50/50)	100.0 (6/6)	100.0 (44/44)
Kimi-K3	94.0 (47/50)	89.3 (25/28)	100.0 (22/22)

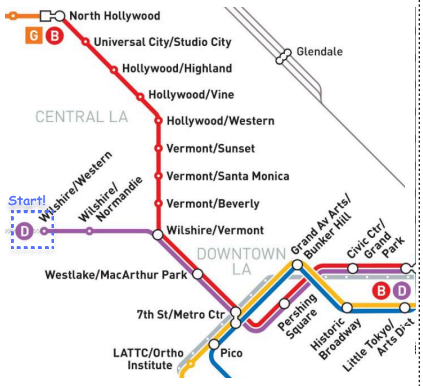
match accuracy. These results are consistent with the controlled and HANDLOOM evaluations: richer visual context and extended reasoning traces do not make line tracing uniformly reliable.

The reasoning length statistics suggest that long deliberations are not a reliable indicator of successful tracing. All models produce more than 1.5K reasoning tokens per query on average, and incorrect responses are consistently longer than correct responses. GLM-4.6V produces the longest traces overall while attaining the lowest accuracy, with incorrect responses averaging 2610 tokens compared to 621 for correct responses. Kimi-K3 exhibits the same direction of association, averaging 2333 tokens on incorrect responses and 1219 on correct responses. We do not interpret this as length causing failure. Rather, the pattern suggests that failures often coincide with unresolved ambiguity, repeated checking, or route-level reconstruction attempts that do not recover the correct local continuation.

The prevalence of self-correction markers in Table 23 does not show a consistent relationship with tracing accuracy. For GPT-5.4 and Gemini3-Flash, markers appear more often in incorrect responses, while GLM-4.6V and Qwen3-VL-235B use them pervasively despite their low accuracy. Kimi-K3 provides a complementary case: markers appear in 47/50 traces even though its accuracy is close to that of the best-performing models, occurring in all 22 incorrect traces as well as in 25 of 28 correct traces. Taken together, these patterns suggest that marker frequency may reflect a model-specific deliberation style rather than successful recovery of a lost path state.

J.2.3 Qualitative Examples

For completeness, we include representative metro-map reasoning traces from all six reasoning-enabled models in Figures 31 to 33. The captions summarize the main patterns in each example. Broadly, the traces exhibit plausible route-level descriptions, reconstructions of shared corridors, distraction from nearby station labels, and repeated self-checks. Yet none of these behaviors reliably ensures correct station-by-station continuation along the queried line. These examples further show that path preservation failures persist in the metro-map setting, particularly around locally ambiguous continuations. Overall, these traces suggest that additional reasoning and self-checking do not by themselves resolve tracing errors; successful correction requires identifying and revising the specific local connection at which the trace went wrong.



Sonnet-4.5 — Wilshire/Western, Wilshire/Normandie, Wilshire/Vermont, Westlake/MacArthur Park, 7th St/Metro Ctr, Pershing Square, Pico, Grand Av Arts/Bunker Hill, Civic Ctr/Grand Park, Historic Broadway, Little Tokyo/Arts Dist ✗

[...] Going east appears to be the more substantial path with additional stations. The map shows several stations in that area: 7th St/Metro Center, Pico, Pershing Square, Grand Avenue Arts/Bunker Hill, [...] I need to trace which of these stations actually lie on the purple line's route [...] the line likely connects them in sequence: 7th St/Metro Ctr through Pershing Square and Pico, then continuing to Grand Av Arts/Bunker Hill [...]

GPT-5.4 — Wilshire/Western, Wilshire/Normandie, Wilshire/Vermont, Westlake/MacArthur Park, 7th St/Metro Ctr, Pershing Square, Civic Ctr/Grand Park ✓

[...] I should identify the visible stations along the purple line from Wilshire/Western, which likely shares stations with the B line through Civic Ctr/Grand Park. [...] Thinking about the geometry: the red/purple line runs northeast from 7th to Pershing and then northwest to Civic Center. However, no actual route goes east to north from 7th to Pershing, since 7th St/Metro Center is southwest of Pershing Square. [...]

Gemini3-Flash — Wilshire/Western, Wilshire/Normandie, Wilshire/Vermont, Westlake/MacArthur Park, 7th St/Metro Ctr, Pershing Square, Civic Ctr/Grand Park ✓

[...] Now, it's a simple matter of recalling the sequential list. [...] I've successfully mapped the stations east of Wilshire/Western, hitting Westlake/MacArthur Park, and 7th St/Metro Ctr [...] navigating the shared section with the B Line, specifically from Wilshire/Vermont, all the way to Civic Ctr/Grand Park. [...] I've confirmed the shared section's stations: Pershing Square and Civic Ctr/Grand Park. [...] The final list is :

Qwen3VL-235B — Wilshire/Western, Wilshire/Normandie, Westlake/MacArthur Park, 7th St/Metro Ctr, Pershing Square, Historic Broadway, Little Tokyo/Arts Dist ✗

[...] the purple line: Wilshire/Western is the first. Then next is Wilshire/Normandie. [...] Then 7th St/Metro Ctr. Then Pershing Square. Then Historic Broadway. Then Little Tokyo/Arts Dist. Wait, let's check the labels. (Same Error) [...] Wait, let's confirm. The purple line is labeled D. The stations on the D line are (Same Error) [...]

GLM-4.6V — Wilshire/Western, Wilshire/Normandie, Wilshire/Vermont, end ✗

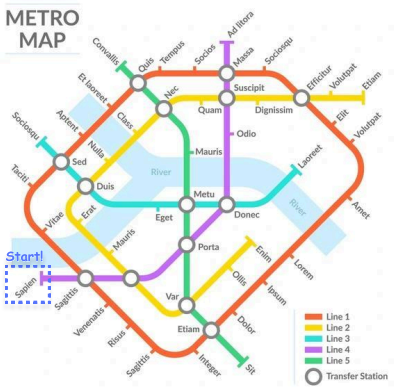
Wait, let me check the image again. The purple line (D) has stations: Wilshire/Western, Wilshire/Normandie, Wilshire/Vermont. Let me confirm. Yes, the purple line's path from Wilshire/Western: first Wilshire/Western, then Wilshire/Normandie, then Wilshire/Vermont.

Kimi-K3 — Wilshire/Western, Wilshire/Normandie, Wilshire/Vermont, Westlake/MacArthur Park, 7th St/Metro Ctr, Pershing Square, Civic Ctr/Grand Park, Little Tokyo/Arts Dist ✗

Image shows purple D line starts at Wilshire/Western (left), then Wilshire/Normandie, Wilshire/Vermont (transfer with red), Westlake/MacArthur Park, 7th St/Metro Ctr, Pershing Square, Civic Ctr/Grand Park? Need determine exact sequence as written and where path goes in image. [...] Need note image cropping right edge maybe Little Tokyo/Arts Dist partially at edge; include.

Figure 31: **Reasoning traces on a Los Angeles metro-map crop.** Models are asked to trace the purple line from *Wilshire/Western* and list the stations in order. Red text marks incorrect station names or erroneous reasoning claims. Sonnet-4.5 and Qwen3-VL-235B produce incorrect answers in the dense downtown portion of the map: Sonnet shifts from local line following to selecting among plausible nearby downtown stations, while Qwen skips a local transition and repeats the same incorrect route after self-checking. GPT-5.4 and Gemini3-Flash return the correct station sequence, but their traces are not fully explicit local traversals: GPT-5.4 includes an inconsistent geometric claim, and Gemini3-Flash assembles the answer through larger route chunks and shared-corridor cues. GLM-4.6V stops at *Wilshire/Vermont* after explicitly checking and confirming the truncated sequence. Kimi-K3 reconstructs the route through *Civic Ctr/Grand Park* but then extends it to *Little Tokyo/Arts Dist*, despite noting uncertainty at the edge of the crop.

Visually tracing the path of the **purple line** from '**Sapien**' in the image, the sequence of stations on its path is strictly ...



Ground Truth

Sapien, Sagittis, Porta, Donec, Odio, Suscipit, Massa, AD litora

Sonnet-4.5 — Sapien, Sagittis, Var, Porta, Metu, Suscipit, Odio ✗

[...] Sapien (bottom left) - Sagittis (next to Sapien) - Going upward/right, the purple line continues... - Porta (it passes through here - this is a transfer station) - Metu (another station on the purple line) - Odio (station on purple line) - Suscipit [...] the line appears to go from Metu toward Mauris, then continues to Suscipit [...] Looking more carefully at the geometry, I notice there's a station between Sagittis and Porta—Var appears to be on this segment. [...]

GPT-5.4 — Sapien, Sagittis, Mauris, Porta, Donec, Odio, Suscipit, Massa, AD litora ✗

Analyzing transit stations: [...] There's a purple line extending from Sagittis to another transfer that bends northeast. I see nearby text like Mauris and Var, [...] It seems the label for the circle station is likely Mauris, [...] The purple line connects Mauris to Porta but doesn't go through Var. I can list the sequence as Sapien, Sagittis, Mauris, and Porta. [...] In the upper area of the purple line, I see AD Litora, Massa, Suscipit, Odio, and Donec. [...]

Gemini3-Flash — Sapien, Porta, Donec, Metu, Odio, Suscipit, Massa, Ad litora ✗

[...] **Tracing the Route** I'm focused on the purple line, code named Line 4. Starting from 'Sapien,' a terminal, the next stop is a transfer station. Nearby labels are 'Sagittis' and 'Vitae'. [...] **Revisiting the Path** I've re-examined the purple line's path, starting from the north terminal, and traced it. Line 4 starts at 'Ad litora', followed by 'Massa', then 'Suscipit', then 'Odio', next is 'Metu', and then it goes to 'Donec'. I'm getting closer to a definitive ordered list. [...]

Qwen3VL-235B — Sapien, Sagittis, Var, Porta, Metu, Mauris, Odio, Suscipit, Massa, AD litora ✗

[...] Sapien is the first. Then next is Sagittis. Then Var. Then Porta. Then Metu. Then Mauris. Then Odio. Then Suscipit. Then Massa. Then Ad litora. [...] Let's confirm each station. Check the map: Purple line (Line 4): (Same Error) [...] Metu: transfer station, on purple.\n- Mauris: is Mauris on purple? Let's see. The purple line goes through Mauris? Wait, looking at the map, the purple line (Line 4) passes through Mauris.[...]

GLM-4.6V — Sapien, Sagittis, Mauris, Porta, Donec, Odio, Massa, Ad litora ✗

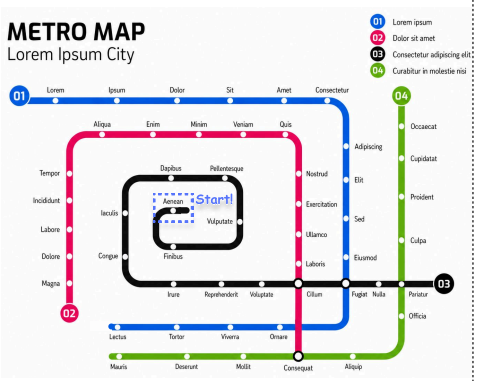
Let's look at the map. The purple line has stations: Sapien, Sagittis, Mauris, Porta, Donec, Odio, Massa, Ad litora. [...] Wait, no, maybe I missed. Purple line (Line 4) has stations: Sapien, Sagittis, Mauris, Porta, Donec, Odio, Massa, Ad litora? Wait, no, let's look again. [...] Let's confirm each station. [...] Yes, that's the sequence.

Kimi-K3 — Sapien, Sagittis, Porta, Donec, Odio, Quam, Suscipit, Massa, Ad litora ✗

At the lower ring (purple/yellow transfer), station name could be Quam (left) while Suscipit is a separate label for purple station just above? But there is only one node. [...] Transfer station might have two line-specific names? Unlikely. [...] Let's not overcomplicate; expected answer may be all visible labels touching purple line in order: Sapien, Sagittis, Porta, Donec, Odio, Quam, Suscipit, Massa, Ad litora.

Figure 32: **Reasoning traces on a synthetic metro-map image.** Models trace the purple line from *Sapien*. Red text marks incorrect station names or erroneous reasoning claims. All six reasoning-enabled models fail on this example. Several traces are drawn toward nearby off-route stations in the central transfer region, especially *Metu*: Sonnet-4.5, Gemini3-Flash, and Qwen3-VL-235B insert *Metu* and other nearby labels, while GPT-5.4 mistakes nearby text such as *Mauris* or *Var* for the continuation after *Sagittis*. Gemini3-Flash skips the early *Sagittis* transition, inserts *Metu*, and briefly verifies the path from the opposite terminal rather than maintaining a forward local traversal from the starting station. GLM-4.6V likewise inserts *Mauris* and, despite repeatedly checking the route, omits *Suscipit*. Kimi-K3 explicitly considers the ambiguity between *Quam* and *Suscipit* but retains both, thereby adding the off-route station *Quam*. The example illustrates how plausible reasoning and explicit rechecking can still fail to preserve the queried line through nearby station-label distractors.

Visually tracing the path of the **black line** from '**Aenean**' in the image, the sequence of stations on its path is strictly ...



Ground Truth

Aenean, Finibus, Vulputate, Pellentesque, Dapibus, Iaculis, Congue, Irure, Reprehenderit, Voluptate, Cillum, Fugiat, Nulla, Pariatur

Sonnet-4.5 **Aenean, Vulputate, Finibus, Irure, Reprehenderit, Volutpate, Cillum, Fugiat, Nulla, Pariatur** ✗

It's an inner loop-like structure [...] Moving clockwise from Aenean: the line goes to Vulputate. From Vulputate, the line goes down to Finibus... [...] There's also a separate bottom section running horizontally through Irure, Reprehenderit, Volutpate, Cillum, Fugiat, Nulla, and Pariatur. The path appears to go: Aenean → Vulputate → Finibus, then transitions to a separate bottom segment containing Irure → Reprehenderit → Volutpate → Cillum → Fugiat → Nulla → Pariatur. [...]

GPT-5.4 **Aenean, Finibus, Vulputate, Pellentesque, Dapibus, Iaculis, Congue, Irure, Reprehenderit, Voluptate, Cillum, Fugiat, Nulla, Pariatur** ✓

[…] It seems the black line forms a spiral loop, starting near Aenean, which is at the top of the inner loop. After Aenean, the next station to the right is Vulputate, followed by Finibus at the bottom. […] I'll need to trace the spiral carefully to clarify! [...] It begins at the station Aenean, moving left and down to the bottom station Finibus. The line then continues right to Vulputate [...]

Gemini3-Flash **Aenean, Vulputate, Pellentesque, Dapibus, Iaculis, Congue, Finibus, Irure, Reprehenderit, Voluptate, Cillum, Fugiat, Nulla, Pariatur** ✗

[…] I'm now carefully tracking the black line's spiral from \"Aenean,\" and mapping out each station accurately. I've re-examined the line, and the route is clear now. I'm focusing on the sequential order of stations like \"Vulputate,\" \"Pellentesque,\" and the rest. [...] I've corrected some earlier assumptions about the spiral's twists [...] My next step is the final formatting for the stations.

Qwen3VL-235B **Aenean, Vulputate, Finibus, Irure, Reprehenderit, Fugiat** ✗

[...] The stations on the black line, starting from Aenean. Let's see: Aenean → Vulputate → Finibus → (then the loop?) Wait, no. Let's check the labels. - Aenean - Then, the next station: Vulputate (there's a white dot there) - Then, Finibus (another white dot) - Then, Irure (next) - Reprehenderit [...] No, Cillum is on the blue line. [...] Wait, but let's count: 1. Aenean 2. Vulputate 3. Finibus 4. Irure 5. Reprehenderit 6. Fugiat. Yes, that's six stations.

GLM-4.6V **Aenean, Vulputate, Finibus, Dapibus, Pellentesque, Iaculis, Congue, Cillum, Voluptate, Reprehenderit, Irureend** ✗

[...] So starting at Aenean, then what's next? [...] Aenean is connected to Vulputate? Then Vulputate to Finibus? Then Finibus to Dapibus? [...] Wait no, wait maybe I'm getting it wrong. [...] Wait I'm stuck. [...] maybe the order is Aenean, Vulputate, Finibus, Dapibus, Pellentesque, Iaculis, Congue? [...]

Kimi-K3 **Aenean, Finibus, Vulputate, Pellentesque, Dapibus, Iaculis, Congue, Irure, Reprehenderit, Voluptate, Cillum, Fugiat, Nulla, Pariatur** ✓

[...] Starting from Aenean, the black line goes through: Aenean → Finibus → Vulputate → Pellentesque → Dapibus → then connects to the main horizontal black line [...] Actually Dapibus connects left to Iaculis/Congue area. [...] Starting at Aenean and tracing: Aenean → Finibus → Vulputate → Pellentesque → Dapibus → Iaculis → Congue → Irure → Reprehenderit → Voluptate → Cillum → Fugiat → Nulla → Pariatur.

Figure 33: **Reasoning traces on a synthetic metro-map image.** Models trace the black line from *Aenean*. Red text marks incorrect station names or erroneous reasoning claims. Most models fail by taking a nearby but incorrect continuation within the inner loop: Sonnet-4.5, Gemini3-Flash, Qwen3-VL-235B, and GLM-4.6V begin by moving from *Aenean* toward *Vulputate* rather than the correct next station *Finibus*. Their subsequent self-checks do not recover the missed local transition; GLM-4.6V in particular repeatedly questions the ordering without repairing it. GPT-5.4 and Kimi-K3 instead arrive at the correct sequence. GPT-5.4 initially describes the wrong continuation toward *Vulputate* but later revisits the loop geometry, while Kimi-K3 restores the initially omitted *Iaculis* after reconsidering the connection from *Dapibus*. This example echoes the swirl setting: in a loop-like local structure, reasoning can be drawn to nearby plausible continuations, and only a correction grounded in the specific local connection restores faithful step-by-step tracing.

J.2.4 Data Sources and Licensing

The metro-map set combines three sources. First, we use synthetic metro-map images from *BabyVision* [9], whose public dataset release lists the MIT license, and we follow its stated license and terms of use. Second, we crop a small number of real transit-map images from the *RewardMap*/ReasonMap-Plus resources [11]; the ReasonMap-Plus dataset is listed under Apache License 2.0, and these crops are used only for evaluation with the original attribution and licensing information preserved. Third, we create one additional synthetic metro-map image ourselves. The dataset does not contain personal or sensitive information.

K LLM Usage

We used LLM-based tools as auxiliary aids for writing, grammar and clarity edits, formatting, and routine coding support, such as debugging, refactoring, and generating boilerplate code. Their use was limited to improving presentation and workflow efficiency, and the authors retained full responsibility for the technical content, implementation, experimental results, analyses, and claims presented in the paper.

L Broader Impacts

This work studies a basic visual limitation of current vision-language models: their difficulty in maintaining a selected visual path under nearby local competition. Its potential positive impact is primarily diagnostic, as it provides a concrete way to evaluate whether VLMs can reliably follow connected visual structures in settings such as diagrams, maps, and deformable objects. The main potential negative impact comes from over-reliance on current VLMs in applications where path-switching errors could lead to incorrect downstream decisions. Our work does not introduce a deployed system, automate decisions about people, or release a model with new capabilities for misuse; rather, it highlights a limitation and suggests that VLM outputs on tracing-dependent tasks should be treated as assistive or diagnostic unless validated by task-specific checks or human review.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction summarize the paper's main claims and intended scope. Supporting evidence appears in Sections 3, 4, 5 and Appendix J, with limitations in Appendix A.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix A discusses task scope, interface assumptions, API/version dependence, and the behavioral nature of the failure-mode interpretations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not present theoretical results, formal theorems, or proofs. The equations introduced in the paper define empirical diagnostic metrics for attention and representation preferences in Section 4.2, rather than theoretical claims requiring proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Model and inference details are given in Appendix B, with task construction, prompts, and evaluation protocols in Appendices C and D. Additional diagnostic and real-world analyses are documented in Appendices E to J.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The experiments use existing model APIs, publicly available open-source models, and standard model code for the internal analyses. To support reproducibility, we provide the experimental data in the supplementary materials and describe the evaluated models, inference settings, task construction, prompts, evaluation criteria, and diagnostic procedures in Appendices B, C, D, and G.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: No model training is performed; evaluated checkpoints and inference settings are listed in Appendix B. Dataset sizes, prompts, response formats, correctness criteria, and diagnostic procedures are provided in Appendices C, D, G, and J.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the main Circuit Connections and Swirl evaluations, the paper runs each model three times and reports mean accuracy with standard deviation in reported accuracy. Other diagnostic experiments report aggregate rates or counts over fixed evaluation sets in the corresponding appendix sections.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide compute-resource information in Appendix B. Proprietary-model experiments were run through external model APIs, while the small sized open-source model evaluations and internal attention/representation analyses were run on NVIDIA L40S GPUs; all experiments were inference-only, with no model training or fine-tuning.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. The paper evaluates existing VLMs on synthetic or benchmark visual stimuli, does not involve personal data, and preserves anonymity in the submission.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses potential positive impact from improving diagnostics for VLMs that must follow connected visual structures, as well as potential negative impact from over-reliance on VLM outputs in tracing-dependent applications. This discussion is provided in Appendix L.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release pretrained models, generative systems, or scraped datasets with high misuse risk. The introduced and evaluated assets are controlled tracing stimuli and benchmark metadata, and the metro-map set is documented as containing no personal or sensitive information in Appendix J.2.4.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Existing assets used in the paper are credited through citations and documented in Appendices B and J.2.4. We report dataset licenses or access terms where available, and access model checkpoints and APIs through their public release or service terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We provide the evaluation image datasets and ground-truth labels for Circuit Connections, Swirl, Cable, and Metro Maps in the anonymized supplementary materials.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, outside human-subject experiments, or participant-provided data. All evaluations are conducted on VLMs using synthetic or benchmark visual stimuli.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The human performance baseline was conducted as an informal, minimal-risk evaluation by an author and fellow lab members, without crowdsourcing or recruiting external participants. The evaluation involved only a non-invasive visual tracing task using synthetic images, and no formal IRB approval or equivalent ethics review was obtained. require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as an important, original, or non-standard component of the core methodology. Their use was limited to writing, editing, formatting, and routine coding support, as described in Appendix K.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.