Is Visual Prompting All You Need? Studying VLM Spatial Reasoning under Progressive Visual Scaffolds

Lars Benedikt Kaesberg*, **Tianyu Yang***, **Florian Valentin Wunderlich**, **Terry Ruas**, **Daniel Kurzawe**, **Jan Philip Wahle**†, **Bela Gipp**†
University of Göttingen, Germany

***Equal contribution** †**Shared last authorship**
Correspondence: {l.kaesberg, tianyu.yang}@uni-goettingen.de

Abstract

Vision-language models (VLMs) have advanced rapidly in multimodal reasoning, yet recent work shows that their failures often reflect an interaction between visual grounding and downstream reasoning. What remains less clear is how the visual presentation of a task shapes model performance and failure modes when the underlying reasoning problem is unchanged. We study this question in SPaRC, a benchmark for grid-based visual spatial planning, by introducing lightweight input-side scaffolds that preserve the visual modality while making spatial structure more accessible. The scaffolds are task-specific diagnostic interventions built on SPaRC's known board structure, not a general-purpose prompting strategy. Across multiple VLMs, these scaffolds improve task accuracy over the original visual setting by up to 34.0 percentage points and further complement GRPO-based training, yielding up to 4.6 additional accuracy points compared with near-zero gains on the original visual input. Analyses on both end-to-end task solving and object detection show that these gains are closely tied to reductions in grounding-related errors, while rule reasoning remains comparatively challenging. With the planning problem held fixed, visual presentation therefore changes both accuracy and the distribution of failure modes, which in turn determines whether a VLM benchmark measures grounded perception, downstream reasoning, or a mixture of both.

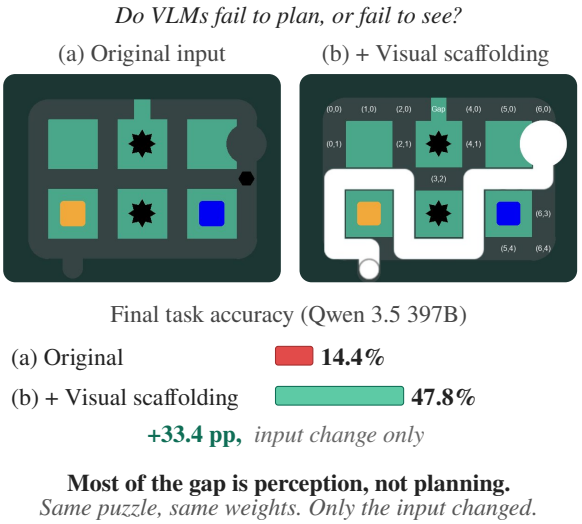


Figure 1: Impact of visual scaffolding on task solving.

1 Introduction

Vision-language models (VLMs) have rapidly advanced as a general framework for integrating visual perception with language-based reasoning (Singh et al., 2026; Comanici et al., 2025; Bai et al., 2025; Wang et al., 2025). By augmenting large language models (LLMs) with pretrained visual encoders such as CLIP-ViT (Liu et al., 2023; Radford et al., 2021), VLMs integrate visual inputs into language modeling, enabling them to support increasingly information-intensive tasks and establishing them as a key building block for applications ranging from embodied artificial intelligence (AI) to world models (Yin et al., 2024). Yet, despite rapid progress, the mechanisms underlying the multimodal understanding ability of VLMs remain poorly understood.

Recent work shows that apparent multimodal reasoning failures often reflect how visual information is grounded, verbalized, or bypassed, not just downstream reasoning limits (Wu et al., 2025; Asadi et al., 2026; Shahgir et al., 2026). Prior work addresses such failures through model-side adaptation (Chen et al., 2025a; Zhang et al., 2026; Jain et al., 2026; Chen et al., 2025b), inference-time visual tools (Hong et al., 2026), or input-side image scaffolding (Yang et al., 2023), yet none isolate which dimensions of the visual representation itself induce difficulty for planning. Diagnostic studies often replace images with text or probe isolated subtasks (Kaesberg et al., 2025; Wu et al., 2025), sidestepping the visual interface or obscuring its

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 51505–51526
October 24-29, 2026 ©2026 Association for Computational Linguistics

effects on downstream planning. We therefore ask how VLM task performance and failure modes change as the same planning problem is presented with progressively stronger visual support.

SPaRC (Kaesberg et al., 2025) is a grid-based visual puzzle benchmark that provides a controllable testbed for studying this question. Each puzzle presents a board with structured elements such as grid cells, rule symbols, and start/end positions, and requires the model to construct a valid path under explicit constraints. Because solving the task depends on both perceiving these elements and reasoning over their relations, errors in visual grounding can easily propagate into downstream planning failures, making the source of model weakness difficult to isolate in the standard evaluation setting. In this paper, we therefore construct a hierarchy of lightweight visual scaffolds, ranging from the original puzzle image to variants with explicit start/end markers, coordinate grids, cell-level coordinate annotations, and text-rendered symbols. Using this hierarchy as a diagnostic intervention, we evaluate end-to-end task accuracy, object detection accuracy, and path-level error categories covering endpoint grounding, path connectivity, self-intersection, and rule satisfaction. Together, these evaluations allow us to track how model performance and failure modes change as perceptual support increases, while the underlying board layout, rule configuration, and target solution remain fixed.

Our results reveal a consistent pattern across the tested configurations. Without any model fine-tuning, increasing input-side perceptual support substantially improves VLM performance on SPaRC, with gains of up to 34.0 percentage points over the original visual format. These large zero-shot gains show that the input representation itself accounts for a substantial share of the difficulty measured on this benchmark, even for strong frontier-scale models. When we fine-tune models using GRPO (Shao et al., 2024), visual interventions yield larger training gains under scaffolded visual inputs than under either the original visual input or the text-formulation baseline. These improvements are reflected in both aggregate accuracy and the structure of model errors. Endpoint localization, coordinate grounding, and invalid path construction have improved substantially, while rule satisfaction remains comparatively challenging. These findings refine prior evidence on perception bottlenecks: rather than showing that perception is *the* bottleneck, they show that with the planning problem held fixed, visual presentation changes both accuracy and the distribution of failure modes.

Overall, our contribution can be summarized as:

- We revisit multimodal reasoning from a data-centric perspective and argue that input-side perceptual support can materially affect downstream planning performance.
- Using the visual spatial planning benchmark SPaRC as a testbed, we introduce a hierarchy of lightweight visual scaffolds that progressively increase perceptual support and use it to diagnose VLM behavior on both the planning task and a board-element detection task.
- Extensive experiments over the benchmark show that our scaffolding hierarchy separates grounding errors from residual reasoning challenges, suggesting that visual presentation shapes not only model performance, but also how perception and reasoning are entangled in benchmark evaluation.[1]

2 Related Work

2.1 Multimodal Reasoning in VLMs

VLMs extend LLMs with pretrained visual encoders and achieve strong performance across visual question answering, chart and document understanding, embodied decision making, and multimodal agent tasks (Liu et al., 2023; Kimi Team et al., 2026; Bai et al., 2025; Wang et al., 2025; Zheng et al., 2026; Yang et al., 2025; HY-World et al., 2026; Qwen Team, 2026a). End-to-end accuracy on such benchmarks, however, conflates the ability to perceive the visual input with the ability to reason over what was perceived (Wang et al., 2026). This conflation is especially acute in visual spatial planning, where a single failure mode may reflect either a reasoning limitation or an earlier failure to ground the spatial structure the reasoning depends on. Our work directly targets this ambiguity by varying only the visual representation while holding the underlying planning task fixed.

[1] Our code and data are available on GitHub.

2.2 Visual Spatial Planning and Representation Design

In visual spatial planning, image-based inputs consistently underperform textual descriptions of the same environment, a gap commonly read as a fundamental modality limitation (Wu et al., 2025; Aghzal et al., 2024). For instance, o4-mini drops from 15.8% accuracy on the SPaRC text version to 5.6% on the visual input (Kaesberg et al., 2025). Prior mitigations adapt the model via fine-tuning or merging (Chen et al., 2025a; Zhang et al., 2026; Jain et al., 2026; Chen et al., 2025b), add inference-time visual tools such as crop or refocus (Yang et al., 2026; Guo et al., 2025; Wu et al., 2026), or scaffold the image with overlays, sketches, or structured renderings (Yang et al., 2023; Hu et al., 2024; Menon et al., 2024). Our proposal is most similar to the last one, but it uses single scaffolds for grounding instead of isolating which factors of perception drive reasoning. CoordConv makes a related point outside VLMs: convolutional networks fail trivial coordinate-transform tasks until (i, j) coordinate channels are added to the input (Liu et al., 2018). Our Axis Labels and Cell Coordinates scaffolds render coordinates into the image instead. We propose no architecture, and unlike that task-agnostic module our scaffolds are built from SPaRC's known structure (§3.2). We intervene on the input representation itself by progressively making endpoint identity, spatial indexing, and symbol semantics explicit while preserving the planning task, which allows us to identify which factors close the vision-text gap and whether the gap is driven by modality or representation.

3 Methodology

In the following, we describe the methodology of this paper, including the task setting (§3.1), input representations (§3.2), input conditions (§3.3), and evaluation metrics (§3.4).

3.1 SPaRC as a Testbed

We conduct our study on SPaRC (Kaesberg et al., 2025), a benchmark for grid-based visual spatial planning. The task takes place on a 2D grid, and each sample is defined by a grid G of dimensions $m \times n$. This grid is populated with a set of rule cells R, a designated start node S, and a designated end node E. The dataset has 1000 examples (500 for training and 500 for testing) with different sizes and difficulty levels, ranging from level 1 (easiest) to level 5 (hardest). The objective for the agent is to generate a sequence of coordinates representing a valid continuous path

$$C = (c_0, c_1, \ldots, c_k),$$

where each coordinate is denoted as $c_i = (x_i, y_i)$. See Appendix B for the full rule set and further details on the dataset.

3.2 Input-Side Visual Scaffolding

To study how perceptual accessibility affects spatial planning, we construct a hierarchy of board representations that progressively expose spatial structure while preserving the underlying task. Each step adds a specific form of perceptual support, and we use changes in downstream error profiles as diagnostic measurements. Figure 2 provides examples of each representation, and we describe the added support and the error types expected to be most sensitive to it.

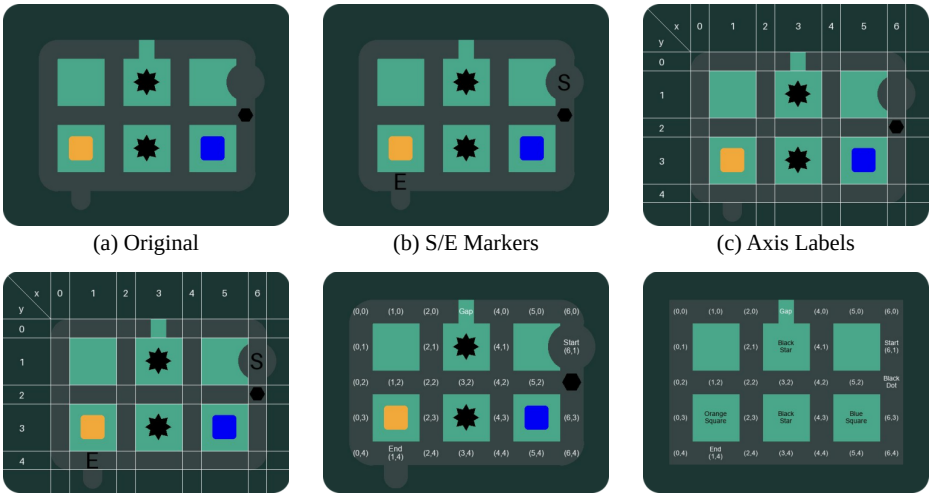
S/E Markers (S/E): Retains the original board layout and symbols, but explicitly marks the start (`S`) and exit (`E`) cells, providing the lowest level of perceptual assistance. We expect this to be most relevant to endpoint-localization errors and downstream complete-path failures from incorrect start or exit grounding.

Axis Labels (Axis): Adds row and column indices along the board borders, analogous to spreadsheet-style coordinates, providing global spatial-reference support. We expect it to affect coordinate-referencing and spatial-alignment errors, as well as path-structure errors from unstable global grounding.

Axis Labels + S/E (Axis+S/E): Combines **Axis** and **S/E**. We expect it to reduce endpoint- and coordinate-related errors, with possible downstream benefits for path connectivity and complete-path validity.

Cell Coordinates (Cell Coord.): Extends spatial scaffolding by printing each cell's coordinate (e.g., $(1, 3)$) directly inside the board, removing ambiguity in cell localization and providing stronger local spatial-grounding support. We expect this to be especially relevant to localization, coordinate-binding, path-connectivity, and complete-path-validity errors.

Text Symbols (Text Sym.): Builds on **Cell Coord.** by replacing graphical rule symbols with text labels (e.g., a star icon as the word "star"),



(a) Original (b) S/E Markers (c) Axis Labels

(d) Axis Labels + S/E (e) Cell Coordinates (f) Text Symbols

Figure 2: Overview of the input representations used in our study: **Original**, **S/E Markers**, **Axis Labels**, **Axis Labels + S/E**, **Cell Coordinates**, and **Text Symbols**. These variants progressively increase perceptual support by making endpoint identity, spatial indexing, and symbol semantics more explicit, while keeping the underlying planning task unchanged.

making both spatial indexing and symbol identity explicit. We expect it to be most relevant to symbol-recognition and rule-grounding errors, while also revealing residual constraint-reasoning errors that persist when visual information is explicit.

Figure 2 illustrates this progression from the original visual board to increasingly scaffolded variants, ending with a representation that renders rule symbols as text. Across all variants, we keep the board layout, rule configuration, and planning objective fixed, changing only how spatial information is presented. Later representations inherit support from earlier ones and may affect multiple error types simultaneously, so we do not assume a one-to-one correspondence between representation and failure category. We use the hierarchy to examine how the model's error profile changes as perceptual burden is reduced, and in particular where grounding-sensitive failures give way to residual reasoning failures. The original SPaRC textual formulation is reported separately in Appendix C as a non-visual reference.

Scope. Each scaffold is built from SPaRC's known structure: the task requires identifying the start and exit, emitting coordinates, and reading rule symbols, and each variant makes one of these explicit. They are therefore task-specific diagnostic interventions rather than a general prompting strategy. The steps are also cumulative, since later variants add coordinate and semantic cues along with visual structure; Appendix E.2 decomposes the Axis Labels step. Text Symbols in particular shifts part of the grounding burden toward reading in-image text.

3.3 Visual Degradation and Recovery

To probe how much of the observed difficulty is tied to visual grounding, we construct three settings that reduce the quality of the original board image while preserving the underlying planning problem (see Appendix D for examples). **Low Contrast** desaturates the board colors, making symbols less visually distinctive. **Low Resolution** downsamples the image, reducing fine-grained pixel detail. **Rotated** rotates the entire board, disrupting its canonical orientation. For each degradation, we construct a recovery variant by overlaying the **Cell Coordinates** representation on the degraded board, allowing us to test whether degraded performance can be recovered by restoring access to the board's spatial structure, without also converting symbol identities into text.

3.4 Evaluation Tasks and Metrics

We evaluate model behavior under two complementary settings: end-to-end task solving and object detection, which together let us analyze how the representation hierarchy in §3.2 affects both planning behavior and the visual grounding that supports it.

Task solving. The model is given a board representation and asked to generate a path from the start cell to the exit that satisfies the rule constraints. Our primary metric is **Final Task Accuracy**, the fraction of puzzles for which the predicted path is completely correct. To track how failures shift with visual scaffolding, we additionally define four path-level validity conditions. **Endpoint correctness** checks whether the generated path connects the designated start and exit cells. **Connectivity** requires every consecutive pair of coordinates to be adjacent under Manhattan distance, i.e., $\|c_{i+1} - c_i\|_1 = 1$. **Non-intersection** requires that the path not revisit any coordinate. **Rule-cell avoidance** requires that the path never enter a rule cell, i.e., $c_i \notin \mathcal{R}$ for all i. We report the satisfaction rates of these four conditions as **Correct Start/End**, **Connected Path**, **Non-Intersecting**, and **No Rule Violation**, and additionally report **Fully Valid**, the fraction of paths satisfying all four at once.

Object detection. The model is asked to identify the rule symbols present on the board, providing a direct readout of visual recognition that complements the path-level analysis. We report **Exact-Match Board Accuracy** (the fraction of boards on which all rule symbols are identified correctly), **Average Rule Detection Accuracy** (per-rule accuracy averaged across boards), and **Per-Type Detection Accuracy** (accuracy per symbol type). Comparing Cell Coordinates with Text Symbols isolates gains from local spatial grounding versus explicit symbol semantics.

Detection uses a separate prompt (Appendix F.3) and is independent of the planning answer. The model sees the board image and emits the whole board as a 2D array of symbol codes in a fixed schema, which a deterministic script parses and compares cell by cell against the ground truth. For a ground-truth Start cell, a predicted label of Path, End, or Gap contributes to the corresponding off-diagonal entry of the confusion matrices in §4.3.

4 Results and Analysis

In this section, we present our experimental findings. We first describe the experimental setup (§4.1), then analyze zero-shot behavior through end-to-end planning accuracy (§4.2), board-element detection (§4.3), and path-level error profiles (§4.4). We further evaluate whether scaffolding recovers performance under visual degradation (§4.5) and whether it complements GRPO post-training (§4.6).

4.1 Experimental Setup

We evaluate seven open-weight VLMs, namely Mistral Small 3.2 (Mistral AI, 2025), Gemma 3 27B (Gemma Team et al., 2025), Llama 4 Scout (Meta AI, 2025), GLM 4.6V (V Team et al., 2025), Gemma 4 31B (Gemma Team, Google DeepMind, 2026), Qwen 3.5 27B and 397B (Qwen Team, 2026b). The selection covers dense and Mixture-of-Experts architectures with 24B to 397B parameters. For the GRPO experiments (§4.6), we use Qwen 3 VL 4B Thinking and 8B Thinking (Bai et al., 2025) as base models. All inferences use greedy decoding unless stated otherwise. Technical details about the models and hardware, including the configuration used for GRPO training, are reported in Appendix A.

The test split contains 500 puzzles. Appendix E.1 reports 95% bootstrap confidence intervals for every model and condition, and paired intervals for the differences we discuss.

4.2 Effect of Visual Scaffolding on Task Solving

We examine end-to-end task-solving performance across the representation hierarchy introduced in §3.2. As shown in Figure 3, input-side visual scaffolding consistently improves accuracy over the original visual board, especially for stronger models. For example, Qwen 3.5 397B improves from 14.4% on the Original board to 48.4% with Text Symbols, Qwen 3.5 27B improves from 8.2% to 40.0% with Cell Coordinates, and Gemma 4 31B improves from 2.2% to 26.2% with Text Symbols. Paired bootstrap intervals exclude zero for all three (Appendix E.1). The two models at the accuracy floor are the exception: Gemma 3 27B and Mistral Small 3.2 stay below 1% on every visual condition and show no resolvable gain from any scaffold. These gains suggest that a large part of the difficulty measured on the Original board is attributable to perceptual access rather than to planning difficulty alone, and that perceptual support is most useful when models have enough planning capacity to exploit it, a pattern further supported by the auxiliary completion-token analysis in Appendix E.6.

The best scaffolded visual inputs exceed the SPaRC text-formulation baseline. Qwen 3.5 397B reaches 48.4% with Text Symbols versus 38.0% on

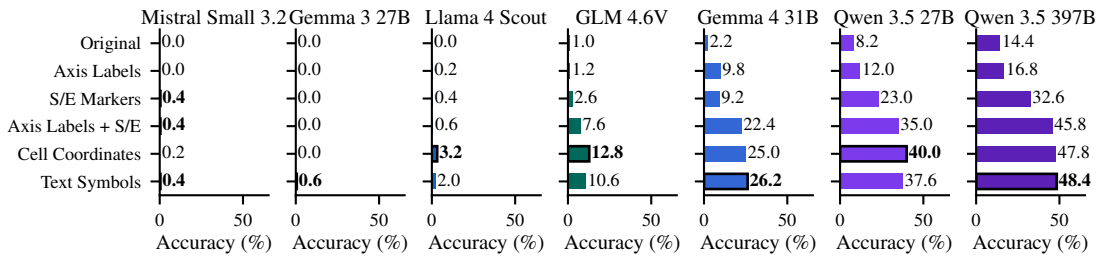


Figure 3: Task-solving accuracy across board representations. We compare the **Original** SPaRC visual input and progressively scaffolded visual variants. The best results are highlighted in **bold**.

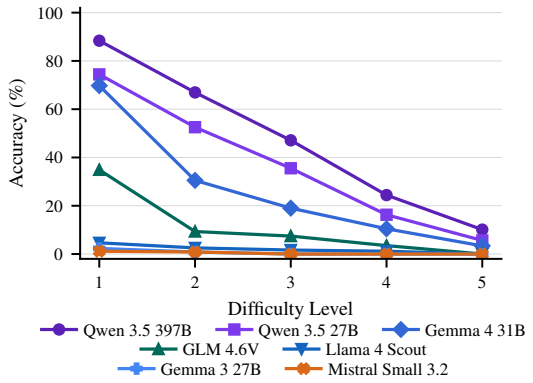


Figure 4: Task-solving accuracy stratified by puzzle difficulty for Text Symbols representation.

the text baseline, a **+10.4 pp** gain from changing the input format alone. The text formulation is thus not an upper bound on reasoning ability for this task. Per-model text-vs-visual comparisons and the human reference are in Appendix C.

Finally, Figure 4 shows task-solving accuracy by puzzle difficulty under the Text Symbols representation. Accuracy decreases monotonically for all models as difficulty increases, indicating that perceptual support does not eliminate the need for nontrivial planning. The strongest models benefit most on easier and medium-difficulty instances, while performance drops sharply at levels 4–5 and the gap between models narrows. This suggests that scaffolding helps when the remaining planning problem is within the model's capacity, but harder puzzles still expose residual long-horizon planning and rule-reasoning limitations.

4.3 Effect of Visual Scaffolding on Perceptual Recognition

Next, we examine whether improvements in task-solving accuracy are accompanied by stronger perceptual recognition by asking models to identify the rule symbols present on the board. Figure 5 reports object-detection performance across board representations. Overall, scaffolded inputs improve rule recognition over the original visual board for most models. The effect is especially clear for stronger models. Gemma 4 31B improves from 85.4% average rule accuracy on the original board to 97.0% with Text Symbols, while Qwen 3.5 397B improves from 87.3% to 98.1% with Cell Coordinates. GLM 4.6V also shows improvements under coordinate-based variants, reaching 92.1% with Axis Labels + S/E.

The gains are not uniformly monotonic across all models and representations. For example, Qwen 3.5 27B shows lower detection accuracy with Axis Labels than with the original board, while Qwen 3.5 397B performs worse under Axis Labels + S/E than under the original board. This suggests that adding visual scaffolds can introduce clutter or change the visual distribution in ways that affect recognition. Nevertheless, the trend shows that representations with stronger spatial support improve perceptual access to board elements.

Figure 6 provides a finer-grained view through per-type confusion patterns for Qwen 3.5 397B. On the Original board, several structural elements are difficult to distinguish. Start and End are correctly detected only 69% and 48% of the time, respectively, with 17% of Start cells and 38% of End cells misclassified as Path. Gap is the most severe case: only 21% of Gap cells are correctly detected, while 65% are misclassified as Path. Text Symbols sharply reduces these confusions. Start, End, Path, and Gap all reach 100% detection accuracy, and most rule symbols also become nearly perfect, including Square, Star, Polyshape, and Neg. Poly at 100%, Triangle at 99%, and Dot at 97%. This shift supports the view that explicit symbolic rendering reduces ambiguity in the visual input and improves grounding of board components. This confusion pattern is not specific to Qwen 3.5 397B. An averaged confusion analysis across all seven models, reported in Appendix E.4,

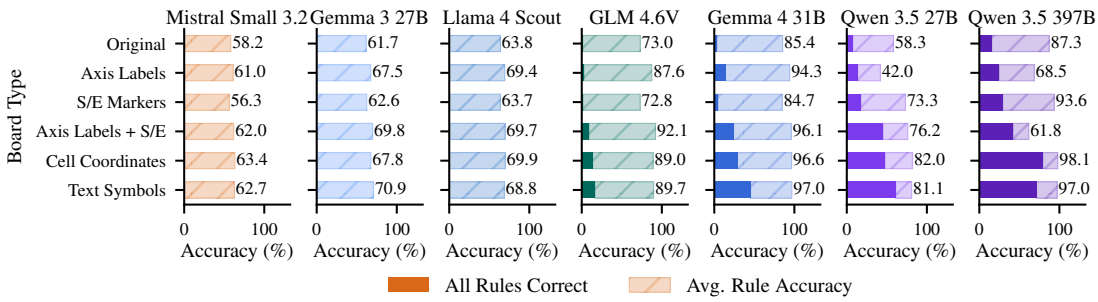


Figure 5: Object-detection performance across board representations. We report exact-match board accuracy and average rule detection accuracy for each model and representation.

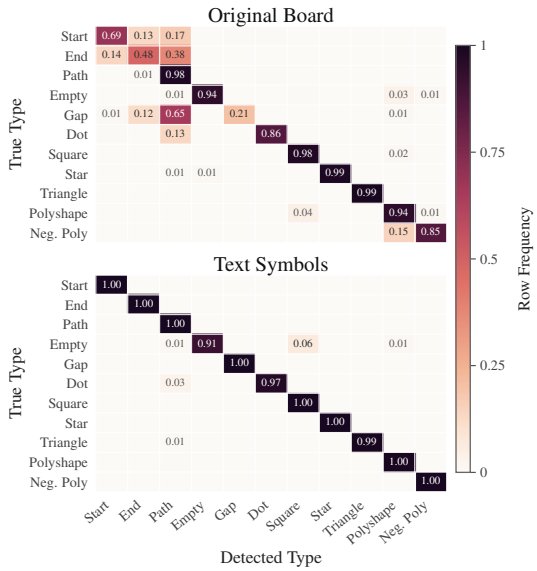


Figure 6: Per-type detection accuracy and confusion patterns for the original board and Text Symbols representation with Qwen 3.5 397B.

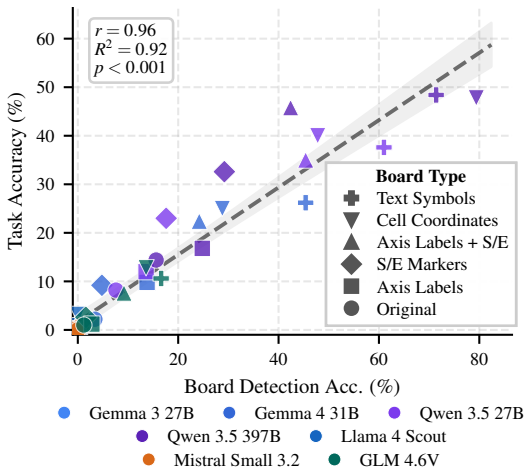


Figure 7: Correlation between object-detection performance and end-to-end task-solving accuracy across models and board representations.

shows the same collapse of symbol classes onto the dominant Path class on the Original board and the same diagonal sharpening under Text Symbols.

Finally, Figure 7 compares object-detection performance with end-to-end task-solving accuracy across models and board representations. The strong positive correlation ($r = 0.96$, $R^2 = 0.92$, $p < 0.001$) indicates that representations and models with better perceptual recognition also tend to achieve higher planning accuracy. This should not be read as proving that perception alone determines task success, since both detection and planning may depend on model capacity. Together with the hierarchy results in §4.2, it indicates that improved perceptual grounding is an important mediator of the scaffolding gains. Appendix E.5 repeats this correlation with Average Rule Detection Accuracy, where it remains significant but weaker ($r = 0.51$).

4.4 Path-Level Error Analysis

To understand how visual scaffolding affects model behavior, we analyze the generated paths using the validity metrics from §3.4. Figure 8 shows the results, averaged across all evaluated models.

The metrics most sensitive to scaffolding are **Correct Start/End**, **Connected Path**, and **Fully Valid**. On the Original board, only 11.8% of paths connect the right endpoints and 8.9% are Fully Valid. Cell Coordinates and Text Symbols push Correct Start/End to around 90% and Fully Valid to roughly 48%, while Connected Path moves more gently (68.2% to 76.3%). For comparison, the SPaRC text baseline reaches 97.2% endpoint correctness, 85.1% connectivity, and 46.7% Fully Valid, so the two strongest scaffolds reach comparable Fully Valid rates. This is a controlled reduction of the visual grounding burden, not a solution to the original visual task: Text Symbols shifts part of that burden toward reading in-image text, though it still preserves the

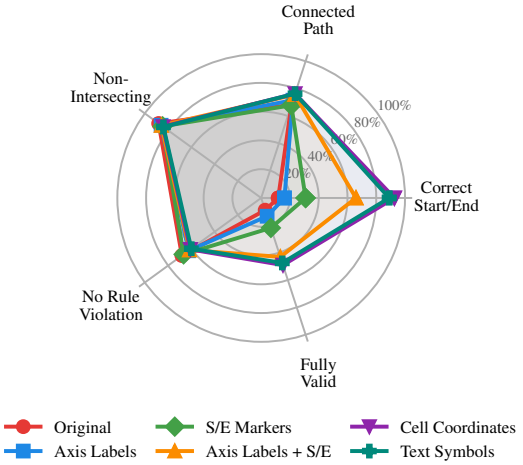


Figure 8: Path-level validity analysis for four representative input settings averaged across all models.

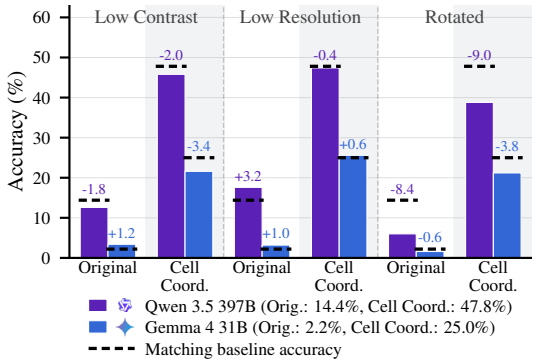


Figure 9: Task-solving accuracy under visual degradation and scaffolded recovery settings.

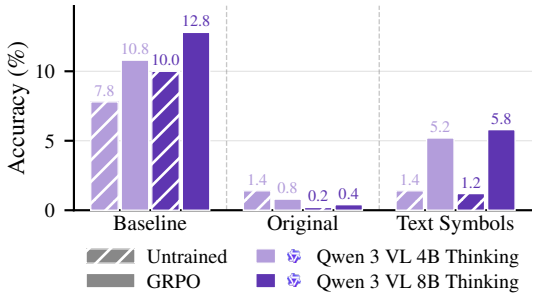


Figure 10: Effect of GRPO post-training on task-solving accuracy for Qwen 3 VL 4B and 8B Thinking, evaluated across three input representations.

spatial layout that the text formulation linearizes. Under the Original visual format, the dominant failure mode is unstable spatial grounding rather than planning.

Non-Intersecting stays high across all configurations (84.0–88.1%, against 89.8% on text) and barely shifts, so revisiting cells is not a meaningful failure mode. **No Rule Violation**, which checks whether the path stays off rule cells, sits between 60.0% and 68.1% across the visual variants and 52.9% on the text baseline, with no clear trend across the scaffolding hierarchy. The Fully Valid gains under scaffolding are therefore driven by improvements in endpoint and connectivity and not by changes in these two conditions.

4.5 Recovery from Visual Degradation

We next use the degradation-and-recovery setting from §3.3 as a stress test of the grounding hypothesis. If scaffolding restores access to the board's spatial structure, degraded inputs should reduce performance, while the corresponding Cell Coordinates variants should remain closer to their clean counterpart.

Figure 9 reports changes in task-solving accuracy relative to the clean Original and Cell Coordinates settings. Low Contrast and Low Resolution have limited effects under Cell Coordinates. For Qwen 3.5 397B, the recovered variants change by only -2.0 and -0.4 points, respectively; for Gemma 4 31B, the corresponding changes are -3.4 and $+0.6$ points. Rotation is more disruptive, with larger drops under Rotated + Cell Coordinates (-9.0 for Qwen 3.5 397B and -3.8 for Gemma 4 31B), suggesting that orientation changes introduce an additional spatial-alignment challenge.

A complementary analysis of object-detection accuracy under the same degradations (Appendix E.7) shows a similar pattern. Rule detection remains largely stable under Low Contrast and Low Resolution, but drops more clearly under Rotation, especially for the Cell Coordinates variants. Overall, Cell Coordinates preserve much of the performance under degradations that reduce salience or resolution, while rotation remains a harder case. This supports the view that explicit spatial scaffolding can recover impaired visual grounding, but does not fully eliminate orientation-sensitive spatial alignment errors.

4.6 Effect on GRPO Post-Training

To test how the input representation interacts with post-training, we apply GRPO to Qwen 3 VL 4B Thinking and 8B Thinking under three input conditions: the text-only **Baseline**, the unannotated **Original** image, and the fully scaffolded **Text Symbols** variant. More details about the setting

can be seen in Appendix E.3.

Figure 10 shows three contrasting patterns. On the text baseline, GRPO improves accuracy by 3.0 points on 4B (7.8% → 10.8%) and 2.8 points on 8B (10.0% → 12.8%), suggesting that the same training recipe is effective on a well-formed input. On the unannotated visual input, the same procedure is essentially ineffective: 4B accuracy moves from 1.4% to 0.8% and 8B from 0.2% to 0.4%, indicating that the policy cannot discover useful gradient signal from naive image observations. On the scaffolded visual input, however, GRPO yields the largest absolute improvements in our study, +3.8 points on 4B (1.4% → 5.2%) and +4.6 points on 8B (1.2% → 5.8%), both exceeding the corresponding text-baseline gains. Only the Text Symbols gains have disjoint pre- and post-training intervals (Appendix E.1); the smaller text-baseline gains are consistent in direction across both sizes.

The pattern suggests visual scaffolding therefore does more than improve zero-shot accuracy. It exposes task-relevant structure in a form that RL can exploit, avoiding both the compression of text linearization and the perceptual opacity of naive image rendering.

5 Conclusion

We investigated how visual presentation shapes VLM performance and failure modes when the underlying reasoning problem is fixed. Using SPaRC as a visual spatial planning testbed, we introduced a hierarchy of lightweight visual scaffolds that progressively increase perceptual support while preserving the same board layout, rule configuration, and target solution. Across multiple VLMs, these scaffolds substantially improve task-solving accuracy over the original visual board by up to 34.0 percentage points and further make GRPO post-training more effective.

Our analyses further clarify the source of these improvements. Object-detection and path-level validity results reveal that scaffolding mainly reduces grounding-related errors while rule satisfaction remains comparatively challenging. Thus, visual scaffolding separates grounding-sensitive errors from residual reasoning challenges.

These findings suggest that visual presentation can change both what VLMs can solve and what their failures measure. Future benchmarks for multimodal planning should control both the underlying reasoning problem and the perceptual interface through which that problem is presented.

Our empirical claims are restricted to SPaRC, and we do not claim that the gains or the scaffolds transfer to natural images, robotics, or document understanding. What is portable is the experimental logic: intervene on the input, hold the task fixed, and observe where the errors move. The residual errors point to directions for improvement, namely coordinate- or layout-aware visual encoders for the grounding-sensitive part of the gap, and planning-aware post-training or search-based decoding for the rule-satisfaction errors that persist under every scaffold.

Limitations

Our scaffolds are hand-designed from SPaRC's known board structure, so each makes explicit a requirement the task already imposes, and the improvements are partly by construction. They are diagnostic interventions rather than a general prompting strategy, and we do not study which scaffold is minimally sufficient or whether they can be built automatically. Because the steps are cumulative rather than orthogonal, we do not read the hierarchy as separating perception from planning, and we count in-image text reading as perception.

All results come from SPaRC, whose grid regularity is both what makes these scaffolds natural and what lets us hold the board layout, rule configuration, and target solution fixed while varying only presentation. We therefore restrict the empirical claims to this benchmark.

Our model pool covers seven open-weight VLMs from 24B to 397B parameters and spans both dense and Mixture-of-Experts architectures, but does not include closed frontier systems such as GPT-5, Gemini 3.1, or Claude, which could in principle respond differently to our scaffolding hierarchy. The trend is consistent across model families (Qwen, Gemma, Llama, GLM, Mistral) and over an order of magnitude in scale, suggesting that the sensitivity to visual presentation reflects current VLM design rather than a quirk of any single family. Using open weights also ensures reproducibility of the results.

Qwen 3.5 397B and GLM 4.6V are run with 4-bit weight quantization to fit our VRAM budget, which likely reduces absolute accuracy for both models. Our central claims are based on within-

model comparisons across input representations, and the quantization level is held constant per model. It therefore cannot account for the large gains in scaffolding we observe, including the 34.0 percentage point jump on Qwen 3.5 397B.

The GRPO experiments are limited to Qwen 3 VL 4B Thinking and Qwen 3 VL 8B Thinking since larger reasoning models exceeded our training budget. Consequently, we cannot directly characterize how RL post-training scales for the largest models. However, the pattern we observe is consistent across the two sizes that we trained on. There are near-zero GRPO gains on the Original board compared to up to 4.6 percentage points on Text Symbols. This comparison is the most important one for our claim that perceptual access constrains what RL post-training can teach.

Finally, accuracies are computed over 500 puzzles. The main scaffolding gains are resolvable against the bootstrap intervals in Appendix E.1, but smaller effects are not, including the GRPO gains on the text baseline (§4.6) and any gain for the two models at the accuracy floor.

Acknowledgments

This work was partially supported by the Lower Saxony Ministry of Science and Culture and the VW Foundation. This work used the Scientific Compute Cluster at GWDG, the joint data center of Max Planck Society for the Advancement of Science (MPG) and University of Göttingen. In part funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 405797229.

References

Zachary Abel, Jeffrey Bosboom, Michael Coulombe, Erik D. Demaine, Linus Hamilton, Adam Hesterberg, Justin Kopinsky, Jayson Lynch, Mikhail Rudoy, and Clemens Thielen. 2018. Who witnesses the witness? finding witnesses in the witness is hard and sometimes impossible. *ArXiv preprint*, abs/1804.10193.

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2024. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning.

Mohammad Asadi, Jack W O'Sullivan, Fang Cao, Tahoura Nedaee, Kamyar Fardi, Fei-Fei Li, Ehsan Adeli, and Euan Ashley. 2026. Mirage the illusion of visual understanding. *ArXiv preprint*, abs/2603.21687.

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

Jonathan Blow. 2016. The witness. [Online; accessed 15-May-2025].

Shiqi Chen, Jinghan Zhang, Tongyao Zhu, Wei Liu, Siyang Gao, Miao Xiong, Manling Li, and Junxian He. 2025a. Bring reason to vision: Understanding perception and reasoning through model merging. In *Forty-second International Conference on Machine Learning*.

Yan Chen, Long Li, Teng Xi, Long Zeng, and Jingdong Wang. 2025b. Perception before reasoning: Two-stage reinforcement learning for visual reasoning in vision-language models. *ArXiv preprint*, abs/2509.13031.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *ArXiv preprint*, abs/2507.06261.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report.

Gemma Team, Google DeepMind. 2026. Gemma 4: Byte for byte, the most capable open models. `https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/`.

Xingang Guo, Utkarsh Tyagi, Advait Gosai, Paula Vergara, Jayeon Park, Ernesto Gabriel Hernández Montoya, Chen Bo Calvin Zhang, Bin Hu, Yunzhong He, Bing Liu, and 1 others. 2025. Beyond seeing: Evaluating multimodal llms on tool-enabled image perception, transformation, and reasoning. *ArXiv preprint*, abs/2510.12712.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and XingYu. 2026. Deepeyesv2: Toward agentic multimodal model. In *The Fourteenth International Conference on Learning Representations*.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. 2024. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Team HY-World, Chenjie Cao, Xuhui Zuo, Zhenwei Wang, Yisu Zhang, Junta Wu, Zhenyang Liu, Yuning Gong, Yang Liu, Bo Yuan, and 1 others. 2026. Hy-world 2.0: A multi-modal world model for reconstructing, generating, and simulating 3d worlds. *ArXiv preprint*, abs/2604.14268.

Jitesh Jain, Zhengyuan Yang, Humphrey Shi, Jianfeng Gao, and Jianwei Yang. 2026. Elevating visual perception in multimodal LLMs with visual embedding distillation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Lars Benedikt Kaesberg, Jan Philip Wahle, Terry Ruas, and Bela Gipp. 2025. SPaRC: A spatial pathfinding reasoning challenge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10359–10390, Suzhou, China. Association for Computational Linguistics.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, and 1 others. 2026. Kimi k2. 5: Visual agentic intelligence. *ArXiv preprint*, abs/2602.02276.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. 2018. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Sachit Menon, Richard Zemel, and Carl Vondrick. 2024. Whiteboard-of-thought: Thinking step-by-step across modalities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20016–20031, Miami, Florida, USA. Association for Computational Linguistics.

Meta AI. 2025. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`.

Mistral AI. 2025. Mistral Small 3.2-24B-Instruct-2506. `https://huggingface.co/mistralai/Mistral-Small-3.2-24B-Instruct-2506`.

Qwen Team. 2026a. Qwen3.5-omni technical report. *ArXiv preprint*, abs/2604.15804.

Qwen Team. 2026b. Qwen3.5: Towards native multimodal agents.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Haz Sameen Shahgir, Xiaofu Chen, Yu Fu, Erfan Shayegani, Nael Abu-Ghazaleh, Yova Kementchedjhieva, and Yue Dong. 2026. Vlms need words: Vision language models ignore visual detail in favor of semantic anchors. *ArXiv preprint*, abs/2604.02486.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, abs/2402.03300.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2026. Openai gpt-5 system card. *ArXiv preprint*, abs/2601.03267.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 74 others. 2025. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning.

J. Wahle, T. Ruas, S. M. Mohammad, N. Meuschke, and B. Gipp. 2023. Ai usage cards: Responsibly reporting ai-generated content. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 282–284, Los Alamitos, CA, USA. IEEE Computer Society.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, and 1 others. 2025. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *ArXiv preprint*, abs/2508.18265.

Zhenhailong Wang, Xuehang Guo, Sofia Stoica, Haiyang Xu, Hongru WANG, Hyeonjeong Ha, Xiusi Chen, Yangyi Chen, Ming Yan, Fei Huang, and Heng Ji. 2026. Perception-aware policy optimization for multimodal reasoning. In *The Fourteenth International Conference on Learning Representations*.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. 2026. Reinforcing spatial reasoning in vision-language

models with interwoven thinking and visual drawing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Qiucheng Wu, Handong Zhao, Michael Saxon, Trung Bui, William Yang Wang, Yang Zhang, and Shiyu Chang. 2025. Vsp: Diagnosing the dual challenges of perception and reasoning in spatial planning tasks for mllms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2270–2280.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *ArXiv preprint*, abs/2310.11441.

Shuo Yang, Yuwei Niu, Yuyang Liu, Yang Ye, Bin Lin, and Li Yuan. 2026. Look-back: Implicit visual re-focusing in mllm reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 11694–11702.

Tianyu Yang, Terry Ruas, Yijun Tian, Jan Philip Wahle, Daniel Kurzawe, and Bela Gipp. 2025. Alden: Reinforcement learning for active navigation and evidence gathering in long documents. *ArXiv preprint*, abs/2510.25668.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2024. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403.

Juntian Zhang, Song Jin, Chuanqi Cheng, Yuhan Liu, Yankai Lin, Xun Zhang, Yufei zhang, Fei Jiang, Guojun Yin, Wei Lin, and Rui Yan. 2026. ViPER: Empowering the self-evolution of visual perception abilities in vision-language models. In *The Fourteenth International Conference on Learning Representations*.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and XingYu. 2026. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*.

A Model and Hardware Details

Hardware All inference and evaluation procedures, across both the baseline and visual scaffolding settings, were performed on a uniform hardware configuration of 4 NVIDIA A100 GPUs, each equipped with 80GB of VRAM. This provided sufficient memory and compute capacity to run most evaluated VLMs at full precision. For the two largest models, Qwen 3.5 397B and GLM 4.6V, we additionally apply 4-bit weight quantization to fit them within our VRAM budget while preserving their multimodal capabilities. All other models are evaluated without quantization. For GRPO training of the Qwen 3 VL 4B and 8B Thinking model, we used up to four nodes, each with four NVIDIA A100 GPUs.

Evaluated Models We evaluate seven vision-language models spanning different sizes, architectures, and training paradigms in order to assess how perceptual accessibility shapes downstream planning behavior across a representative cross-section of current open-weight VLMs. The selection covers both dense and Mixture-of-Experts (MoE) architectures, ranges from 24B-parameter models to a 397B MoE, and includes models from several major model families. The full list is given below:

- **Qwen 3.5 397B (4-bit)**: A large-scale Mixture-of-Experts vision-language model from Alibaba Cloud's Qwen 3.5 generation. We apply 4-bit weight quantization to enable inference within our hardware budget while retaining its multimodal reasoning capability (Qwen Team, 2026b).
- **Qwen 3.5 27B**: A mid-sized dense vision-language model from the same Qwen 3.5 family, evaluated in full precision and serving as a reference point for the quantized 397B variant within the same model family (Qwen Team, 2026b).
- **Gemma 4 31B**: Google's latest open-weight multimodal model in the Gemma series, building on Gemini technology with improved visual grounding and long-context reasoning over its predecessor (Gemma Team, Google DeepMind, 2026).
- **GLM 4.6V (4-bit)**: Zhipu AI's vision-language model from the GLM 4.6 series, designed for multimodal understanding tasks. We apply 4-bit weight quantization to fit it within our VRAM budget (V Team et al., 2025).
- **Llama 4 Scout**: Meta's natively multimodal Mixture-of-Experts model from the Llama 4 family, with 17B active parameters and a long context window, designed as the efficient tier of the Llama 4 release (Meta AI, 2025).
- **Gemma 3 27B**: Google's previous-generation open-weight multimodal model,

built on Gemini technology and designed for long-context multimodal reasoning. It serves as a within-family comparison point against Gemma 4 31B (Gemma Team et al., 2025).

- **Mistral Small 3.2**: A 24B-parameter multimodal model from Mistral AI in the Mistral Small line, included as a smaller-scale dense vision-language baseline (Mistral AI, 2025).

Together, these seven models allow us to study how the effect of input-side visual scaffolding varies with model scale, architecture, and training paradigm, while keeping the underlying planning task fixed.

Compute Budget Table 1 reports the aggregate inference cost for the seven evaluated VLMs, summed over all controlled board representations, prompt variants, the visual-degradation runs, and the object-detection evaluations. Token counts are taken from the per-run `token_usage` fields emitted by our evaluation pipeline; wall-clock hours are taken from the corresponding `evaluation_duration_seconds` entries. A100-hours are reported as wall-clock hours multiplied by four (the fixed hardware configuration described above).

Completion tokens account for roughly 88% of the total, reflecting the long reasoning chains produced by the Qwen 3.5 and GLM 4.6V models on the SPaRC puzzles. Qwen 3.5 397B is the dominant cost contributor at $\approx$ 720 A100-hours, both because it participates in the additional worsening-study runs and because its quantized inference is the slowest per-token. By contrast, Llama 4 Scout terminates quickly with very short outputs and contributes only $\approx$ 4 A100-hours despite being run on the same set of board representations. The reported numbers cover inference only; GRPO training of Qwen 3 VL 4B / 8B Thinking is accounted for separately in Appendix E.3.

B SPaRC Dataset

SPaRC (Kaesberg et al., 2025) consists of 1,000 2D grid pathfinding puzzles (500 train, 500 test) inspired by the puzzle mechanics of *The Witness* (Blow, 2016). Each puzzle is an $m \times n$ grid of **rule cells** with $(x, y) = (0, 0)$ at the top-left corner, x increasing rightward and y downward. Rule cells are surrounded by **edges** along which the solution path is drawn. Each puzzle has exactly one **start point** (large circle) and one **end point** (edge extension). The goal is to draw a single, continuous, non-self-intersecting path along edges from start to end that satisfies all rule cell constraints.

B.1 Rules

Seven rule types can appear in SPaRC puzzles:

Item Collection (Dots): The solution path needs to pass through every dot.

Path Breaks (Gaps): The solution path cannot go through any edge segment containing a gap. Gaps act as local barriers.

Color Separation (Stones): The solution path must be drawn to separate stones of different colors. All stones located within any single enclosed region must be of the same color.

Pairing (Stars): Each star must share its region with exactly one other symbol of the same color. No unpaired stars are allowed.

Edge Count (Triangles): The solution path must touch the number of edges shown by the triangles in the cell, e.g., two triangles mean the path must touch exactly two edges of that cell.

Shape Fitting (Polyominoes): If a cell contains a polyomino (poly), the solution path must enclose a region that matches its exact shape and area. The region must not rotate or mirror the poly. Multiple polys can share a region if their shapes fit without overlapping.

Shape Subtraction (Ylop): A ylop must be enclosed in the same region as one or more polys. Its shape and area subtract from the total required by the polys. If a ylop cancels out a poly exactly, that pair imposes no constraint.

B.2 Dataset Creation

Puzzles are generated by randomly creating an $m \times n$ grid ($m, n \in [2, 6]$), filling approximately half the cells with rules (*rule density*), and placing random start and end points. A generation-validation loop solves each puzzle via brute-force search over all valid paths.[2] If no solution exists, rule density is decreased; if more than k=50 solutions exist, density is increased, and the puzzle is regenerated.

[2] Brute-force is necessary because many puzzles fall into NP or NP-Complete complexity classes (Abel et al., 2018).

Model	Prompt (M)	Completion (M)	Total (M)	Wall (h)	A100-h
Mistral Small 3.2	12.1	15.2	27.3	7.9	31.7
Gemma 3 27B	10.5	4.5	15.0	2.6	10.4
Llama 4 Scout	14.6	4.6	19.2	1.1	4.2
GLM 4.6V (4-bit)	11.7	65.1	76.8	30.9	123.8
Gemma 4 31B	21.0	38.5	59.5	16.7	66.9
Qwen 3.5 27B	11.2	215.4	226.6	102.6	410.3
Qwen 3.5 397B (4-bit)	29.0	489.8	518.7	180.0	719.8
Total	**110.0**	**833.1**	**943.1**	**341.8**	**1,367.1**

Table 1: Per-model token usage and compute consumed on 4×A100 (80 GB). Token counts are in millions; wall-clock and A100-hours are aggregated across all evaluation runs for that model (controlled boards, prompt variants, visual degradations, and object-detection sub-tasks).

Difficulty estimation. Puzzle complexity is quantified by a weighted sum of the number of distinct rule types, total rule cells, rule density, grid size, and estimated rule interactions, normalized to a 1 (easiest) to 5 (hardest) scale. The test set contains 86 puzzles at level 1, 118 at level 2, 121 at level 3, 86 at level 4, and 89 at level 5.

C Comparison with the SPaRC Text Baseline

Figure 11 reports accuracy on the original SPaRC text-formulation baseline for human solvers from Kaesberg et al. (2025) and the seven evaluated VLMs. Humans reach 98%. The strongest VLM (Qwen 3.5 397B) reaches 38.0%, and four of seven models score below 10%. The human-model gap on a purely textual presentation shows that SPaRC is non-trivial independent of visual encoding.

Comparing against the best scaffolded visual result per model from §4.2: Qwen 3.5 397B improves from 38.0% to 48.4% (+10.4 pp, Text Symbols), Gemma 4 31B from 16.6% to 26.2% (+9.6 pp, Text Symbols), Qwen 3.5 27B from 33.9% to 40.0% (+6.1 pp, Cell Coordinates), and GLM 4.6V from 8.4% to 12.8% (+4.4 pp, Cell Coordinates). The three weakest models (Llama 4 Scout, Gemma 3 27B, Mistral Small 3.2) stay within ±1 pp of their text baseline, all near the floor. The visual-over-text gain scales with model capability, matching the scaffolding-gain pattern in §4.2.

D Recovery Variants

Figure 12 shows the three visual degradation variants introduced in Section 3.3 alongside their recovery counterparts. Each recovery variant overlays the Cell Coordinates representation on the degraded board, leaving the planning task and degradation parameters unchanged. This isolates the contribution of explicit spatial indexing from any change to the visual signal itself.

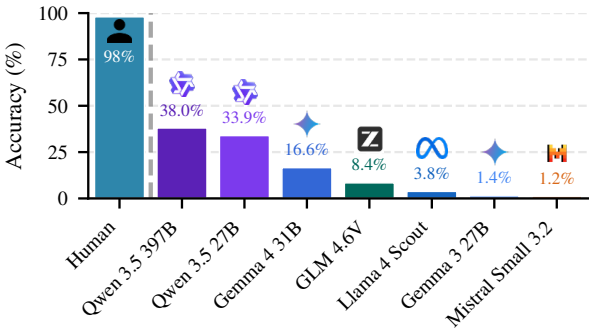


Figure 11: Accuracy on the SPaRC text-formulation baseline for human solvers from Kaesberg et al. (2025) and the seven evaluated VLMs.

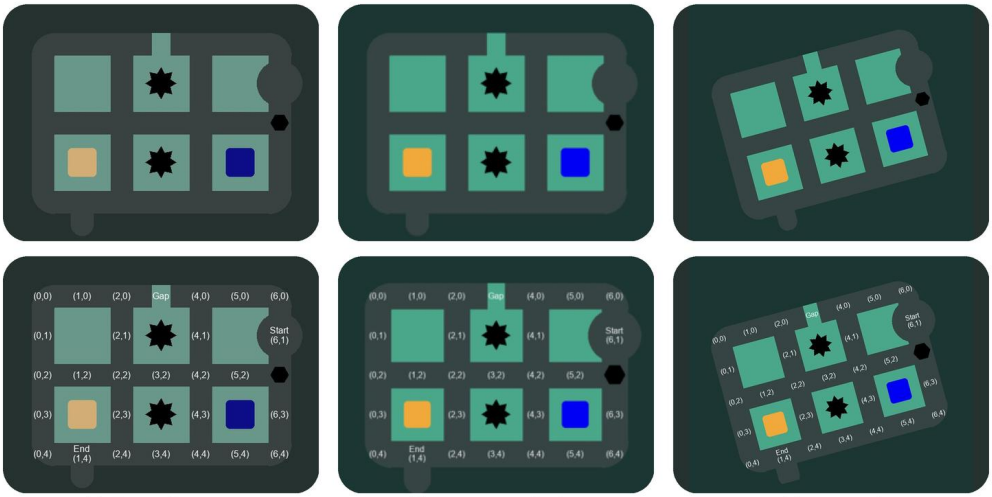
E Additional Experiments

E.1 Bootstrap Confidence Intervals

Accuracies are computed over 500 test puzzles, so we report 95% bootstrap percentile intervals obtained by resampling puzzle IDs with replacement (200,000 replicates, seed 20260821). Table 2 covers every model and condition and Table 4 the GRPO runs. Table 3 reports differences to each model's best scaffold, where each replicate resamples puzzle IDs once and evaluates both conditions on that same resample. The gains are resolvable for the five models above the accuracy floor. Gemma 3 27B and Mistral Small 3.2 stay below 1% on every visual condition and show no resolvable gain from any scaffold.

E.2 Grid Lines versus Coordinate Labels

The Axis Labels scaffold changes two things at once. It draws grid lines over the board, and it prints coordinate labels in a margin around it. To



(a) Low Contrast (b) Low Resolution (c) Rotated

Figure 12: The three visual degradations (low contrast, low resolution, and rotation) shown above their recovery counterparts, in which Cell Coordinates are overlaid on the degraded board.

separate visual structure from the labels themselves, we decompose it into three variants on the same 500 test puzzles: grid lines only, drawn with equal margins on all four sides; grid lines with the label margin reserved but left blank; and the full Axis Labels variant.

Table 5 reports task-solving accuracy for the three strongest models. Grid lines alone give small gains over the Original board (+1.4, +0.2, and +2.6 pp). Reserving the margin but leaving it blank hurts every model, falling below the Original board in all three cases. Printing the labels into the same margin recovers and exceeds it (+2.4 to +7.6 pp).

An error analysis explains the blank-margin result. We measure the rate of ± 1 coordinate offsets in the predicted start and end nodes, that is, predictions naming a cell one step away from the correct one along either axis. This rate is 4–5% with explicit labels and rises to 17–46% with the blank margin. Removing the margin and keeping only the grid lines brings it back to 6–14% for the two Qwen models, while Gemma 4 31B stays shift-prone at 35%.

An uninformative reserved margin therefore disrupts cell indexing, since the model appears to treat the blank band as part of the board and offsets its indices accordingly. Explicit labels instead act as coordinate anchors, and they matter most for the weaker models. The gains come from explicit spatial grounding cues rather than from generic visual modification.

E.3 GRPO Post-Training

In addition to the input-side interventions in (§3.2) and (§3.3), we study a weight-side intervention. We apply Group Relative Policy Optimization (GRPO) to Qwen 3 VL 4B Thinking and Qwen 3 VL 8B Thinking, two open-weight reasoning models that fit our hardware budget for training. Training uses the SPaRC train split (500 examples), and evaluation uses the held-out test split. The reward is based on whether the predicted path satisfies all validity conditions defined in (§3.4). Hyperparameters and compute details are provided in Appendix A. To isolate the role of perceptual access in trainability, we evaluate GRPO under three input conditions, namely the SPaRC text baseline, the Original visual board, and the Text Symbols scaffold. For each condition, the model is trained and evaluated on the same input representation, so any difference in GRPO gain across conditions reflects how the input format affects the training signal. We use this comparison as a diagnostic for how perceptual access interacts with RL post-training.

GRPO training of the Qwen 3 VL 4B and 8B Thinking models was run on up to four 4×A100 nodes (16 GPUs in total) and required approximately 30 h and 60 h of wall-clock time, respectively. This corresponds to on the order of 500 and 1,000 A100-hours, bringing the combined GRPO training budget to roughly 1,500 A100-hours on top of the inference cost reported in Table 1.

Model	Setting	Acc.	95% CI
Gemma 3 27B	Original	0.0	[0.0, 0.0]
	S/E	0.0	[0.0, 0.0]
	Axis	0.0	[0.0, 0.0]
	Axis+S/E	0.0	[0.0, 0.0]
	Cell Coord.	0.0	[0.0, 0.0]
	Text Sym.	0.6	[0.0, 1.4]
	Text base.	1.4	[0.4, 2.6]
Gemma 4 31B	Original	2.2	[1.0, 3.6]
	S/E	9.2	[6.8, 11.8]
	Axis	9.8	[7.2, 12.4]
	Axis+S/E	22.4	[18.8, 26.2]
	Cell Coord.	25.0	[21.2, 28.8]
	Text Sym.	26.2	[22.4, 30.2]
	Text base.	16.6	[13.4, 20.0]
Qwen 3.5 27B	Original	8.2	[5.8, 10.6]
	S/E	23.0	[19.4, 26.8]
	Axis	12.0	[9.2, 15.0]
	Axis+S/E	35.0	[30.8, 39.2]
	Cell Coord.	40.0	[35.8, 44.4]
	Text Sym.	37.6	[33.4, 41.8]
	Text base.	33.9	[29.7, 38.2]
Qwen 3.5 397B	Original	14.4	[11.4, 17.6]
	S/E	32.6	[28.6, 36.8]
	Axis	16.8	[13.6, 20.2]
	Axis+S/E	45.8	[41.4, 50.2]
	Cell Coord.	47.8	[43.4, 52.2]
	Text Sym.	48.4	[44.0, 52.8]
	Text base.	38.0	[33.8, 42.2]
Llama 4 Scout	Original	0.0	[0.0, 0.0]
	S/E	0.4	[0.0, 1.0]
	Axis	0.2	[0.0, 0.6]
	Axis+S/E	0.6	[0.0, 1.4]
	Cell Coord.	3.2	[1.8, 4.8]
	Text Sym.	2.0	[0.8, 3.4]
	Text base.	3.8	[2.2, 5.6]
Mistral Small 3.2	Original	0.0	[0.0, 0.0]
	S/E	0.4	[0.0, 1.0]
	Axis	0.0	[0.0, 0.0]
	Axis+S/E	0.4	[0.0, 1.0]
	Cell Coord.	0.2	[0.0, 0.6]
	Text Sym.	0.4	[0.0, 1.0]
	Text base.	1.2	[0.4, 2.2]
GLM 4.6V	Original	1.0	[0.2, 2.0]
	S/E	2.6	[1.4, 4.0]
	Axis	1.2	[0.4, 2.2]
	Axis+S/E	7.6	[5.4, 10.0]
	Cell Coord.	12.8	[10.0, 15.8]
	Text Sym.	10.6	[8.0, 13.4]
	Text base.	8.4	[6.0, 10.8]

Table 2: Task accuracy (%) with 95% bootstrap confidence intervals over the test split ($B = 200{,}000$). *Text base.* is the non-visual text formulation; *Text Sym.* renders rule symbols as words.

Model	from Original	from Text base.
Mistral 3.2	+0.4 [0.0, 1.0]	−0.8 [−1.8, 0.0]
Gemma 3 27B	+0.6 [0.0, 1.4]	−0.8 [−2.0, 0.2]
Llama 4	+3.2 [1.8, 4.8]	−0.6 [−2.2, 1.0]
GLM 4.6V	+11.8 [9.0, 14.8]	+4.4 [1.4, 7.4]
Gemma 4 31B	+24.0 [20.2, 27.8]	+9.6 [5.8, 13.4]
Qwen 27B	+31.8 [27.6, 36.0]	+6.2 [2.2, 10.2]
Qwen 397B	+34.0 [29.4, 38.6]	+10.4 [6.2, 14.6]

Table 3: Paired differences (percentage points) from the Original board and from the text baseline to each model's best scaffold.

Setting	Before	After
Qwen 3 VL 4B		
Text base.	7.8 [5.6,10.2]	10.8 [8.2,13.6]
Original	1.4 [0.4, 2.6]	0.8 [0.2, 1.6]
Text Sym.	1.4 [0.4, 2.6]	5.2 [3.4, 7.2]
Qwen 3 VL 8B		
Text base.	10.0 [7.4,12.8]	12.8 [10.0,15.8]
Original	0.2 [0.0, 0.6]	0.4 [0.0, 1.0]
Text Sym.	1.2 [0.4, 2.2]	5.8 [3.8, 8.0]

Table 4: Task accuracy (%) before and after GRPO post-training, with 95% bootstrap confidence intervals.

E.4 Averaged Confusion Matrix

Figure 6 in §4.3 shows per-type confusion patterns for the strongest model, Qwen 3.5 397B. To check that those patterns are not specific to a single model, Figure 13 reports the same analysis averaged across all seven evaluated VLMs. On the Original board, all symbol types are heavily confused with the dominant Path class: diagonal correctness drops as low as 0.23 for End, 0.29 for Negative Polyshape, and 0.37 for Dot, with most of the mass on the Path column. The Text Symbols representation produces a substantially sharper diagonal across models. A residual fraction of cells is still labeled Path or Empty (∼0.20–0.34 for several rule types), which is consistent with weaker models defaulting to background classes even when symbols are made explicit; the dominant gain, however, comes from disambiguating the symbol classes themselves.

E.5 Correlation with Average Rule Detection Accuracy

In §4.3 (Figure 7) we report the correlation between Exact-Match Board Accuracy and end-to-end task accuracy. As a complement, Figure 14 shows the same correlation using Average Rule Detection Accuracy as the perception metric. The relationship remains positive and statistically significant ($r = 0.51$, $R^2 = 0.26$, $p < 0.001$), but is notably

Model	Orig.	+grid	+gutter	+labels
Qwen 3.5 397B	14.4	15.8	4.4	**16.8**
Qwen 3.5 27B	8.2	8.4	2.2	**12.0**
Gemma 4 31B	2.2	4.8	1.6	**9.8**

Table 5: Decomposition of the Axis Labels scaffold. Task-solving accuracy (%) on the 500-puzzle test split for the Original board, grid lines only (equal margins, no labels), grid lines with the label gutter reserved but left empty, and the full Axis Labels variant.

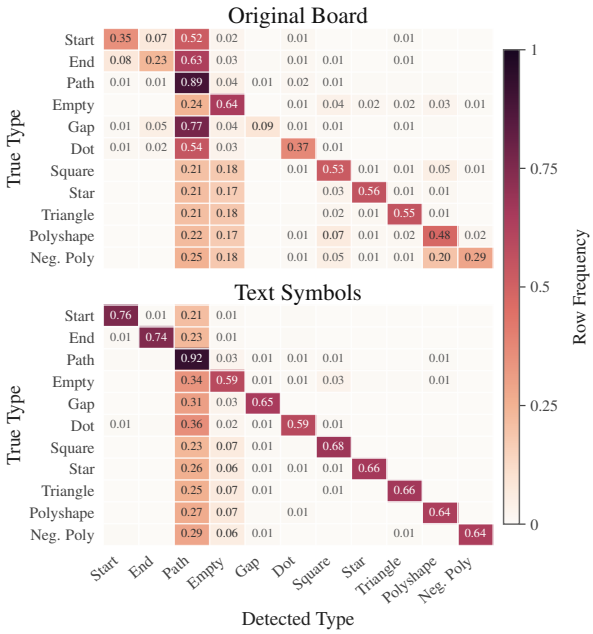


Figure 13: Per-type detection accuracy and confusion patterns averaged across all seven evaluated models, for the Original board (top) and Text Symbols representation (bottom).

weaker than the exact-match correlation ($r = 0.96$). This contrast is informative: solving a puzzle requires all task-relevant board elements to be perceived correctly, so a strict, board-level perception measure aligns more tightly with downstream success than per-rule averages, which are dominated by the many Empty and Path cells and can stay high even when a few critical symbols are mislabelled.

E.6 Token Usage

Figure 15 reports the average number of completion tokens generated per puzzle, stratified by difficulty level. Inference budgets vary by more than an order of magnitude across model families. The two Qwen 3.5 models scale their completion length with difficulty, rising from ∼31k to ∼50k tokens for Qwen 3.5 27B and from ∼24k to ∼36k for Qwen 3.5 397B. GLM 4.6V uses a

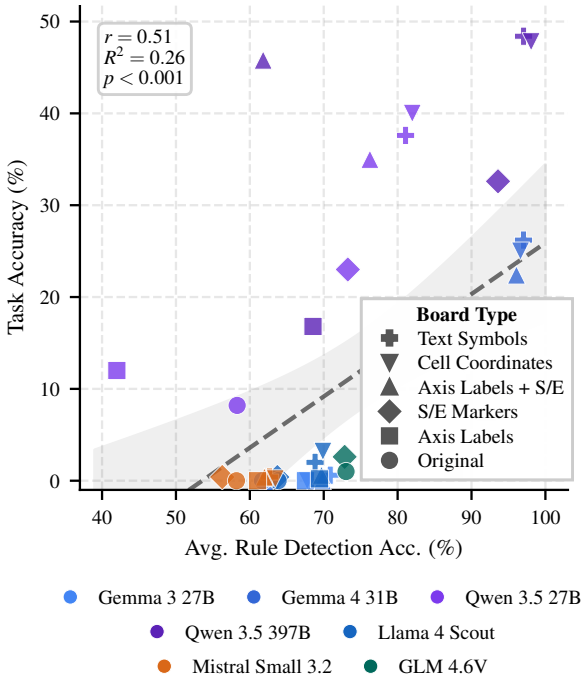


Figure 14: Correlation between average rule detection accuracy and end-to-end task-solving accuracy across models and board representations.

moderate and roughly constant budget (∼14k–17k tokens), while Mistral Small 3.2, Gemma 3 27B, Gemma 4 31B, and Llama 4 Scout stay below ∼6k tokens at all difficulties and do not visibly scale with difficulty.

Two observations connect this picture to the scaffolding results in §4.2. First, the two models that already invest the largest compute budgets are also the ones with the largest scaffolding gains (+31.8 pp for Qwen 3.5 27B and +34.0 pp for Qwen 3.5 397B). Even extensive chain-of-thought reasoning does not let these models overcome the grounding difficulty of the Original board; once spatial structure is made explicit, the same reasoning capacity translates into substantially better task accuracy. Second, Gemma 4 31B achieves a comparable +24.0 pp gain with completion budgets below ∼4k tokens. This shows that the benefit of scaffolding does not require extended reasoning. Models with short responses can still benefit from improved visual representations to produce correct solutions. Together, these patterns indicate that perceptual access acts upstream of inference-time compute: improving it benefits both long-reasoning and short-reasoning models, and it is not a substitute for reasoning capacity.

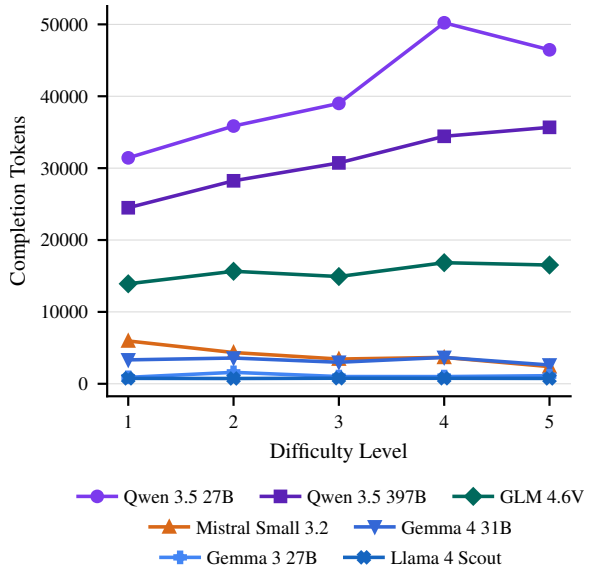


Figure 15: Completion tokens per puzzle, stratified by difficulty level, for each evaluated model on the Original visual input.

E.7 Object Detection under Visual Degradation

Figure 16 reports the change in Full Board Rule Detection Accuracy under the three visual degradation settings introduced in §4.5, both on the Original board and after overlaying Cell Coordinates, for the two strongest models (Qwen 3.5 397B and Gemma 4 31B). Two patterns stand out. First, low contrast and low resolution have only a marginal effect on rule detection on the Original board (all $|\Delta| \leq 6.2$ percentage points), in line with the similarly small changes in end-to-end accuracy under the same degradations in Figure 9. Symbol perception is therefore largely robust to the contrast and resolution reductions considered here. Second, rotation reduces rule detection substantially, and the Cell Coordinates overlay degrades sharply on rotated boards (-33.4 pp for Qwen 3.5 397B, -16.2 pp for Gemma 4 31B); a similar but smaller drop appears under low resolution with Cell Coordinates for Qwen 3.5 397B (-10.8 pp). A degraded or rotated coordinate overlay is itself misaligned with the model's canonical reading order and introduces additional visual clutter that competes with, rather than supports, rule recognition. These rule-detection drops on the Cell Coord overlay markedly exceed the corresponding end-to-end accuracy drops (e.g. -9.0 pp end-to-end vs. -33.4 pp rule detection for Qwen 3.5 397B under rotation), indicating that the scaffolding's contribution to rule-level perception is the component most affected by geometric transformations, even where downstream reasoning partially compensates. Together, these observations refine the interpretation from §4.4. Cell-level scaffolding primarily supports spatial grounding, and the scaffold itself becomes a liability when the board is rotated.

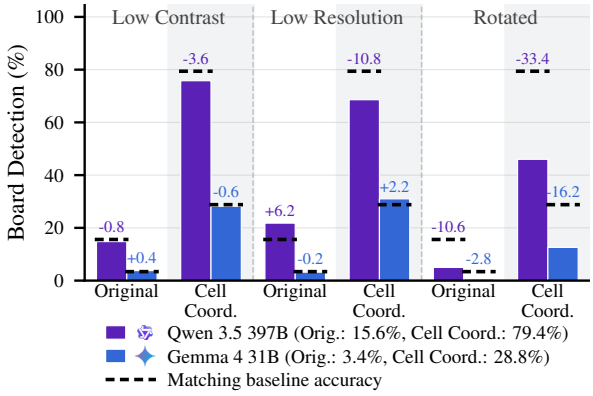


Figure 16: Object-detection accuracy under visual degradation and scaffolded recovery settings.

F Prompts

This section lists the three prompts used in the image-grounded evaluation. All three are presented to the model together with a rendered image of the puzzle. The *default* prompt (Appendix F.1) is a compact baseline that lists the rules and the expected output format. The *improved* prompt (Appendix F.2) extends the default with explicit definitions of path cells, rule cells, and regions, and adds worked examples to clarify region detection. The *object-detection* prompt (Appendix F.3) replaces the solving task with a perception-only task: the model must reconstruct the symbolic grid representation of the puzzle from the rendered image, which lets us isolate visual grounding errors from path-finding errors.

The object-detection prompt embeds the full symbol legend and color codes; the default and improved prompts rely on the rendered image plus the descriptions given in their listings. Curly-brace tokens (e.g. `{cols}`, `{rows}`) are placeholders filled in at runtime: `{cols}` and `{rows}` expand to $2W+1$ and $2H+1$ for a puzzle with a $W \times H$ grid of rule cells, while `{max_x}` and `{max_y}` expand to $2W$ and $2H$.

F.1 Default Prompt

The default prompt is the baseline image-grounded prompt. It describes the goal, the coordinate system, and the puzzle rules in a compact form, and instructs the model to output the solution as a list of `(x,y)` coordinates after a `####` marker. The block `{text_visualization}` is replaced by a textual rendering of the puzzle when the textual-representation flag is enabled, and is empty otherwise.

```
You are an expert spatial reasoning AI specializing in
      solving puzzles from the game 'The Witness'.
Your task is to solve the puzzle in the image by finding a
      valid line from the Start Node to the End Node.

The image shows a Witness puzzle grid of size
      {cols}x{rows}. In this puzzle:
- The solution is a continuous line from the start circle
      to the end marker
- The line travels along grid edges, connecting adjacent
      nodes horizontally or vertically
- The line cannot visit the same node twice
- The line must satisfy all constraints represented by the
      symbols on the grid
- The line can not be placed on rule cells
- The line can only travel 1 cell per step (no diagonal
      moves and provide each step as a separate coordinate)

COORDINATE SYSTEM:
- Nodes are indexed (x, y) where (0,0) is the top-left node
- x increases to the right, y increases downward
- The grid cells have rule symbols located at cells with
      all odd coordinates
- The line goes AROUND cells containing rules, forming
      boundaries
- Both line and rule cells are on the same grid. Therefore
      each intersection has a distance of 2 to the next
      intersection.

SOLVING RULES:
1. Draw a continuous line from the START NODE (big circle
      on the line) to the END NODE (rounded end) without
      visiting the same node twice.
2. The line can only be placed on valid path cells.
3. The line acts as a boundary, potentially dividing the
      grid cells into one or more distinct regions.
4. All rules associated with symbols on the grid must be
      satisfied:
   - Dots: The line MUST pass through each dot.
   - Colored squares: All squares within a single region
      created by the line must be the same color. Different
      colored squares MUST be separated into different
      regions by the line.
   - Colored stars: Each star must be paired with EXACTLY
      one other element of the same color in a region.
      Other colors are ignored.
   - Triangles: The line must touch EXACTLY the number of
      edges specified by the number of triangles in that
      cell (edges are top, right, bottom, left of the cell).
   - Tetris-like polyomino shapes: The region containing
      this symbol must be shaped EXACTLY like the defined
      polyshape.
   - Negative polyshapes: These cancel out regular
      polyshapes if they overlap.

Text description of the puzzle:
{text_visualization}

Analyze the puzzle image carefully and determine the
      solution path.
First, explain your reasoning step-by-step, including key
      deductions and constraint checks made along the way.
Then, provide the final solution as a sequence of node
      coordinates in (x, y) format, starting with the start
      node and ending with the end node, after this string:
      "####". DON'T SKIP ANY intermediate nodes (the
      distance between each node must be 1).
Example coordinate list: [(0,0), (1,0), (2,0), (2,1), ...]
```

F.2 Improved Prompt

The improved prompt expands the default with explicit definitions of path cells, rule cells, and regions; it spells out the parity convention for rule-cell coordinates and includes a worked example for a non-trivial region. It also separates positive and negative polyshapes and gives the precise output format.

Board variants. The placeholders `{start_marker}` and `{end_marker}` are replaced by the strings “ with an ‘S’ on it” and “, indicated by the letter ‘E”’ respectively when the board configuration includes start/end markers (i.e. `start_end_marked` appears in the `board_type`), and are empty otherwise.

```
## Objective
You are a specialized AI proficient in spatial reasoning
      and solving puzzles from the game 'The Witness'. Your
      goal is to find a valid path (a continuous line) from
      the specified Start Node to the End Node on the
      provided image, adhering to all puzzle rules.

## Core Concepts & Grid Basics
* **Grid Dimensions:** The puzzle grid has {cols} columns
      and {rows} rows.
* **Coordinate System:** Nodes are identified by `(x, y)`
      coordinates. `(0,0)` is the top-left node.
      `({max_x},{max_y})` is the bottom-right node. `x`
      increases to the right, `y` increases downwards.
      Nodes can either be path cells or rule cells.
* **Path Cells:** Path cells are dark grey and are at all
      positions where at least one of `x` or `y` is even.
* **Path:** The solution is a single, continuous line
      connecting adjacent path cells either horizontally or
      vertically. Each step has a distance 1.
* **No Revisits:** The path **CANNOT** visit the same node
      more than once.
* **Valid Path Cells:** The path travels along the grid
      lines (edges between nodes). It can only occupy
      positions on the path (these correspond to positions
      with at least one even coordinate and no gap).
* **Rule Cells:** Light green cells are rule cells and have
      coordinates where both `x` and `y` are odd. The path
      goes *around* these rule cells, never *on* them. Rule
      cells can contain rule symbols (square, star,
      triangles, polyshapes) but can also be empty (no
      constraint).
* **Regions:** The drawn path (in combination with the grid
      border) divides the rule cells into one or more
      distinct enclosed areas (regions). Many rules apply
      based on the contents of these regions.

## Detailed Solving Rules
The drawn path must satisfy **ALL** applicable constraints:

1. **Path Constraints:**
* Path **MUST** start at `Start Node` (big circle on the
      line in the same color as the grey
      path{start_marker}) and end at `End Node` (node from
      which you can escape the grid to the rounded path
      outside{end_marker}). Both these nodes are guaranteed
      to be on valid path cells on the edge of the board
      (at least one of `x` and `y` has to be an even
      number).
* Path connects adjacent nodes (horizontal/vertical moves
      only).
* Nodes **CANNOT** be revisited.
* Path **MUST** pass through all Dots (black hexagons) that
      lay on the path cells.
* Path **CANNOT** pass through any Gap in the path cells (a
      Gap is the absence of a path at a cell which would
      normally be a path cell).

2. **Region-Based Rules** (Apply to areas enclosed by the
      path):
```

```
* **Squares:** All squares within a single region **MUST**
      be the same color. Squares of different colors
      **MUST** be separated into different regions by the
      path.
* **Stars:** Within a single region, each star symbol
      **MUST** be paired with exactly **ONE** other element
      (star or square) *of the same color*. Other colors
      within the region are irrelevant to this specific
      star's rule.
* **Polyshapes** (one or multiple filled squares in a
      specific arrangement): The region containing this
      symbol **MUST** be in the specified shape (defined by
      the specific Polyshape arrangement). The shape must
      fit entirely within the region's boundaries. If
      multiple positive polyshapes are in one region, the
      region must accommodate their combined,
      non-overlapping forms.
* **Negative Polyshapes** (one or multiple unfilled squares
      in a specific arrangement): These "subtract" shape
      requirements, typically within the same region as
      corresponding positive polyshapes. A negative
      polyshape cancels out a positive polyshape of the
      exact same shape and color within that region. If all
      positive shapes are canceled, the region has no shape
      constraint. A negative shape is only considered
      'used' if it cancels a positive one. Negative shapes
      can sometimes rationalize apparent overlaps or
      boundary violations of positive shapes if interpreted
      as cancellations.
* **Simple Example Region:** If the a path goes through
      (2,0), (2,1), (2,2), (1,2), (0,2), then the rule cell
      (1,1) is enclosed into a region containing only
      itself since the grid border does the rest of the
      enclosure. To double check: There exist no nonvisited
      path cells (legal or non legal) that connect (1,1) to
      any other rule cell.
* **Complex Example Region:** Suppose the grid is 7x7 with
      rule cells at (1,1), (1,3), (1,5), (3,1), (3,3),
      (3,5), (5,1), (5,3), (5,5) and suppose one would want
      to form a region in an L-shape containing (1,1),
      (1,3), (3,3), then a path to separate them from the
      rest of the rule cells would for example be (4,0),
      (4,1), (4,2), (3,2), (2,2), (2,3), (2,4), (1,4),
      (0,4). The rest of the path cells around them at the
      edge don't have to be visited since they are only on
      the outside (x is (0 or width - 1) OR (y is (0 or
      height - 1)) and therefore not needed to seperate
      regions.

3. **Path-Based Rules (Edge Touching):**
* **Triangles** (one, two, three or four): The path
      **MUST** touch a specific number of edges of the cell
      containing the triangle symbol.
   * One Triangle: Path touches **EXACTLY 1** edge of the
      triangle's cell.
   * Two Triangle: Path touches **EXACTLY 2** edges of the
      triangle's cell.
   * Three Triangle: Path touches **EXACTLY 3** edges of
      the triangle's cell.
   * Four Triangle: Path touches **EXACTLY 4** edges (fully
      surrounds) the triangle's cell.
* Example: If a cell at (3, 3) has two triangles, the path
      has to go through **EXACTLY** two places of top (3,
      2), right (4, 3), bottom (3, 4), or left (2, 3) to
      touch the rule cell.

## Task & Output Format
1. **Identifying Objects:** Analyze the grid to identify
      the coordinates of the Start Node, End Node, and all
      objects (dots and gaps on path cells; squares, stars,
      triangles, polyshapes on rule cells - not all types
      must be present in the puzzle). Keep in mind that
      rule cells are guaranteed to be located at
      coordinates where both `x` and `y` are odd. The path
      MUST NOT pass through these cells.
2. **Solve the Puzzle:** Determine the valid path from the
      Start Node to the End Node that satisfies all rules.
      Double check that the path doesn't go through any
      rule cells (where both coordinates are odd). This is
      the most common beginner mistake.
3. **Explain Reasoning:** Provide a step-by-step
      explanation of your thought process. Detail key
      deductions, how constraints were applied, and any
      backtracking or choices made.
4. **Provide Solution Path:** After the reasoning, output
      the exact marker string `####` followed immediately
      by the solution path as a list of node coordinates
      `(x, y)`. Include all intermediate nodes from start
      to end (the distance between each node must be 1).

**Example Solution Path Format:**
####
[(0, 0), (1, 0), (2, 0), (2, 1), ...]
```

F.3 Object-Detection Prompt

The object-detection prompt replaces the solving objective with a perception-only objective: rather than producing a path, the model must reconstruct the symbolic grid representation of the puzzle from the rendered image. The model outputs a 2D array of symbols using the legend defined inside the prompt, which we compare against the ground-truth grid to measure perception accuracy in isolation from path-finding. The placeholders `{start_marker}` and `{end_marker}` follow the same convention as in Appendix F.2.

```
You are an expert spatial detection AI specializing in
      detecting objects from the game 'The Witness'.
Your task is to provide an array of all objects present in
      the puzzle grid following the specified format.

## Grid and Object Basics
* **Grid Dimensions:** The puzzle grid has {cols} columns
      and {rows} rows.
* **Coordinate System:** Nodes are identified by `(x, y)`
      coordinates. `(0,0)` is the top-left node.
      `({max_x},{max_y})` is the bottom-right node. `x`
      increases to the right, `y` increases downwards.
      Nodes can either be path cells or rule cells.
* **Path Cells:** Path cells are dark grey and are at all
      positions where at least one of `x` or `y` is even.
      Path cells can contain dots (black hexagons) or gaps
      (absence of a path where there would normally be one)
      but can also be empty (normal path).
* **Rule Cells:** Light green cells are rule cells and have
      coordinates where both `x` and `y` are odd. Rule
      cells can contain rule symbols (square, star,
      triangles, polyshapes) but can also be empty.
* **Start and End Nodes:** Exactly one of the path cells on
      the edge of the board contains a `Start Node` (big
      circle on the line in the same color as the grey
      path{start_marker}) and exactly one of the path cells
      on the edge of the board contains an `End Node` (node
      from which you can escape the grid to the rounded
      path outside{end_marker}). Both these nodes are
      guaranteed to be on valid path cells on the edge of
      the board (at least one of `x` and `y` has to be an
      even number).
* **Polyshapes and Negative Polyshapes:** One or multiple
      filled squares (polyshapes) or unfilled squares
      (negative polyshapes) in a specific arrangement that
      defines a shape. The arrangement of the squares
      defines the shape.
* **Shape Representation:** Represent the shape (Y) of
      polyshapes and negative polyshapes as a string of 1s
      and 0s, where 1 represents a filled square and 0
      represents an empty square and - represents a line
      break (from top top left to bottom right). For
      example, a 2x2 square would be represented as
      "11-11", a small L-shape could be "10-11", and a
      small T-shape could be "111-010", a large T-shape
      could be "111-010-010".
* **Color Codes:** R=Red, B=Blue, G=Green, Y=Yellow,
      W=White, O=Orange, P=Purple, K=Black

## Symbol Legend
* `S`: **Start Node**
* `E`: **End Node**
* `+`: Empty path cell
* `N`: Empty rule cell
* `G`: **Gap** (gap where a path would normally be)
* `.`: **Dot** (black hexagon on a path cell)
* `o-X`: **Square** of color X (fills our around half of
      the rule cell)
```

```
* `*-X`: **Star** of color X
* `A-X`: **1 Triangle** of color X
* `B-X`: **2 Triangles** of color X
* `C-X`: **3 Triangles** of color X
* `D-X`: **4 Triangles** of color X
* `P-X-Y`: **Polyshape** (positive) of color X and shape Y
      (one or multiple filled small squares in a specific
      arrangement)
* `Y-X-Y`: **Negative Polyshape** (ylop) of color X and
      shape Y (one or multiple unfilled small squares in a
      specific arrangement)

## Task & Output Format
1. **Identifying Objects:** Analyze the grid to identify
      the coordinates of all objects and non-objects. Keep
      in mind that rule cells are guaranteed to be located
      at coordinates where both `x` and `y` are odd.
2. **Identify Object Colors and Shapes:** For every
      detected object, determine its color and, for
      polyshapes and negative polyshapes, its shape. Double
      check the parity convention: rule cells are at
      coordinates where both `x` and `y` are odd, and path
      cells are at coordinates where at least one of `x`
      and `y` is even. Do NOT solve the puzzle; report only
      what is visible in the image.
3. **Explain Reasoning:** For all coordinates, write down
      objects, colors, shapes and explain your reasoning if
      things were unclear.
4. **Provide Solution Array:** After the reasoning, output
      the exact marker string `####` followed immediately
      by the solution array as a list of list of strings
      (use '' to indicate a string). Include the
      abbreviation for all objects and all non-objects at
      all coordinates in the specified format. The output
      *MUST* follow this format to be correctly parsed.

**Example Solution Format:**
* Assume we have a 5x5 grid with a Start Node at (0, 3) and
      an End Node at (4, 2) and a black hexagon (dot) at
      (2, 0) with the rest of the path cells empty and rule
      cells at (1,1), (1,3), (3,1), (3,3) with a red square
      at (1,1), 3 green triangles at (1,3) and positive
      polyshapes (multiple filled squares) of color blue
      and small L shape at (3,3). Then the output should
      end with:
####
[['+', '+', '+', 'S', '+'], ['+', 'o-R', '+', 'C-G', '+'],
      ['.', '+', '+', '+', '+'], ['+', 'N', '+',
      'P-B-11-10', '+'], ['+', '+', 'E', '+', '+']]
```

G AI-Usage Card

AI Usage Card

PROJECT DETAILS	PROJECT NAME Is Visual Prompting All You Need? Studying VLM Spatial Reasoning under Progressive Visual Scaffolds	DOMAIN Multimodal Machine Learning	KEY APPLICATION Spatial Visual Reasoning
CONTACT(S)	NAME(S) Lars Benedikt Kaesberg	EMAIL(S) l.kaesberg@uni-goettingen.de	AFFILIATION(S) University of Göttingen, Germany
MODEL(S)	MODEL NAME(S) Claude Opus 4.7		
LITERATURE REVIEW	FINDING LITERATURE Claude	FINDING EXAMPLES FROM KNOWN LITERATURE OR ADDING LITERATURE FOR EXISTING STATEMENTS Claude	COMPARING LITERATURE
WRITING	GENERATING NEW TEXT BASED ON INSTRUCTIONS	ASSISTING IN IMPROVING OWN CONTENT OR PARAPHRASING RELATED WORK Claude	PUTTING OTHER WORKS IN PERSPECTIVE
CODING	GENERATING NEW CODE BASED ON DESCRIPTIONS OR EXISTING CODE Claude	REFACTORING AND OPTIMIZING EXISTING CODE Claude	COMPARING ASPECTS OF EXISTING CODE
ETHICS	WHY DID WE USE AI FOR THIS PROJECT? Speed, finding new perspectives, and proofreading	WHAT STEPS ARE WE TAKING TO MITIGATE ERRORS OF AI? Code Reviews	WHAT STEPS ARE WE TAKING TO MINIMIZE THE CHANCE OF HARM OR INAPPROPRIATE USE OF AI?

THE CORRESPONDING AUTHORS VERIFY AND AGREE WITH THE MODIFICATIONS OR GENERATIONS OF THEIR USED AI-GENERATED CONTENT

AI Usage Card v2.6 https://ai-cards.org (Wahle et al., 2023)