SketchVLM: Vision-Language Models Can Annotate Images to Explain Thoughts and Guide Users

Brandon Collins[1,*], **Logan Bolton**[1,*], **Hung Huy Nguyen**[1,*]
Mohammad Reza Taesiri[2], **Trung Bui**[3], **Anh Totti Nguyen**[1]

[1]Auburn University, Auburn, Alabama, USA
[2]Independent Researcher [3]Adobe Research

blc0063@auburn.edu, logan.bolton@auburn.edu, huyhung.dknec@gmail.com
mtaesiri@gmail.com, bui@adobe.com, anh.ng8@gmail.com

Abstract

When answering questions about images, humans naturally point, label, and draw to explain their reasoning. In contrast, modern vision–language models (VLMs) such as Gemini-3-Pro and GPT-5 only respond with text, which can be difficult for users to verify. We present SketchVLM, a training-free, model-agnostic framework that enables VLMs to produce non-destructive, editable SVG overlays on the input image to visually explain their answers. Across seven benchmarks spanning visual reasoning (maze navigation, ball-drop trajectory prediction, and object counting) and drawing (part labeling, connecting-the-dots, and drawing shapes around objects), SketchVLM improves visual reasoning task accuracy by up to +28.5 percentage points and annotation quality by up to 1.48× relative to image-editing and fine-tuned sketching baselines, while also producing annotations that are more faithful to the model's stated answer. We find that single-turn generation already achieves strong accuracy and annotation quality, and multi-turn generation opens up further opportunities for human-AI collaboration. An interactive demo and code can be found at sketchVLM.github.io.

1 Introduction

From browsers (Atlas, MS Edge (Swain)) to office products (MS Copilot, Gemini (Team et al., 2023)), the market for LLM-powered chatbots for general end users is estimated to grow from $1.1B in 2023 to $83B in 2032 (Bloomberg Intelligence, 2024). As these assistants increasingly answer image-based questions in high-frequency consumer and productivity workflows (Swain; Perez, 2025; Team et al., 2023; Bloomberg Intelligence, 2024), users need responses they can quickly understand and verify (Nguyen et al., 2025).

However, modern VLMs such as Gemini-3-Pro and GPT-5 typically respond with a block of text, which can be difficult to verify (Zhou et al., 2025) (Fig. A2). To explain their thoughts on an image problem, humans often point, circle, underline, label, and annotate directly on the image. For example, when a user asks to check a car's oil level, a visual annotation is easier to verify than a paragraph of text (Fig. 1) (Masters, 2026; OpenAI, 2026).

Some VLMs emit point coordinates for referencing objects (e.g., MoonDream (Vikhyat, 2023), Molmo (Allen Institute for AI, 2024)), but they do not support free-form visual annotation. Other models are trained to generate visual reasoning traces but do not generalize well beyond their training domains (Wu et al., 2025; Gu et al., 2025). Image-editing models can visualize intermediate reasoning steps for multimodal questions (Zou et al., 2025), but risk altering the source image in unintended ways, undermining user trust (Taesiri et al., 2025).

To address these issues, we propose SketchVLM, a state-of-the-art (SotA) system that draws SVG annotations in a separate layer overlaid on top of the input image to explain its reasoning. SketchVLM grounds its annotations directly on the original image without modifying it and without requiring task-specific training. We test SketchVLM with multiple VLM backbones, including Gemini-3-Pro-Preview (✦) (Pichai et al., 2025) and GPT-5 (⬡) (OpenAI, 2025). We collectively refer to these models harnessed with SketchVLM as SketchVLMs and individually as ✦ sketchVLM and ⬡ sketchVLM. We compare against the leading alternative approaches for generating visual annotations such as the SotA image-editing model Nano Banana Pro (DeepMind, 2025b) and specialized VLMs fine-tuned to produce image annotations, ViLaSR (Wu et al., 2025) and ThinkMorph (Gu et al., 2025).

Across **seven tasks**: (a) **three drawing** tasks

[*]All three co-first authors made major contributions to dataset creation, experiments, and the manuscript. See the detailed Author contribution statement.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 55534–55600
October 24-29, 2026 ©2026 Association for Computational Linguistics

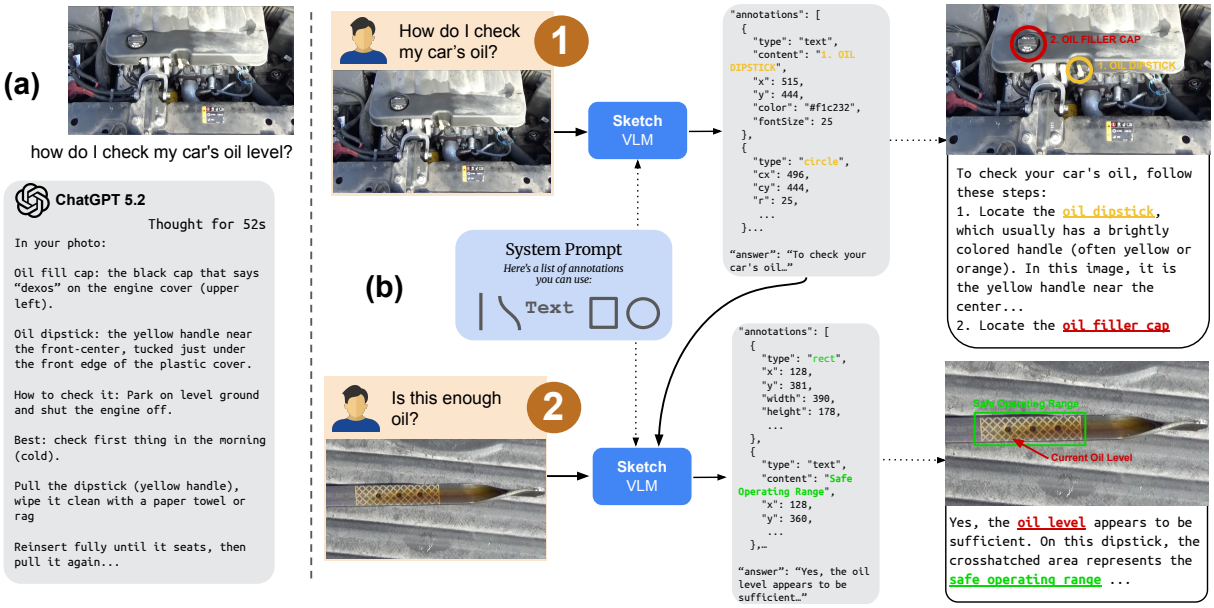


Figure 1: For complex questions, modern chatbots like ChatGPT often return long text responses **(a)** that are hard for users to understand, verify, and follow. In contrast, SketchVLM guides users **(b)** step-by-step by annotating the input image and grounding answers to relevant image regions—here, guiding a user how to check their car's oil level (source: https://www.youtube.com/watch?v=tNNyu9S65E4).

(connecting-the-dots, labeling parts of an object, and drawing shapes around objects) and (b) **four visual reasoning** tasks (two physics understanding tasks, one counting task and one maze navigation task), our main findings are:

- SketchVLMs based on frontier models (and ✦) generate annotations of superior **generalizability, accuracy, and annotation quality** compared to those by specialized fine-tuned sketching models (ViLaSR and ThinkMorph) and image-editing models (Nano Banana Pro) (Secs. 5.2 to 5.8).
- SketchVLM transfers without model-specific training to four additional open-weight VLMs, with Kimi K3 achieving task accuracy and spatial precision competitive with our primary Gemini-3-Pro-Preview backbone (Sec. 5.9).
- SketchVLMs have similar accuracy in single-turn and multi-turn settings, but are significantly faster with single-turn generation (Sec. 5.10).
- Adding an external grid of x–y coordinates to the input image improves sketchVLM's drawing capability and question answering accuracy but is not necessary for ✦ sketchVLM (Sec. 5.1).

2 Related Work

Native image-editing models (e.g., GPT-Image-1.5 and Nano Banana Pro) can directly modify images to annotate, but they can struggle for reasoning-heavy tasks and may cause unintended alterations to the source image (Zou et al., 2025). Open-source native multimodal autoregressive models (e.g., Chameleon (Team, 2024) and Bagel (Deng et al., 2025)) enable interleaved text–image generation, but lack editable overlays. In contrast, SketchVLMs generate a non-destructive SVG overlay on the input image (Tab. 1).

Tool-calling and code generation Agentic systems improve visual reasoning by writing code for external tools or to manipulate the input image. V* (Wu and Xie, 2023) uses LLM-guided visual search to localize target objects, then crops relevant regions. Visual Sketchpad (Hu et al., 2024) and OpenThinkIMG (Su et al., 2025) equip VLMs with tools such as segmentation and OCR. PyVision (Zhao et al., 2025) generates Python code to draw structured overlays. Other training-free methods leverage internal attention or gradient maps to crop and zoom (Zhang et al., 2025c). These systems add additional complexity to the VLM generation process by using external tools or code rather than utilizing the VLM's natural ability to generate annotations.

Visual prompting e.g., drawing coordinate points

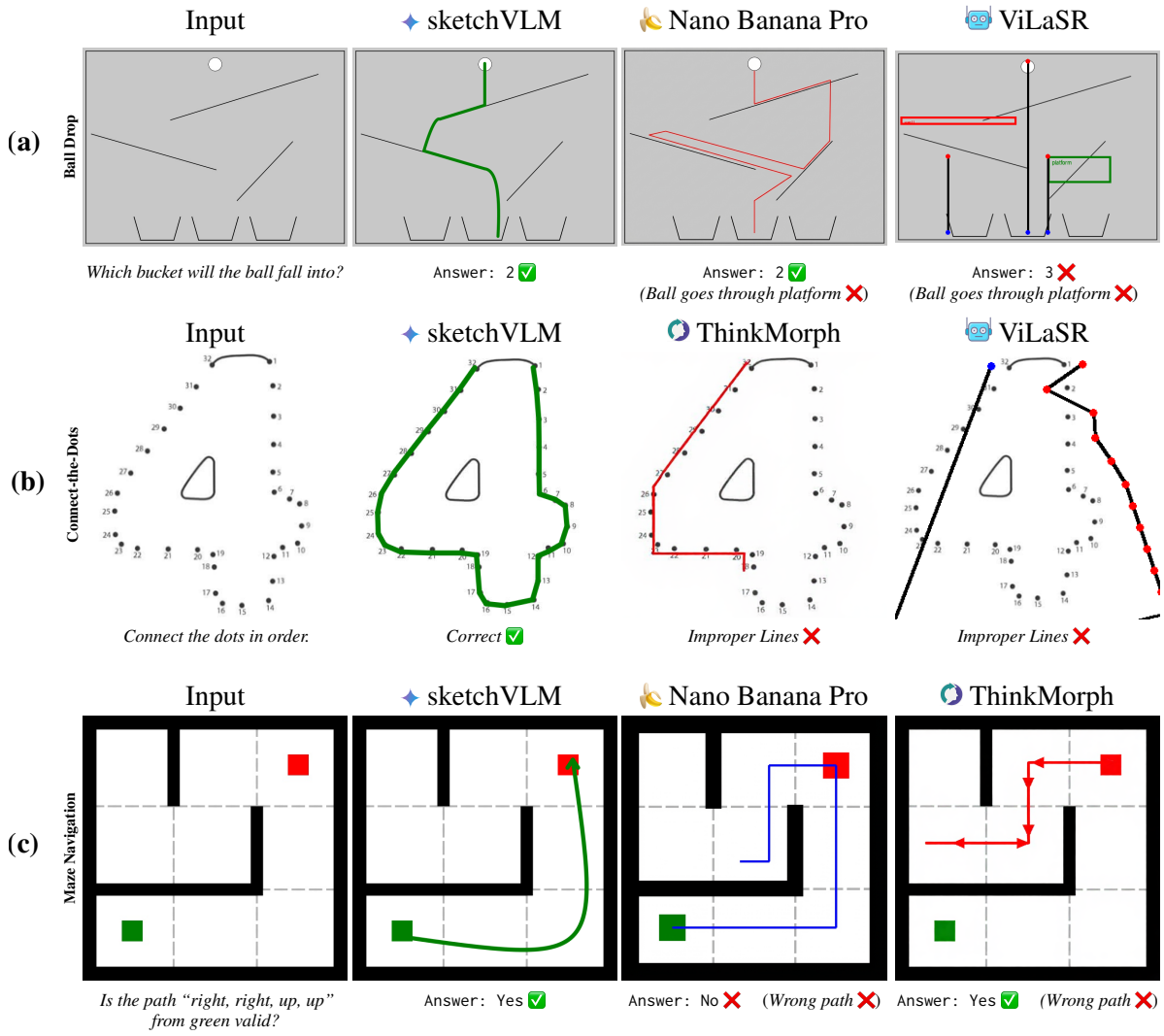
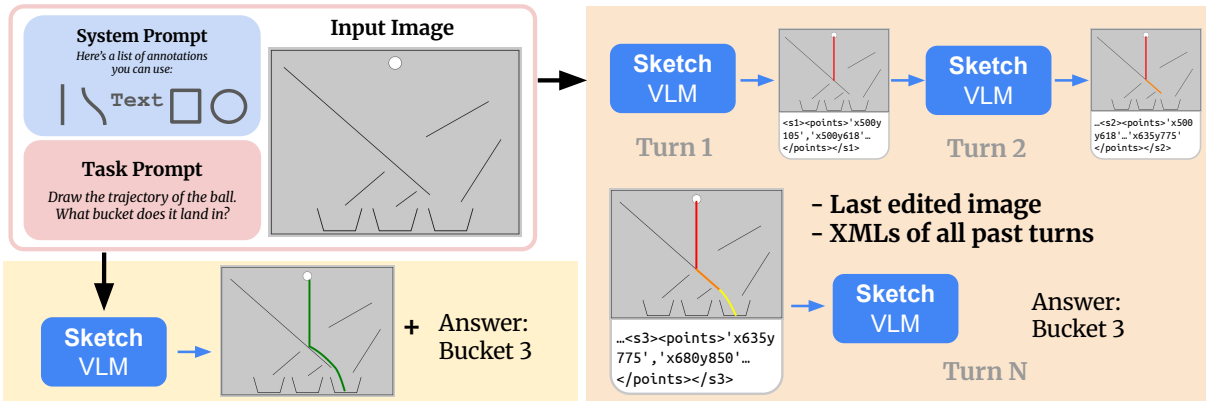


Figure 2: Our ✦ sketchVLM (with Gemini-3-Pro-Preview) draws more accurate trajectories in Ball Drop (a), connects the dots more accurately (b), and correctly verifies the proposed maze path (c), compared with the other baselines. Nano Banana Pro generates implausible trajectories while specialist fine-tuned VLMs (ThinkMorph and ViLaSR) often fail to generalize to new tasks.

or horizontal lines directly on an input image can improve the vision capabilities of VLMs (Yu et al., 2024; Lei et al., 2025; Izadi et al., 2025). Similarly, SketchAgent (Vinker et al., 2025) appends a coordinate grid to the edge of the input image to allow the model to reference precise x–y positions.

Visual sketching Whiteboard-of-Thought (Menon et al., 2024) renders Matplotlib sketches before answering questions. D2R (Ou et al., 2025) interleaves textual reasoning with visual drafts at each reasoning step. VDLM (Wang et al., 2025) converts images into SVG-based representations to improve visual understanding. SketchAgent (Vinker et al., 2025) and SketchFormer (Ribeiro et al., 2020) focus on sketch generation on a blank canvas. In contrast, SketchVLMs generate non-destructive, editable SVG annotations directly on input images, allowing users to inspect model reasoning without modifying the source image.

Fine-tuned sketching models MVoT (Li et al., 2025) fine-tunes a model to generate interleaved text and image reasoning traces. LatentSketchpad (Zhang et al., 2025b) and DeepSketcher (Zhang et al., 2025a) both move visual reasoning into latent spaces. ViLaSR (Wu et al., 2025) and ThinkMorph fine-tune VLMs to sketch on images, with ThinkMorph editing the image rather than producing overlays. Instead, we build a training-free SVG-based harness around SotA VLMs that enables them to annotate images and generalize to new domains.



(a) Single-turn (b) Multi-turn

Figure 3: Single-turn and multi-turn generation on the same VPCT sample. In (a) *single-turn*, SketchVLM receives the system prompt, the task prompt, and the input image, then outputs all annotations and the final answer in a single model call. In (b) *multi-turn*, Turn 1 uses the same inputs and outputs one annotation. For later turns, the model reuses the system prompt, the task prompt, and the previous annotations, which are provided in both the rendered image and text form. This process repeats until the model outputs its final text answer on the last turn.

Table 1: Comparison of sketching models and methods. *Annotation type* describes the visual artifact used during reasoning: *Vector overlay* denotes structured, non-destructive marks (e.g., strokes/boxes/text) aligned to an image or canvas, while *Image edit* denotes pixel-space image modification or synthesis that may change image content. *Input image* is marked only when a provided image is the primary visual context (vs. blank canvas or purely generative visual thoughts). *Free-form drawing* indicates support for arbitrary stroke-like annotations beyond a fixed mark set.

Model name	Training free	Multi turn	Input image	Free-form drawing	Annotation type
SketchVLM (Ours)	✓	✓	✓	✓	Vector overlay
VisualSketchPad (Hu et al., 2024)	✓	✓	✓	✗	Vector overlay
PyVision (Zhao et al., 2025)	✓	✓	✓	✗	Vector overlay
SketchAgent (Vinker et al., 2025)	✓	✓	✗	✓	Vector overlay
D2R (Ou et al., 2025)	✓	✓	✓	✗	Image edit
Whiteboard-of-Thought (Menon et al., 2024)	✓	✗	✗	✗	Vector overlay
ViLaSR (Wu et al., 2025)	✗	✓	✓	✗	Vector overlay
OpenThinkIMG (Su et al., 2025)	✗	✓	✓	✗	Image edit
ThinkMorph (Gu et al., 2025)	✗	✗	✓	✗	Image edit
MVoT (Li et al., 2025)	✗	✓	✓	✗	Image edit
LatentSketchpad (Zhang et al., 2025b)	✗	✓	✓	✗	Image edit
DeepSketcher (Zhang et al., 2025a)	✗	✓	✓	✗	Image edit

3 SketchVLM

SketchVLM combines three components: visual prompting to aid spatial reference, a system prompt that elicits structured stroke outputs, and XML-to-SVG conversion that renders those strokes as an overlay on the source image (Figs. 1 and 3).

Visual prompting Following SketchAgent (Vinker et al., 2025), we append a coordinate grid to the left and bottom of each image to improve spatial precision (Fig. D22).

Input prompt We introduce a system prompt (Sec. F.2) that instructs the model to output structured annotations (e.g., XML-style <s1>, <s2>, . . . , <sN> stroke tags). We provide instructions for drawing primitives including rectangles, arrows, text labels, straight lines, and Bézier curves. Given the task prompt (e.g., *"Which bucket will the ball fall into?"*), the model annotates its reasoning before producing a final answer (Figs. 1 and 2).

SVG conversion We parse the model's XML outputs into SVG overlays rendered on the input image. Two-point strokes are rendered as straight lines. Otherwise, given a stroke described by m ordered samples $S_i = \{(x_j, y_j)\}_{j=1}^{m}$ and corresponding normalized timestamps $T_i = \{t_j\}_{j=1}^{m}$ with $t_j \in [0, 1]$, we fit a smooth Bézier curve. Following SketchAgent (Vinker et al., 2025), a least squares solution is found for the control points in a cubic Bézier curve (see Fig. F5 for an example). Following SketchAgent, we run all experiments using XML format. Given that VLMs produce JSON at comparable quality and JSON is more human-readable, we use JSON for our interactive demo.

4 Evaluation

4.1 7 Tasks

1. Connect-the-Dots Models must locate each dot and connect them in order (Secs. B.1 and D.3 and Fig. 2).

2. Counting Objects Models must count the target objects and place numbered markers on each one (Secs. B.2 and D.4 and Fig. 4).

3. Drawing Shapes around Objects Models must localize objects by drawing rectangles or ovals (Secs. B.3 and D.5 and Fig. 5).

4. Part Labeling Models must place the correct text labels at the corresponding part locations (Secs. B.4 and D.6 and Fig. D1).
5. Maze Navigation Models must trace a proposed path and determine if it reaches the goal without crossing walls (Secs. B.5 and D.2 and Fig. 2).
6. VPCT is the Visual Physics Comprehension Test (cbrower, 2025). Models must predict which container a dropped ball will land in (Sec. B.6 and Figs. 2 and D3).
7. Ball Drop contains synthetically generated images with ground-truth ball trajectories produced using PHYRE (Bakhtin et al., 2019). Models must predict the landing container and trace the ball trajectory (Sec. B.6 and Figs. D2, D4 and D5).

4.2 Setup

Single-turn and multi-turn We evaluate SketchVLMs in (1) *single-turn*, where the models output all annotations and their final answer in one response, and (2) *multi-turn*, producing one stroke per turn, to simulate iterative, real-world conversations. During each turn, the VLM receives the image with all previous annotations rendered, and the annotations' text representations. The model then outputs its final text answer on the last turn (Fig. 3).
Visual prompting We evaluate the necessity of a coordinate grid for the model to reference specific points in an image. When omitting the grid, the model outputs coordinates on a normalized 1000×1000 scale (Fig. D24).

4.3 Baselines

Image Editing Model + VLM: Nano Banana Pro is an image-editing model that can annotate images, but produces no text answer. To obtain a text response, we feed its edited image to Gemini-3-Pro-Preview (✦), denoted as Nano Banana + Gemini.
Fine-tuned Sketching Models: ViLaSR is fine-tuned from Qwen-2.5-VL-7B (Bai et al., 2025) to autoregressively generate SVG annotations on the input image over multiple turns, and ThinkMorph is fine-tuned from BAGEL-7B-MoT (Deng et al., 2025) to directly edit the input image and produce a text answer.
Default VLMs: We additionally evaluate text-only Gemini-3-Pro-Preview (✦) and GPT-5 ().

4.4 Metrics

We measure three aspects to test the accuracy and consistency between text responses and annotations. **1) Accuracy** serves as the primary measure of task performance and tests the effect of sketching on answer correctness. **2) Annotation–text alignment** measures how faithful the visual traces are to the text answer. We ask a VLM judge (Chen et al., 2024) to infer the answer from the annotations and report how often the inferred answer matches the final text answer. **3) Annotation quality** allows us to distinguish models that produce informative annotations from those that output low-quality or incoherent drawings. We adopt a VLM-as-a-Judge approach using a rubric scored from 1 to 5 (Sec. E) that evaluates annotation plausibility and visual clarity for each task.

5 Results

5.1 Grid prompting improves GPT-5's annotation precision, but hurts Gemini-3-Pro-Preview ✦

Table 2: Ablation across inputs in single-turn mode. "Sketch" adds strokes/system prompt; "Grid" additionally overlays the coordinate grid. Median per-sample RMSE is reported for Connect-the-Dots while accuracy is reported for the other tasks. Lower median RMSE is better. sketchVLM works best with the grid while ✦ sketchVLM works best without the grid.
✦ = Gemini-3-Pro-Preview, = GPT-5.

Model	Input	VPCT			
	Image	63.5	66.0	92.3	N/A
	+ Grid	67.0	67.2	**98.0**	N/A
	+ Sketch	59.0	63.1	88.7	110.5
	+ Sketch + Grid + Multi-turn	68.0	63.1	82.1	41.7
	+ Sketch + Grid (i.e., **sketchVLM**)	**70.0**	**68.5**	92.8	**40.1**
	Image	89.3	**83.8**	99.3	N/A
	+ Grid	90.0	81.3	**99.5**	N/A
✦	+ Sketch (i.e., ✦ **sketchVLM**)	**96.0**	79.7	98.0	**4.6**
	+ Sketch + Multi-turn	80.0	79.8	98.2	4.8
	+ Sketch + Grid	91.0	82.3	**99.5**	33.2

To understand how applying an external coordinate grid to the image affects SketchVLMs' annotation accuracy, we run ablations with different setups.
Experiment We evaluate four input configurations in single-turn mode: the base image alone, image with grid, image with sketching prompt, and image with both. We report accuracy on VPCT, Ball Drop, and Maze Navigation, and median per-sample RMSE on Connect-the-Dots (Tab. 2).
Results sketchVLM performs best with both the grid and sketching prompt, with the grid providing a consistent boost to spatial precision (Fig. D22), consistent with prior findings (Izadi et al., 2025). ✦ sketchVLM performs better without the grid,

Table 3: SketchVLMs (using ✦ Gemini-3-Pro-Preview and GPT-5) produce visual reasoning traces while maintaining competitive accuracy. Nano Banana + Gemini underperforms default ✦ Gemini-3-Pro-Preview, and fine-tuned sketching models (ViLaSR, ThinkMorph) perform poorly on all tasks.

Model	VPCT video physics	ball drop	maze trace	counting	labeling	draw shapes □	draw shapes ○	connect dots Median RMSE	connect dots Order Acc%
✦ sketchVLM	**96.0 ± 1.4**	79.7 ± 2.8	98.0 ± 1.7	**94.5**	60.3	58.8	55.4	**4.6**	**99.0**
✦ Gemini-3-Pro-Preview	89.3 ± 2.2	**83.8 ± 3.4**	**99.3 ± 0.8**	93.0	**64.1**	**63.1**	**59.8**	†	†
sketchVLM	**70.0 ± 2.9**	**68.5 ± 2.2**	**92.8 ± 2.5**	**75.4**	**20.4**	18.7	11.2	40.1	74.0
GPT-5	63.5 ± 2.5	66.0 ± 4.7	92.3 ± 2.3	72.0	19.1	**22.4**	**15.4**	†	†
Nano Banana + Gemini	63.0	62.6	93.3 ± 3.0	91.7	†	†	†	†	39.0
ViLaSR	37.0	35.9	50.8 ± 1.5	48.6	–	–	†	175.5	9.0
ThinkMorph	27.0	30.3	62.5 ± 2.1	68.1	†	†	†	†	0.0

† *Model is unable to output this format.*

and adding it notably degrades localization on Connect-the-Dots (median per-sample RMSE increases from 4.6 to 33.2). We therefore report all results in Tab. 3 with the input grid for sketchVLM but without the grid for ✦ sketchVLM. We treat grid usage as a model-dependent hyperparameter rather than an essential component of the method. A discretized grid limits the minimum possible RMSE, and we hypothesize that stronger models localize better than this limit while smaller models improve with the grid. We investigate this discrepancy further in Fig. D24 and Fig. C2.

5.2 SketchVLMs can localize points and connect them in order

Connecting the dots tests both spatial grounding and producing multiple coherent strokes in a row.

Experiment For each sample, we compute root mean squared error (RMSE) in pixels between the ground-truth points and their closest generated points. We report the median per-sample RMSE across the dataset. To evaluate whether models connect points in the correct order, we compare each predicted segment *i* against all ground-truth segment pairs using MSE. If segment *i* has lower MSE to a different ground-truth pair than its expected pair *(i, i+1)*, it is counted as an ordering error. Because Nano Banana Pro and ThinkMorph only produce an image response with no x-y coordinates, we manually evaluate them.

Results ✦ sketchVLM can connect up to 35 points with a low median per-sample RMSE of 4.6 (Tab. 3). sketchVLM has a higher median per-sample RMSE of 40.1, but performs better than ViLaSR's median per-sample 175.5 RMSE. Nano Banana Pro, ThinkMorph, and ViLaSR frequently produce ordering errors and inadvertently alter the input image (Sec. C.3). In contrast, sketchVLM and ✦ sketchVLM correctly order points 74% and 99% of the time, respectively, demonstrating that SketchVLMs can reliably scale to many strokes while maintaining spatial accuracy and logical coherence (Fig. D23).

5.3 SketchVLM improves counting accuracy over text-only baselines

Existing VLMs can output point coordinates to count, but these points are unlabeled and can be tedious to verify. In contrast, SketchVLMs enable direct visual verification with numeric markers.

Experiment We measure the accuracy of SketchVLMs and also how well they ground their markers on counting datasets (Sec. 4.1). We consider a marker correct if it lies within the corresponding ground-truth bounding box, obtained via SAM-3 (Carion et al., 2025), allowing at most one marker per object.

Results ✦ sketchVLM exhibits strong consistency between counting accuracy (94.5) and numeric-marker location accuracy (95.9), whereas sketchVLM achieves high counting accuracy (75.4) but substantially lower numeric-marker location accuracy (51.0) (Tabs. 3 and C3). ViLaSR attains low counting accuracy (48.6) and numeric-marker location accuracy (59.9), indicating limited performance in both aspects (Tabs. 3 and C3). SketchVLM improves counting accuracy, yielding gains of (+1.5) points for ✦ sketchVLM and (+3.4) points for sketchVLM (Tab. 3). The explicit numeric markers further enable direct visual verification of model outputs (Fig. 4 and Sec. D.4).

5.4 SketchVLMs localize objects more accurately with pre-defined shape primitives than with free-form annotations

One design choice is whether to have SketchVLMs generate all of their drawings through free-form strokes, or to allow them to output shape information (such as the x–y position of the center of the

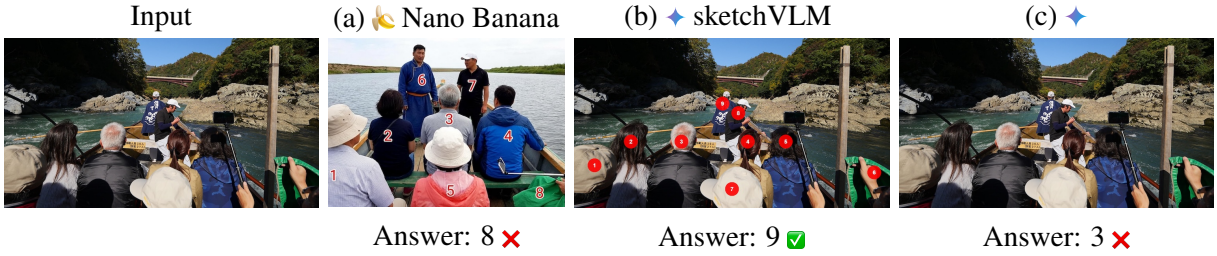


Figure 4: (a) Nano Banana Pro generates a different image and predicts an incorrect count. (c) Gemini-3-Pro-Preview directly outputs only a number without annotations and severely undercounts. In contrast, our sketchVLM (b) outputs the correct answer and produces visual annotations to explain its answer.

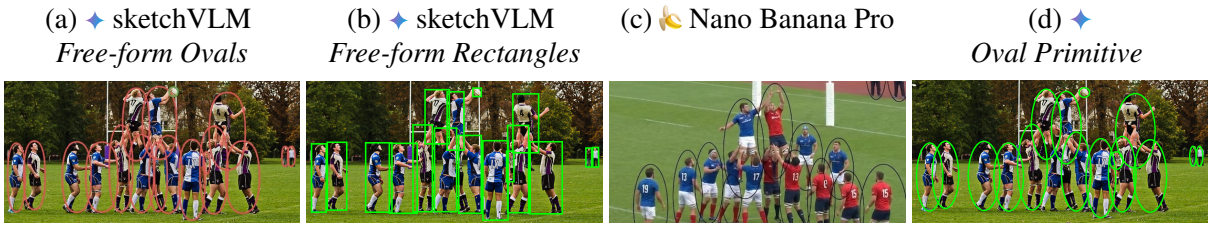


Figure 5: When prompted to outline the classes "person" and "sports-ball", (c) Nano Banana Pro replaces the original image with a newly generated one, whereas SketchVLM in (a) and (b) preserves the original image and draws shapes that align with object boundaries and locations as compared to default (d) Gemini-3-Pro-Preview.

circle and the radius) and then render through SVG conversion. Free-form strokes are more flexible but require point-by-point drawing. Pre-defined primitives for rectangles, ovals, etc. can be specified by a few parameters and rendered automatically.

Experiment We compare the accuracy of SketchVLMs' stroke-based annotations against the baseline models that directly output parameters for shape locations and shape size (Sec. 4.1). To evaluate oval annotations, we convert the rendered shape into its tight enclosing bounding box before computing metrics. Performance is measured using Average Precision (AP) at an IoU threshold of 0.5.

Results For sketchVLM , stroke-based rectangles slightly improve performance on medium (+0.6) and large objects (+1.4), but significantly degrade small-object detection (-10.1), reducing overall AP50 from 63.1 to 58.8 (Fig. 5, Sec. D.5, and Tab. C5).

Further analysis shows that SketchVLMs match the original model in precision but exhibit lower recall (Tab. C6), while prompt variations have minimal effect (Tab. C7). To combine the localization accuracy of pre-defined primitives with the expressiveness of free-form strokes, we allow the model to output both simultaneously (Fig. 1).

5.5 SketchVLM improves localization accuracy for GPT-5 but not for Gemini-3-Pro-Preview when labeling parts of an object

SketchVLMs can place semantically and spatially correct labels, enabling step-by-step instructions and visual explanations (Figs. 1 and A1).

Experiment Models predict both part labels and locations from a predefined label set. In SketchVLM, the framework automatically renders the predicted text at the predicted location (with adaptive size and color). For fair comparison, we render predicted labels from the original VLMs onto the image using a Python post-processing pipeline with manually chosen text size and color. We follow (Cheng et al., 2021) and evaluate under boundary dilation radius r to allow tolerance in spatial matching.

Results SketchVLM improves part labeling for GPT-5 (+1.3) but slightly underperforms Gemini-3-Pro-Preview at strict boundary matching (-3.8) (Tab. C10). Under boundary dilation, the Gemini gap narrows from (-3.8) at $r = 0$ to (-0.6) at $r = 7$. Error analysis shows the original VLMs produce more missing-label errors, while SketchVLMs produce more position errors (Tab. C9) that are typically visually negligible (Figs. C3 and D1 and Sec. D.6). These findings suggest that most SketchVLM errors are minor spatial shifts rather

than semantic labeling failures (Fig. C3).

5.6 SketchVLMs outperform models that were fine-tuned directly on path-tracing tasks

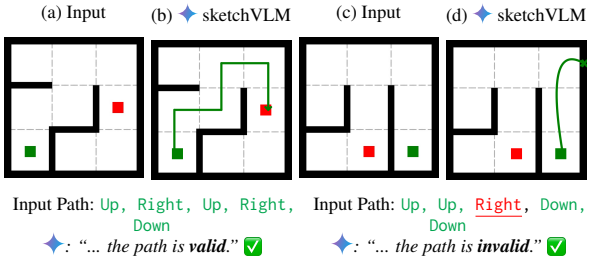


Figure 6: Models are presented with a blank maze such as (a) or (c) and are asked to verify whether a proposed path from the green square to the red square is feasible through annotations (b) and (d). SketchVLMs correctly verify both valid and invalid paths by drawing the trajectory and marking where an invalid move occurs.

Maze Navigation evaluates spatial reasoning by requiring models to follow a sequence of directions to reach a goal while avoiding obstacles. Given that ViLaSR and ThinkMorph are trained on a similar maze task, we expect them to perform well.

Experiment Given a set of directions, the model must sketch the path while also determining if the path reaches the goal without crossing any walls.

Results Surprisingly, we find that ViLaSR has an accuracy of 50.8% and ThinkMorph has an accuracy of 62.5%, both near the random choice baseline of 50%. Nano Banana Pro performs much stronger at 93.3%; however, it sometimes alters the entire image and has other unusual outputs (Fig. D6). Both sketchVLM and sketchVLM perform well with accuracies of 98.0% and 92.8% respectively. The difference in performance between the default model baseline and the SketchVLM mode is minor (Tab. 3), but the annotations benefit user verification.

5.7 Fine-tuned sketching models fail to generalize to unseen physics understanding tasks

In addition to spatial reasoning, we evaluate whether SketchVLM can predict trajectories involving physical dynamics, such as a ball falling and rolling.

Experiment Given an image with a ball and platforms, the model must sketch the ball's trajectory and output the container number it lands in.

Table 4: VLM-judged annotation–text alignment (higher is better) and annotation quality (1–5, higher is better). Models fine-tuned to generate annotations show lower alignment, while sketchVLM has the highest alignment and annotation quality.

Model	Annotation–Text Alignment				Annotation Quality			
	VPCT			Mean	VPCT			Mean
sketchVLM	99.0	**99.0**	**88.4**	**95.5**	1.83	1.74	3.20	2.26
sketchVLM	**100.0**	98.5	84.2	94.2	**3.12**	**4.28**	**3.69**	**3.70**
+Nano Banana	93.0	41.4	80.3	71.6	1.56	2.56	3.68	2.60
ViLaSR	32.0	45.5	8.2	28.6	1.36	1.28	2.78	1.81
ThinkMorph	54.0	38.4	48.1	46.8	1.62	2.11	1.17	1.63

Results We find that ViLaSR and ThinkMorph perform poorly on VPCT with accuracies of 37% and 27%, which are near the random choice baseline of 33.3%. On our Ball Drop dataset, they have accuracies of 35.9% and 30.3%, only slightly above the random choice baseline of 25.0%, showing how these models fail to generalize to tasks that they are not trained on. Nano Banana Pro often removes ledges in the image (Fig. D3) and performs significantly worse than the baseline Gemini-3-Pro-Preview accuracy (Tab. 3). In contrast, SketchVLMs output coherent annotations and have high accuracy with sketchVLM reaching 96.0% accuracy on VPCT and 79.7% on Ball Drop (Fig. D2).

5.8 SketchVLMs produce more faithful and higher-quality annotations than image-editing and fine-tuned models

Annotations should be visually clear, logically coherent, and aligned with the text answer. A trajectory that clips through walls, a label that floats in empty space, or uninformative annotations all undermine the user's ability to interpret the model's reasoning (Fig. E1).

Experiment To measure annotation–text alignment, we show each model's annotations for VPCT, Ball Drop and Maze Navigation (without its text answer) to a VLM judge (Gemini-3-Flash-Preview (DeepMind, 2025a)) and ask it to infer what the final answer should be from the annotations alone. Then to measure annotation quality, we give a VLM judge a 1–5 rubric (Sec. E) in which the judge is asked to evaluate annotation plausibility and visual clarity. We validate judge ratings in Sec. E.

Results Despite being training-free, SketchVLMs achieve substantially higher annotation–text alignment than all baselines (Tab. 4). sketchVLM and sketchVLM reach mean alignment scores of

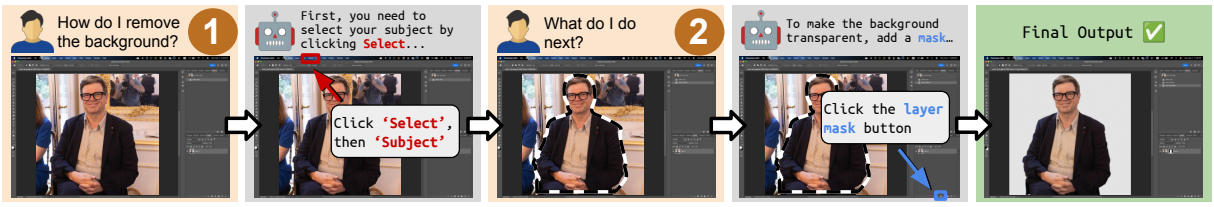


Figure 7: Multi-turn example of SketchVLM guiding a user through how to remove an image's background. At each turn, the model receives a screenshot then annotates the screenshot with labeled arrows and highlights UI elements to indicate the next step.

95.5% and 94.2%, respectively, compared to 71.6% for Nano Banana Pro, 46.8% for ThinkMorph, and 28.6% for ViLaSR. Based on the VLM judge's ratings, ✦ sketchVLM achieves the highest mean annotation quality score of 3.70, which is 1.10 points higher than the next-best method, Nano Banana Pro, and 1.44 points higher than sketchVLM (Tab. 4). Similarly, based on human ratings, ✦ sketchVLM achieves the highest mean quality score of 4.14, followed by sketchVLM at 3.70, Nano Banana Pro at 3.08, ViLaSR at 1.74, and ThinkMorph at 1.24 (Tab. E4). SketchVLMs' high annotation–text alignment and quality indicate that users can reliably look at the annotations to verify whether the model's reasoning makes sense.

5.9 SketchVLM generalizes to open-weight VLMs

Experiment To evaluate whether SketchVLM generalizes beyond our primary GPT-5 and Gemini-3-Pro-Preview backbones, we additionally test Gemma-4-31B (Gemma Team, 2026), Qwen3.5-397B-A17B (Qwen Team, 2026), Kimi K2.5 (Kimi Team, 2026a), and Kimi K3 (Kimi Team, 2026b). We evaluate Direct VQA and SketchVLM on the six tasks with directly comparable answer outputs. The complete results and input ablations are provided in Tabs. C12 and C14.

Results SketchVLM successfully elicits structured visual annotations from all four additional VLMs, demonstrating that the framework transfers without model-specific training. Kimi K3 is the strongest additional open-weight backbone with results competitive with Gemini-3-Pro-Preview.

Results are mixed for the other open-weight models. Gemma-4-31B significantly improves its accuracy in some tasks (+8 points in VPCT and +18.8 points in Counting Objects) while Kimi K2.5 and Qwen3.5-397B-A17B do not improve overall accuracy with SketchVLM. However, all models are successfully able to generate interpretable visual annotations without task-specific training (Sec. D).

5.10 Single-turn is as accurate as multi-turn but requires significantly fewer turns

We want SketchVLMs to visually guide users through tasks with multiple turns, such as removing the background of a photo (Fig. 7) or setting up an EC2 instance on AWS (Fig. A1). We compare accuracy between single-turn and multi-turn, and identify the best configuration for multi-turn.

Experiment We compare single-turn and multi-turn generation on VPCT, , , and . In multi-turn, the model receives all previously drawn strokes rendered onto the input image at each turn. We also test the necessity of providing the text representation of rendered strokes across turns.

Results Single-turn achieves comparable or higher accuracy than multi-turn across all tasks (Tab. 2), while requiring about 5.92x fewer turns (Tab. C13). This shows that SketchVLMs can produce high-quality annotations and accurate answers in a single pass. When we remove the text representation of prior strokes to force the model to rely solely on the rendered image, we observe notable degradation in annotation quality (Fig. D25). We therefore report all multi-turn results with both the rendered image and the text history provided to the model.

6 Conclusion

We present SketchVLM, a training-free framework that prompts frontier VLMs to produce editable, non-destructive SVG annotations grounded on the input image. These visual explanations let users verify model reasoning at a glance, outperform sketching models by +28.5 percentage points in accuracy (Tab. C1) and +48.3% in annotation quality (Tab. C2), while also generalizing well to real-world tasks (Sec. A).

Limitations

SketchVLM works best with GPT-5 and Gemini-3-Pro-Preview and can transfer to strong open-weight VLMs like Kimi K2.5 (Kimi Team, 2026a), Kimi K3, Gemma-4-31B, and Qwen3.5-397B-A17B (Tab. C12 and Sec. D), but does not perform well on small VLMs that struggle with instruction following like Qwen2.5-VL-7B (Bai et al., 2025). Additionally, enabling models to undo and edit strokes could be explored as a way to improve multi-turn performance.

Users may over-trust incorrect sketches, so SketchVLM should be used as a verification aid rather than a guarantee of correctness, and annotations should be checked carefully when used for safety-critical, medical, legal, or financial tasks.

Author contribution statement

Brandon Collins (BC), Logan Bolton (LB), and Hung Huy Nguyen (HN) made major contributions to benchmark creation and experimental evaluation.

- HN curated the Counting, Drawing Shapes, and Part Labeling benchmarks from existing datasets, led their evaluation, and led the human-versus-VLM agreement study.
- BC created the Connect-the-Dots benchmark and led method development, development of the inference and evaluation code, ablations, and evaluation on Connect-the-Dots and VPCT.
- LB created the Ball Drop and Maze Navigation benchmarks, led their evaluation, and ran the open-source model experiments as well as the annotation-quality and annotation–text alignment evaluations.

BC and LB led the writing of the manuscript, while all authors contributed to editing and reviewing it. BC, LB, and Anh Totti Nguyen (AN) developed the interactive demo. BC led development of the project website, with additional contributions from LB. AN supervised the project.

Acknowledgments

We thank Pooyan Rahmanzadehgervi for feedback. AN was supported by NSF Grant No. 2145767 and donations from the NaphCare Foundation and Adobe Research. LB and HN were supported by the AU URF and PGRF programs, respectively.

References

Manoj Acharya, Kushal Kafle, and Christopher Kanan. 2019. Tallyqa: Answering complex counting questions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8076–8084.

Allen Institute for AI. 2024. Molmo: An open vision-language model from allen ai. https://github.com/allenai/molmo. Open-source multimodal model family for vision-language tasks; accessed 2026-01-18.

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

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. 2019. PHYRE: A new benchmark for physical reasoning. In *Advances in Neural Information Processing Systems*, volume 32.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, and 1 others. 2024. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*.

Bloomberg Intelligence. 2024. Generative ai outlook. Technical report, Bloomberg, New York. Accessed: 2026-01-18.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, and 1 others. 2025. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*.

cbrower. 2025. Vpct ball drop benchmark. https://cbrower.dev/vpct. Accessed: 2025-11-09.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024. MLLM-as-a-judge: Assessing multimodal LLM-as-a-judge with vision-language benchmark. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6562–6595. PMLR.

Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. 2014. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pages 1971–1978.

Bowen Cheng, Ross Girshick, Piotr Dollar, Alexander C. Berg, and Alexander Kirillov. 2021. Boundary iou: Improving object-centric image segmentation evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15334–15342.

Google DeepMind. 2025a. Gemini 3 flash: frontier intelligence built for speed. The Keyword (Google Blog).

Google DeepMind. 2025b. Introducing nano banana pro. . Accessed: 2026-01-25.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, and 1 others. 2025. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 91–104.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. 2025. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*.

David H. Douglas and Thomas K. Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122.

Gemma Team. 2026. Gemma 4 technical report. *Preprint*, arXiv:2607.02770.

Jiawei Gu, Yunzhuo Hao, Huichen Will Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. 2025. Thinkmorph: Emergent properties in multimodal interleaved chain-of-thought reasoning. *arXiv preprint arXiv:2510.27492*.

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. *Advances in Neural Information Processing Systems*, 37:139348–139379.

Amirmohammad Izadi, Mohammad Ali Banayeeanzade, Fatemeh Askari, Ali Rahimiakbar, Mohammad Mahdi Vahedi, Hosein Hasani, and Mahdieh Soleymani Baghshah. 2025. Visual structures helps visual reasoning: Addressing the binding problem in vlms. *arXiv preprint arXiv:2506.22146*.

Kimi Team. 2026a. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

Kimi Team. 2026b. Kimi k3: Open frontier intelligence. *Preprint*, arXiv:2607.24653.

Ehsan Latif, Zirak Khan, and Xiaoming Zhai. 2025. Sketchmind: A multi-agent cognitive framework for assessing student-drawn scientific sketches. *arXiv preprint arXiv:2507.22904*.

Xuanyu Lei, Zonghan Yang, Xinrui Chen, Peng Li, and Yang Liu. 2025. Scaffolding coordinates to promote vision-language coordination in large multimodal models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2886–2903, Abu Dhabi, UAE. Association for Computational Linguistics.

Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. 2025. Imagine while reasoning in space: Multimodal visualization-of-thought. *Preprint*, arXiv:2501.07542.

Haodong Li, Jingwei Wu, Quan Sun, Guopeng Li, Juanxi Tian, Huanyu Zhang, Yanlin Lai, Ruichuan An, Hongbo Peng, Yuhong Dai, and 1 others. 2026. Gebench: Benchmarking image generation models as gui environments. *arXiv preprint arXiv:2602.09007*.

Kiri Masters. 2026. Why OpenAI's ad announcement should worry retail media networks.

Sachit Menon, Richard Zemel, and Carl Vondrick. 2024. Whiteboard-of-thought: Thinking step-by-step across modalities. *arXiv*.

Tin Nguyen, Logan Bolton, Mohammad Reza Taesiri, Trung Bui, and Anh Totti Nguyen. 2025. Hot: Highlighted chain of thought for referencing supporting facts from inputs. *arXiv preprint arXiv:2503.02003*.

OpenAI. 2025. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

OpenAI. 2026. Fix with chatgpt.

Openclipart Contributors. 2025. Openclipart silhouette collection. `https://openclipart.org/search/?query=silhouette`. Accessed: 2025-11-10.

Siqu Ou, Hongcheng Liu, Pingjie Wang, Yusheng Liao, Chuan Xuan, Yanfeng Wang, and Yu Wang. 2025. Bridging the dynamic perception gap: Training-free draft chain-of-thought for dynamic multimodal spatial reasoning. *Preprint*, arXiv:2505.16579.

Sarah Perez. 2025. Chatgpt's user growth has slowed, report finds | techcrunch. [Online; accessed 2026-01-28].

Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. 2025. A new era of intelligence with gemini 3. The Keyword (Google Blog).

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents. Accessed: 2026-08-27.

Vignesh Ramanathan, Anmol Kalia, Vladan Petrovic, Yi Wen, Baixue Zheng, Baishan Guo, Rui Wang, Aaron Marquez, Rama Kovvuri, Abhishek Kadian, and 1 others. 2023. Paco: Parts and attributes of

common objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7141–7151.

Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. 2020. Sketchformer: Transformer-based representation for sketched structure. *Preprint*, arXiv:2002.10381.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and 1 others. 2025. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*.

By Gyana Swain. Keep ai browsers out of your enterprise, warns gartner – computerworld. [Online; accessed 2026-01-28].

Mohammad Reza Taesiri, Brandon Collins, Logan Bolton, Viet Dac Lai, Franck Dernoncourt, Trung Bui, and Anh Totti Nguyen. 2025. Understanding generative ai capabilities in everyday image editing tasks. *arXiv preprint arXiv:2505.16181*. Version 2, submitted 26 May 2025.

Chameleon Team. 2024. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Vikhyat. 2023. Moondream: Tiny vision language model. `https://github.com/vikhyat/moondream`. Open-source vision-language model with small-footprint multimodal capabilities.

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

Zhenhailong Wang, Joy Hsu, Xingyao Wang, Kuan-Hao Huang, Manling Li, Jiajun Wu, and Heng Ji. 2025. Visually descriptive language model for vector graphics reasoning. *Transactions on Machine Learning Research*.

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

Penghao Wu and Saining Xie. 2023. V*: Guided visual search as a core mechanism in multimodal llms. *Preprint*, arXiv:2312.14135.

Tong Yu and 1 others. 2024. Visual prompting in multimodal large language models: A survey. *arXiv preprint arXiv:2409.15310*.

Chi Zhang, Haibo Qiu, Qiming Zhang, Zhixiong Zeng, Lin Ma, and Jing Zhang. 2025a. Deepsketcher: Internalizing visual manipulation for multimodal reasoning. *Preprint*, arXiv:2509.25866.

Huanyu Zhang, Wenshan Wu, Chengzu Li, Ning Shang, Yan Xia, Yangyu Huang, Yifan Zhang, Li Dong, Zhang Zhang, Liang Wang, Tieniu Tan, and Furu Wei. 2025b. Latent sketchpad: Sketching visual thoughts to elicit multimodal reasoning in mllms. *arXiv preprint arXiv:2510.24514*.

Jiarui Zhang, Mahyar Khayatkhoei, Prateek Chhikara, and Filip Ilievski. 2025c. MLLMs know where to look: Training-free perception of small visual details with multimodal LLMs. In *The Thirteenth International Conference on Learning Representations*.

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. 2025. Pyvision: Agentic vision with dynamic tooling. *Preprint*, arXiv:2507.07998.

Runtao Zhou, Giang Nguyen, Nikita Kharya, Anh Totti Nguyen, and Chirag Agarwal. 2025. Improving human verification of llm reasoning through interactive explanation interfaces. *arXiv preprint arXiv:2510.22922*.

Kai Zou, Ziqi Huang, Yuhao Dong, Shulin Tian, Dian Zheng, Hongbo Liu, Jingwen He, Bin Liu, Yu Qiao, and Ziwei Liu. 2025. Uni-MMMU: A massive multi-discipline multimodal unified benchmark. *arXiv preprint arXiv:2510.13759*.

Contents

1 Introduction 1

2 Related Work 2

3 SketchVLM 4

4 Evaluation 4
- 4.1 7 Tasks . . . 4
- 4.2 Setup . . . 5
- 4.3 Baselines . . . 5
- 4.4 Metrics . . . 5

5 Results 5
- 5.1 Grid prompting improves GPT-5's annotation precision, but hurts Gemini-3-Pro-Preview . . . 5
- 5.2 SketchVLMs can localize points and connect them in order . . . 6
- 5.3 SketchVLM improves counting accuracy over text-only baselines . . . 6
- 5.4 SketchVLMs localize objects more accurately with pre-defined shape primitives than with free-form annotations . . . 6
- 5.5 SketchVLM improves localization accuracy for GPT-5 but not for Gemini-3-Pro-Preview when labeling parts of an object . . . 7
- 5.6 SketchVLMs outperform models that were fine-tuned directly on path-tracing tasks . . . 8
- 5.7 Fine-tuned sketching models fail to generalize to unseen physics understanding tasks . . . 8
- 5.8 SketchVLMs produce more faithful and higher-quality annotations than image-editing and fine-tuned models . . . 8
- 5.9 SketchVLM generalizes to open-weight VLMs . . . 9
- 5.10 Single-turn is as accurate as multi-turn but requires significantly fewer turns . . . 9

6 Conclusion 9

A Real World Applications 14

B Dataset Creation 16
- B.1 Connect-the-Dots . . . 16
- B.2 Counting . . . 16
- B.3 Drawing Shapes around Objects . . . 16
- B.4 Part Labeling . . . 16
- B.5 Maze Navigation . . . 16
- B.6 Ball Drop . . . 16

C Additional Task Results 18
- C.1 Combined Results . . . 18
- C.2 Counting . . . 18
- C.3 Connect-the-Dots . . . 19
- C.4 Drawing Shapes around Objects . . . 20
- C.5 Part Labeling . . . 21
- C.6 Additional Model Results . . . 22
- C.7 Latency and Cost . . . 24
- C.8 Ablation Study of Kimi K3 with Different Reasoning Levels . . . 26

D Qualitative Samples 27
- D.1 Ball Drop . . . 28
- D.2 Maze Navigation . . . 31
- D.3 Connect Dots . . . 33
- D.4 Counting . . . 36
- D.5 Drawing Shape . . . 39
- D.6 Part Labeling . . . 42
- D.7 Error Analysis . . . 45
- D.8 Connect-the-Dots: Grid versus No Grid . . . 46
- D.9 Connect-the-Dots: Bézier Curves versus Lines . . . 47
- D.10 Gemini-3-Pro-Preview Coordinate Systems . . . 48
- D.11 Multi-turn Ablation . . . 49

E VLM-Judge Details 50
- E.1 Annotation Score Explanation Examples . . . 51
- E.2 Qualitative Examples . . . 52
- E.3 Rubric Prompts . . . 57

F Model Settings and Prompts 60
- F.1 API Settings . . . 60
- F.2 SketchVLM System Prompt . . . 61
- F.3 SketchVLM Output Example . . . 65

G Other Baselines 66
- G.1 Baselines . . . 66

H AI Use 67

A Real World Applications

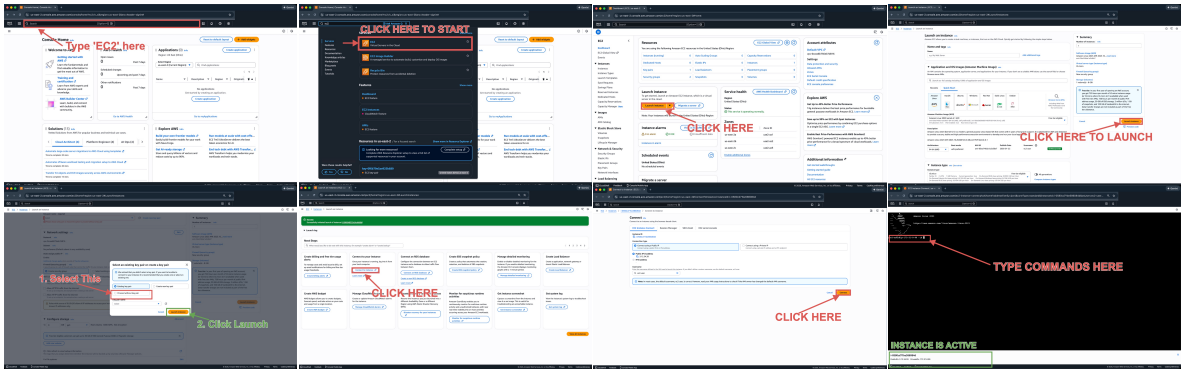


Figure A1: Beginning at the top left image, our SketchVLM framework visually guides the user how to set up a free EC2 instance through the notoriously non-user-friendly Amazon Web Services web interface.

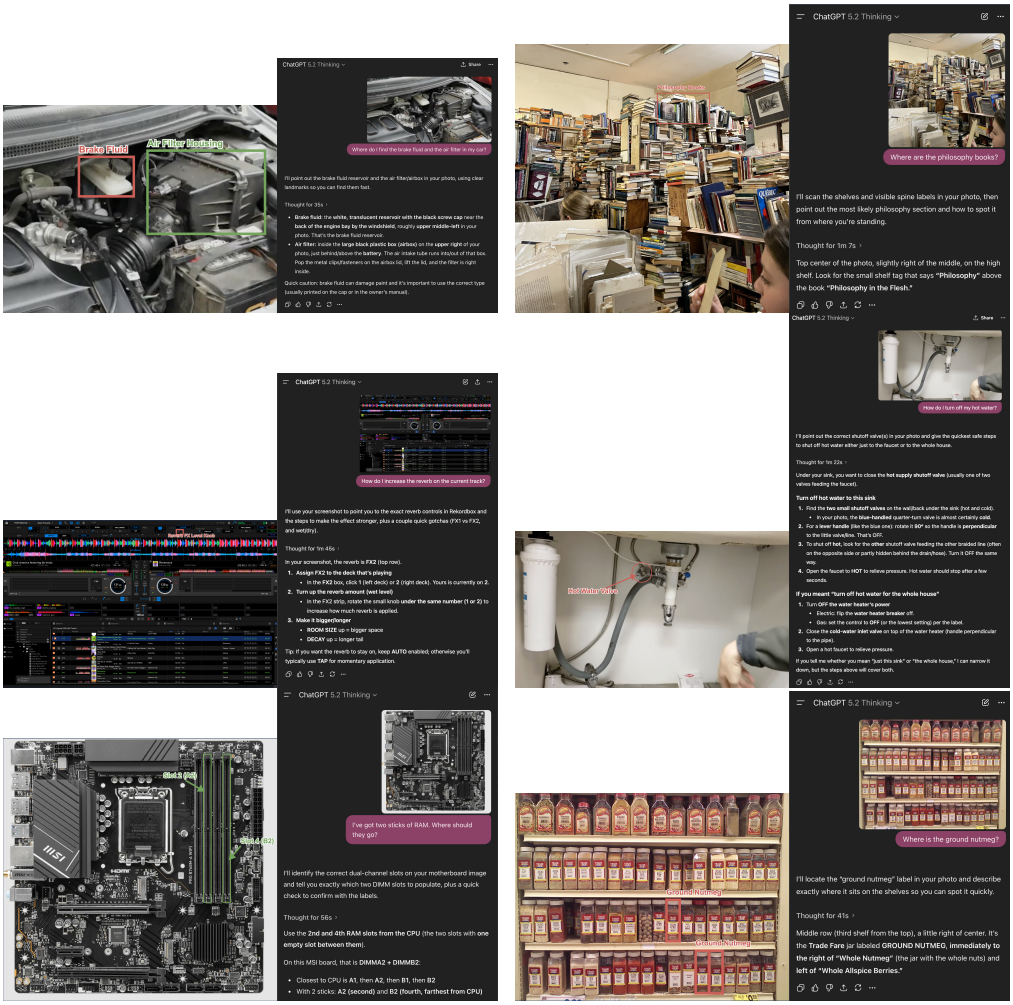
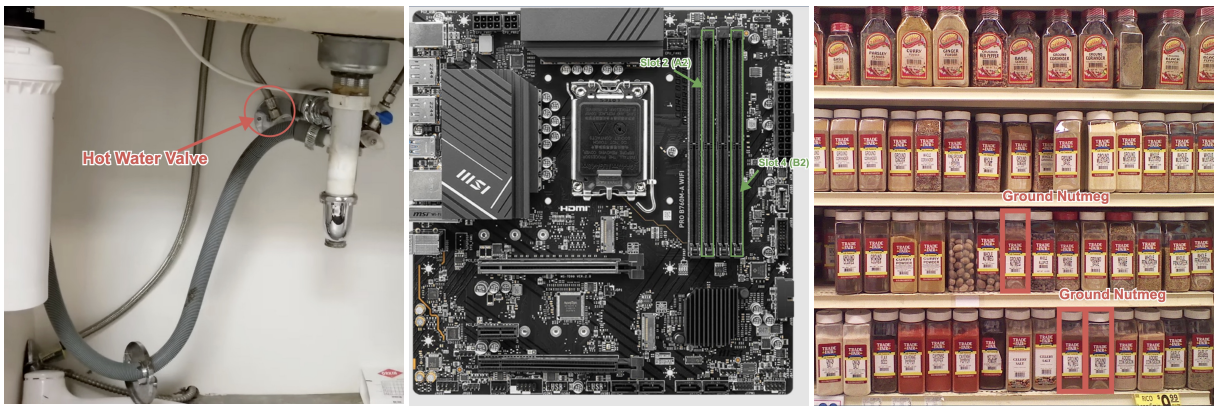


Figure A2: SketchVLM responds via visual annotations while ChatGPT responds with text only.



How do I turn off my hot water?

I've got two sticks of RAM. Where do they go?

Where is the ground nutmeg?

Figure A3: SketchVLM can be used in a variety of real-world use cases.



Figure A4: Multi-turn session where the model explains how to set up PyTorch in a Docker container.

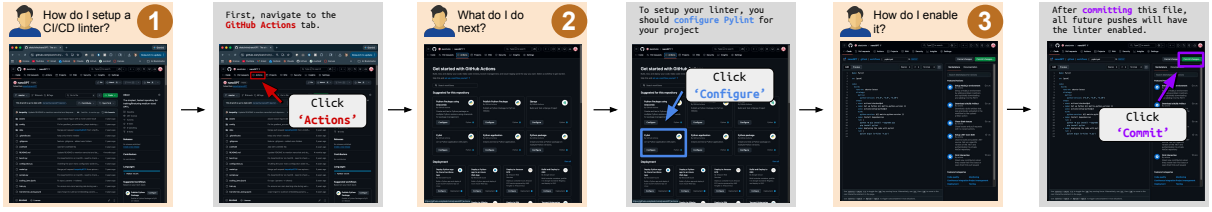


Figure A5: Multi-turn session where the model explains how to set up a CI/CD pipeline in GitHub.

B Dataset Creation

We use publicly available datasets, benchmarks, models, and APIs only for research evaluation. We cite the creators of the datasets, models, and tools used in our experiments in the corresponding dataset, baseline, and model-settings sections. We use these artifacts consistently with their intended research or API use. Our generated benchmarks, code, and demo are intended for research and should not be used for high-stakes decision making without additional validation. For real-world demo screenshots, we manually redact or crop user-specific information before inclusion.

B.1 Connect-the-Dots

We collect 100 connect-the-dots images spanning three subsets with varying dot counts and background clutter.

1. **Random dots:** We use a Python script with the Pillow image library to generate 3 images each with 4, 5...10 dots randomly placed for a total of 21 images. Each dot has a number that corresponds to the order it should be connected.

2. **Outlines:** We convert 30 silhouette SVGs from Openclipart (Openclipart Contributors, 2025) into connect-the-dots puzzles. Each SVG was flattened to polylines, simplified with the Douglas–Peucker algorithm (Douglas and Peucker, 1973), and normalized to a unit square. The main contour was then resampled to place 30 evenly spaced dots along the boundary, with number labels slightly offset. We manually filtered out cases with distorted or self-intersecting shapes, which often occurred for concave outlines.

3. **Worksheets:** We gather 49 images of preexisting connect-the-dot worksheets from online sources. These images differ from the outlined images in how they often contain irrelevant information such as lines at the top of the image for students to mark their name. To obtain the ground truth strokes for these images, we manually annotate the coordinates of each dot.

B.2 Counting

The counting dataset combines three sources: CountBench (Beyer et al., 2024), TallyQA (Acharya et al., 2019), and Pixmo-Count (Deitke et al., 2025). The CountBench and TallyQA subsets together contain 746 samples, covering object counts ranging from 0 to 10. Pixmo-Count contributes 443 samples after removing unsuitable cases from the original 526-image test split and similarly includes object counts from 1 to 10.

B.3 Drawing Shapes around Objects

The testing dataset consists of 1,000 carefully selected images from the 5,000 COCO validation images, chosen to ensure a balanced distribution of object counts across classes and object sizes (small, medium, and large).

B.4 Part Labeling

We carefully selected images from two datasets, PACO (Ramanathan et al., 2023) and Pascal-Part (Chen et al., 2014). The selected images satisfy the following criteria:

1. Each image contains only one object corresponding to the target class name.

2. The object's size occupies at least 10% of the total image area.

3. Each selected object has at least four part labels annotated.

4. The dataset maintains a balanced distribution of objects across different classes.

After selection, the final dataset used for the part labeling task consists of 985 images covering 52 class names.

B.5 Maze Navigation

Given a start point, an end point, and a set of direction commands (e.g., `Up, Down, Left, Right`), the model must determine if the path reaches the goal without crossing any border walls. We create 200 unique 3x3 grids where the shortest path length between the starting green square and ending red square varies from 3 to 8 steps. For each maze, we take the ground truth path and randomly change one of the direction steps in order to make an invalid path (for example, `Left, Right, Down` could be changed to `Right, Right, Down`) (Fig. 6).

B.6 Ball Drop

We evaluate our framework using Visual Physics Comprehension Test (VPCT (cbrower, 2025)), which consists of 100 hand-crafted images where the model must determine which of the buckets

the ball will fall into after it is dropped. While VPCT does provide a simple way to evaluate physics understanding of VLMs, it does not contain any ground truth data of the trajectory of the ball. Therefore, in order to evaluate how well SketchVLM models can draw the true trajectory of the ball paths, we generate our own benchmark (Ball Drop). We simulate the trajectory of the ball using PHYRE (Bakhtin et al., 2019) to obtain ground truth ball trajectory data. In contrast to VPCT, our Ball Drop benchmark is synthetically generated. We generate 198 unique images, with an equal number of images containing 1, 2, and 3 randomly placed lines. We randomize the ball's X position and fix its Y position near the top. There are four containers at the bottom of the image compared to VPCT's three containers, making it harder to guess answers correctly.

C Additional Task Results

C.1 Combined Results

Table C1: SketchVLM improves visual reasoning task accuracy by +28.5 points over alternative sketching approaches. Averages are computed over VPCT, Ball Drop, Maze, and Counting tasks.

Category	Model	VPCT				Avg	Δ
SketchVLM	✦ sketchVLM	96.0	79.7	98.0	94.5	**84.4**	+28.5
	sketchVLM	70.0	68.5	92.8	75.4		
Other Sketching	Nano Banana Pro	63.0	62.6	93.3	91.7	55.9	
	ViLaSR	37.0	35.9	50.8	48.6		
	ThinkMorph	27.0	30.3	62.5	68.1		

Table C2: SketchVLM produces higher quality annotations, scoring +48.3% above alternative sketching approaches on a 1–5 VLM-judged drawing quality scale.

Category	Model	VPCT			Avg	Δ
SketchVLM (Ours)	✦ sketchVLM	3.12	4.28	3.69	**2.98**	+48.3%
	sketchVLM	1.83	1.74	3.20		
Other Sketching	Nano Banana Pro	1.56	2.56	3.68	2.01	
	ViLaSR	1.36	1.28	2.78		
	ThinkMorph	1.62	2.11	1.17		

C.2 Counting

Table C3: ✦ sketchVLM achieves high text-location accuracy, indicating that predicted counts are well aligned with target objects, whereas sketchVLM shows substantially weaker grounding.

Model	Accuracy (%)
✦ sketchVLM	**95.9**
sketchVLM	51.0
ViLaSR	59.9
sketchVLM	70.4

Table C4: Counting accuracy for SketchVLM and Direct VQA across Gemma-4-31B, Qwen3.5-397B, Kimi K2.5, and Kimi K3.

Model	SketchVLM	Direct VQA
Gemma-4-31B	**55.11**	36.33
Qwen3.5-397B	58.15	**74.31**
Kimi K2.5	63.26	**66.85**
Kimi K3	**93.65**	86.19

Kimi K3 is the strongest open-source backbone for Counting. We compare SketchVLM and Direct VQA on Counting across four open-source backbones (Tab. C4). Kimi K3 is best under both setups, reaching 93.65 with SketchVLM, 19.34 points above the best configuration of any other model. Its lead also widens under SketchVLM, from 11.88 to 30.39 points over the runner-up. The gain is not universal: SketchVLM adds 18.78 points on Gemma-4-31B but costs 16.16 on Qwen3.5-397B and 3.59 on Kimi K2.5. The best open-source configuration is therefore SketchVLM with Kimi K3, while weaker backbones require per-model validation.

C.3 Connect-the-Dots

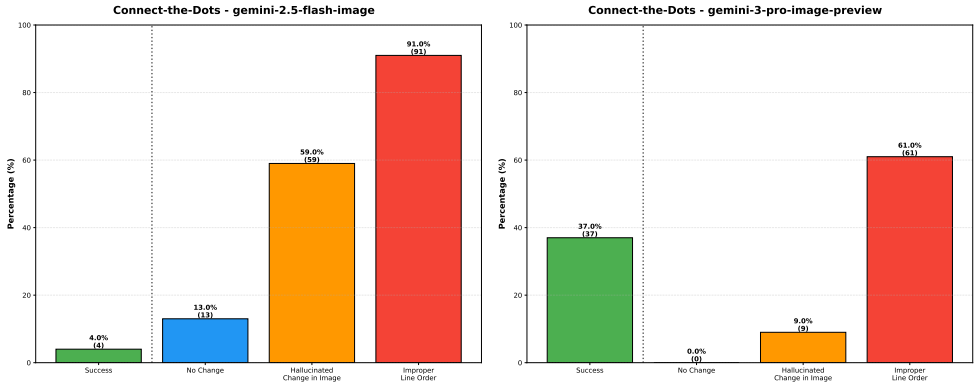


Figure C1: Gemini-2.5-Flash-Image frequently adds non-existent points to the image or does not correctly follow the proper order of the connected dots. Nano Banana Pro is significantly better than Gemini-2.5-Flash-Image, but still only completes the task without any errors 37% of the time.

	Random Dots (n=21)			Worksheets (n=49)			Outlines (n=30)			Total (n=100)		
Model	No Grid	With Grid	Δ	No Grid	With Grid	Δ	No Grid	With Grid	Δ	No Grid	With Grid	Δ
Gemini-2.5-Pro	32.1	**20.7**	-35.51%	64.5	**30.9**	-52.09%	82.3	**34.6**	-57.96%	61.1	**29.5**	-51.72%
Gemini-3-Pro	**2.4**	23.2	+866.67%	**5.0**	31.9	+538.00%	**4.7**	65.1	+1285.11%	**4.6**	33.2	+621.74%
GPT-5 (low)	90.8	**19.7**	-78.30%	109.4	**37.7**	-65.54%	124.4	**54.7**	-56.03%	110.5	**40.1**	-63.71%
Qwen3.5-397B-A17B	**18.4**	20.7	+12.50%	44.7	**27.8**	-37.81%	39.4	**27.1**	-31.22%	34.2	**26.5**	-22.51%
Kimi K2.5	22.1	**18.3**	-17.19%	56.7	**44.5**	-21.52%	111.9	**45.7**	-59.16%	57.7	**40.2**	-30.33%
Kimi K3	1.1	–	–	4.1	–	–	3.2	–	–	3.3	–	–
Gemma-4-31B	**5.8**	20.5	+253.45%	**24.7**	34.1	+38.06%	**27.7**	39.7	+43.32%	**23.7**	33.9	+43.04%
ViLaSR	154.8	–	–	155.9	–	–	188.8	–	–	175.5	–	–

Figure C2: **Connect-the-Dots median per-sample RMSE with categories as columns.** Each entry shows the median per-sample RMSE without and with the grid, and the percent change Δ (negative is better). Gemini-3-Pro-Preview and Gemma-4-31B perform better without the grid which we hypothesize about in Fig. D24, while the models that are worse at localizing benefit from the grid, including GPT-5, Kimi K2.5, and Qwen3.5-397B-A17B.

C.4 Drawing Shapes around Objects

Table C5: Sketch-based localization improves accuracy for medium and large objects, while reducing performance on small objects, compared to coordinate-based bounding boxes (AP50).

	✦	✦ sketchVLM	✦ sketchVLM
Output format	Bounding box	Rectangle	Oval
AP50 (all)	**63.1**	58.8 (-4.3)	55.4 (-7.7)
AP50 (small)	**35.6**	25.5 (-10.1)	26.4 (-9.2)
AP50 (medium)	62.6	**63.2** (+0.6)	55.4 (-7.2)
AP50 (large)	77.7	**79.1** (+1.4)	77.6 (-0.1)

Table C6: ✦ and ✦ SketchVLM achieve the same precision; however, SketchVLM has lower recall than the baseline because it produces fewer true positive (TP) detections.

GT	✦			✦ SketchVLM			TP		FP		FN	
	P	R	AP	P	R	AP	✦	✦ sketchVLM	✦	✦ sketchVLM	✦	✦ sketchVLM
1060	18.7	47.9	33.1	18.7	35.4	26.0	**508**	375	2204	**1629**	**552**	685

Table C7: Different prompt variations do not improve AP performance for the drawing shape task.

	✦	✦ SketchVLM Draw rectangles	✦ SketchVLM Draw all visible/part visible	✦ SketchVLM Scan region before draw	✦ SketchVLM Count object before draw
Output shape	Bounding box	Rectangles	Rectangles	Rectangles	Rectangles
AP50 (all)	**63.1**	58.8	57.1	59.0	56.2
AP50 (small)	**35.6**	25.5	27.5	26.8	23.2
AP50 (medium)	62.6	**63.2**	60.7	59.9	58.9
AP50 (large)	77.7	79.1	77.8	**80.7**	78.3

Table C8: SketchVLM improves drawing-shape localization for Gemma-4-31B, while Direct VQA performs better for Qwen3.5-397B and Kimi K2.5.

	Gemma-4-31B SketchVLM	Gemma-4-31B Direct VQA	Qwen3.5-397B SketchVLM	Qwen3.5-397B Direct VQA	Kimi K2.5 SketchVLM	Kimi K2.5 Direct VQA	Kimi K3 SketchVLM	Kimi K3 Direct VQA
AP50 (all)	50.70	41.01	45.43	49.71	29.36	45.21	46.02	**55.22**
AP50 (small)	12.09	12.64	11.68	**19.31**	4.52	10.44	8.96	15.55
AP50 (medium)	**52.51**	42.14	43.34	48.53	25.67	44.65	40.72	51.26
AP50 (large)	75.78	53.35	73.03	67.40	51.99	65.71	71.05	**76.84**

Direct VQA is the stronger setup on Drawing Shape except for Gemma-4-31B. We compare both setups on Drawing Shape across four open-source backbones (Tab. C8). SketchVLM helps only Gemma-4-31B, raising AP50 by 9.69 points overall and letting the smallest model outperform Qwen3.5-397B and Kimi K2.5 in both setups. For the larger backbones it is harmful, costing 4.28, 15.85, and 9.20 points on Qwen3.5-397B, Kimi K2.5, and Kimi K3. The gain also depends on object scale: Gemma-4-31B improves by 22.43 points on large objects but not on small ones (-0.55), and every backbone loses accuracy on small objects under SketchVLM. Explicit intermediate representations therefore substitute for capability the backbone lacks rather than adding to strong backbones, and do not help at small object scales. The best configuration remains Direct VQA with Kimi K3 at 55.22 AP50.

C.5 Part Labeling

Table C9: Error type breakdown for labeling and part labeling with ✦ and original models and SketchVLM.

Wrong Type	✦ (%)	✦ SketchVLM(%)	(%)	SketchVLM(%)
Missing Label	**10.8**	3.6	**0.69**	0.32
Wrong Position	25.1	**36.1**	**80.17**	79.32
Total	35.9	**39.7**	**80.86**	79.64

Table C10: Labels placed by SketchVLMs land very close to the correct location. sketchVLM is more accurate than the baseline at every tolerance level, and ✦ sketchVLM matches it within a few pixels.

Dilation r (px)	✦	✦ sketchVLM		sketchVLM
0	**64.1**	60.3 (-3.8)	19.1	**20.4** (+1.3)
3	**69.0**	66.5 (-2.5)	21.9	**23.3** (+1.4)
5	**70.8**	69.3 (-1.5)	23.9	**25.3** (+1.4)
7	**71.8**	71.2 (-0.6)	25.6	**27.2** (+1.6)
10	73.1	**73.5** (+0.4)	28.6	**30.2** (+1.6)
15	75.1	**78.3** (+3.2)	33.7	**35.9** (+2.2)

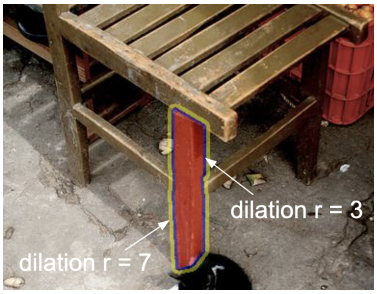


Figure C3: From $r = 0$ (red) to modest dilation ($r = 7$, yellow), small boundary offsets become visually negligible.

Table C11: Part-labeling accuracy across different dilation radii for Gemma-4-31B, Qwen3.5-397B, Kimi K2.5, and Kimi K3.

Dilation r (px)	Gemma-4-31B SketchVLM	Gemma-4-31B Direct VQA	Qwen3.5-397B SketchVLM	Qwen3.5-397B Direct VQA	Kimi K2.5 SketchVLM	Kimi K2.5 Direct VQA	Kimi K3 SketchVLM	Kimi K3 Direct VQA
0	48.31	**50.12**	38.53	**46.44**	23.31	**42.94**	45.46	**51.34**
3	54.19	**55.30**	44.43	**50.85**	26.84	**47.64**	51.13	**56.99**
5	57.74	**57.97**	48.49	**52.83**	29.44	**50.20**	54.47	**60.32**
7	**60.44**	59.91	51.50	**54.42**	31.61	**52.12**	56.73	**62.63**
10	**63.74**	62.16	56.10	**56.29**	35.10	**54.42**	60.07	**66.10**
15	**67.90**	64.70	**62.36**	58.71	40.79	**57.61**	63.80	**70.13**

Direct VQA is the stronger setup on Part Labeling. We compare both setups on Part Labeling across four open-source backbones at dilation radii $r \in \{0, \ldots, 15\}$ px (Tab. C11). Direct VQA is better in 20 of 24 settings, and Kimi K3 with Direct VQA is the best configuration at every radius, reaching 70.13 at $r = 15$. The deficit of SketchVLM is largest for Kimi K2.5 (-19.6 points at $r = 0$) and smallest for Gemma-4-31B (-1.8), and it narrows as r increases for Gemma-4-31B and Qwen3.5-397B while remaining constant for the two Kimi models. Unlike Counting and Drawing Shape, Part Labeling shows no backbone for which SketchVLM is clearly preferable.

C.6 Additional Model Results

Table C12: Ablation across inputs in single-turn mode for additional models. "Sketch" adds strokes/system prompt; "Grid" additionally overlays the coordinate grid. Median per-sample RMSE is reported for Connect-the-Dots while accuracy is reported for the other tasks. "Order Accuracy" is the percentage of connect-the-dots samples with correct point ordering (higher is better).

Model	Input	VPCT				Order Accuracy
Kimi K2.5	Image	44.0	50.50	**93.5**	–	–
	+ Grid	55.0	**58.08**	97.0	–	–
	+ Sketch	**56.0**	48.48	88.00	57.7	59%
	+ Sketch + Grid	**56.0**	50.50	89.75	**40.2**	67%
Gemma-4-31B	Image	38.0	38.90	90.5	–	–
	+ Grid	44.0	**44.40**	86.0	–	–
	+ Sketch	46.0	40.90	89.0	**23.7**	56%
	+ Sketch + Grid	**51.0**	39.40	**93.8**	33.9	**62%**
Qwen3.5-397B-A17B	Image	**83.00**	**69.19**	96.75	–	–
	+ Grid	76.00	67.67	**97.50**	–	–
	+ Sketch	81.00	61.11	96.00	34.2	64.00%
	+ Sketch + Grid	79.00	61.62	95.75	**26.5**	**87.00%**
Kimi K3	Image	**95.0**	**85.35**	**100**	–	–
	+ Sketch	92.0	79.29	99.75	**3.3**	97.62%

Table C13: Average number of turns per task group in multi-turn evaluation.

Task	**Gemini-3-Pro**	**GPT-5 (low)**
VPCT	5.10	6.18
	5.00	2.58
	16.61	19.00
	3.42	4.29
Overall	**5.91**	**5.92**

Table C14: Accuracy comparison between SketchVLM and Direct VQA across seven tasks using Gemma-4-31B, Qwen3.5-397B, Kimi K2.5, and Kimi K3. Drawing Shape reports AP50 (all) for rectangle localization, Part Labeling reports accuracy at dilation radius $r = 0$, and Connecting Dots reports median per-sample RMSE (lower is better).

Model	Setup	VPCT Video Physics Acc. (%)	Ball Drop Acc. (%)	Maze Acc. (%)	Counting Acc. (%)	Part Labeling Acc. (%)	Drawing Shape AP50 (%)	Connecting Dots Median RMSE ↓
Gemma4-31B	SketchVLM	46.00	40.90	89.00	55.11	48.31	50.70	23.7
	Direct VQA	38.00	38.90	90.50	36.33	50.12	41.01	–
Qwen3.5-397B	SketchVLM	81.00	61.11	96.00	58.15	38.53	45.43	34.2
	Direct VQA	83.00	69.19	96.75	74.31	46.44	49.71	–
Kimi K2.5	SketchVLM	56.00	48.48	88.00	63.26	23.31	29.36	57.7
	Direct VQA	44.00	50.50	93.50	66.85	42.94	45.21	–
Kimi K3	SketchVLM	92.00	79.29	99.75	93.65	45.46	46.02	3.3
	Direct VQA	95.00	85.35	100.00	86.19	51.34	55.22	–

Bold indicates the best result in each column. Higher is better for all metrics except Connecting Dots median RMSE (↓). Drawing Shape uses AP50 (all) for rectangle localization, and Part Labeling uses dilation radius $r = 0$.

Generalization to open-source models. Since our main results use closed-source models, we further evaluate SketchVLM with four open-source models to assess whether its benefits generalize across different backbones. Among them, Kimi K3 achieves the strongest overall performance. The benefit of SketchVLM is task-dependent. On Counting, SketchVLM improves over Direct VQA by 18.8 points with Gemma4-31B and 7.5 points with Kimi K3. It also enables Connecting Dots, which Direct VQA cannot address because it does not produce the required coordinate-based outputs. In contrast, Direct VQA performs better on Part Labeling across all four models and on Drawing Shape for all but Gemma-4-31B. Overall, these results show that SketchVLM generalizes to open-source backbones, with the largest gains on tasks that benefit from explicit intermediate visual representations.

C.7 Latency and Cost

Table C15: Average latency and cost per item for Direct VQA and Sketch + Grid across different models on VPCT and Maze Navigation.

Model	Method	Avg. Latency (s/item)	Avg. Cost ($/item)
GPT-5	Direct VQA	24.87 ± 20.52	0.014225 ± 0.005350
	Sketch + Grid	31.24 ± 30.67	0.021859 ± 0.006280
Gemini-3.1-Pro	Direct VQA	27.13 ± 16.94	0.044237 ± 0.038279
	Sketch + Grid	14.33 ± 4.81	0.029351 ± 0.006184
Gemma-4-31B	Direct VQA	9.59 ± 6.98	0.000399 ± 0.000069
	Sketch + Grid	10.11 ± 6.24	0.000397 ± 0.000070
Qwen3.5-397B	Direct VQA	229.6 ± 203.8	0.03461 ± 0.02075
	Sketch + Grid	212.1 ± 110.4	0.03685 ± 0.02083

Table C16: Average latency and running cost across Counting, Drawing Shape, and Part Labeling tasks.

Task	**Model**	**Setup**	**Avg. Latency (s/item)**	**Avg. Cost ($/item)**
Counting	Gemma4-31B	SketchVLM	1.874 ± 1.18	0.000730 ± 0.00
		Direct VQA	0.616 ± 0.52	0.000064 ± 0.00
	Qwen3.5-397B	SketchVLM	24.916 ± 28.93	0.008143 ± 0.01
		Direct VQA	26.273 ± 59.52	0.006952 ± 0.02
	Kimi K2.5	SketchVLM	14.659 ± 19.08	0.003915 ± 0.00
		Direct VQA	5.611 ± 13.17	0.001291 ± 0.00
Drawing Shape	Gemma4-31B	SketchVLM	3.434 ± 13.74	0.000936 ± 0.00
		Direct VQA	0.885 ± 1.00	0.000198 ± 0.00
	Qwen3.5-397B	SketchVLM	36.307 ± 41.85	0.011462 ± 0.01
		Direct VQA	9.676 ± 25.00	0.002588 ± 0.01
	Kimi K2.5	SketchVLM	34.220 ± 26.93	0.008392 ± 0.01
		Direct VQA	36.106 ± 31.54	0.007568 ± 0.01
Part Labeling	Gemma4-31B	SketchVLM	2.556 ± 1.46	0.000878 ± 0.00
		Direct VQA	1.085 ± 0.66	0.000190 ± 0.00
	Qwen3.5-397B	SketchVLM	55.898 ± 32.03	0.018339 ± 0.01
		Direct VQA	18.406 ± 33.30	0.004773 ± 0.01
	Kimi K2.5	SketchVLM	47.406 ± 24.82	0.011871 ± 0.00
		Direct VQA	55.227 ± 36.94	0.011977 ± 0.01

Table C17: Average latency and cost of Kimi K3 with SketchVLM and Direct VQA across seven datasets. Lower is better for both metrics.

Metric	**Setup**	**VPCT** Video Physics	Ball Drop	Maze	Counting	Part Labeling	Drawing Shape	Connecting Dots
Latency (s) ↓	SketchVLM	155.47 ± 85.09	$\mathbf{112.26 \pm 87.93}$	60.25 ± 31.07	40.31 ± 25.89	54.00 ± 36.25	24.47 ± 19.75	40.32 ± 25.48
	Direct VQA	$\mathbf{101.33 \pm 47.19}$	124.55 ± 73.83	$\mathbf{23.60 \pm 10.35}$	$\mathbf{6.99 \pm 8.62}$	$\mathbf{14.45 \pm 10.02}$	$\mathbf{9.20 \pm 6.80}$	–
Cost ($) ↓	SketchVLM	$\mathbf{0.13 \pm 0.05}$	$\mathbf{0.10 \pm 0.05}$	0.05 ± 0.01	0.03 ± 0.01	0.04 ± 0.01	0.02 ± 0.02	0.04 ± 0.02
	Direct VQA	0.15 ± 0.06	0.15 ± 0.08	$\mathbf{0.03 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	–

Overhead of SketchVLM. SketchVLM issues additional generation calls, so we measure per-item latency and cost against Direct VQA under the same model and task set. On tasks where a Direct VQA baseline exists, SketchVLM is uniformly slower, by $1.5\times$ on VPCT up to $5.8\times$ on Counting, since the sketch is produced before the answer. Cost, however, does not follow latency. On the short tasks the sketching call dominates and cost increases by 1.5–3×, whereas on the VPCT and Ball Drop tasks SketchVLM is cheaper than Direct VQA despite comparable latency ($0.125 vs. $0.154 on VPCT and $0.098 vs. $0.149 on Ball Drop). Connecting Dots is excluded from this com-

parison, as Direct VQA does not produce the required coordinate outputs. The overhead is therefore concentrated on short tasks with small input contexts, and is smallest, or negative in cost, on tasks with large visual inputs.

C.8 Ablation Study of Kimi K3 with Different Reasoning Levels

Table C18: Comparison of Kimi K3 across different reasoning levels. Bold indicates the best result for each metric. ↑ indicates higher is better, while ↓ indicates lower is better. Cost and latency are averaged per sample.

Metric	**Reasoning Level**	VPCT			
		Video Physics	Part Labeling	Drawing Shape	Connecting Dots
		Acc. (%) ↑	Acc. (%) ↑	AP50 (%) ↑	RMSE ↓
Performance	Low	86.00	39.29	55.51	6.6
	High	**94.00**	45.81	57.08	**4.1**
	Max	91.80	**46.93**	**57.70**	4.2
Cost ($) ↓	Low	**0.044**	**0.017**	**0.022**	**0.025**
	High	0.123	0.030	0.030	0.042
	Max	0.227	0.035	0.082	0.086
Latency (s) ↓	Low	**52.648**	**13.190**	**11.015**	**19.570**
	High	140.813	19.070	16.185	39.600
	Max	208.245	18.794	69.265	86.230

Accuracy–cost trade-off across reasoning levels. Reasoning models expose an inference-time budget that increases accuracy at the expense of cost and latency, so the practical question is not which level is most accurate but which level offers the best trade-off. We therefore evaluate Kimi K3 at three reasoning levels (Low, High, and Max), keeping the model, prompts, and evaluation protocol fixed, and report per-sample cost and end-to-end latency alongside task performance. Low is clearly insufficient, trailing High by 8.0 points on VPCT and 6.5 points on Part Labeling. High matches or outperforms Max across all four tasks: it is within 1.1 points on Part Labeling and 0.6 points on Drawing Shape, and is in fact 2.2 points better on VPCT and achieves a lower median per-sample RMSE on Connecting Dots (4.1 versus 4.2, worksheets subset only), while requiring up to $2.7\times$ less cost and $4.3\times$ less latency. Unlike the other tasks, additional reasoning budget does not help fine-grained coordinate prediction: Max's median RMSE is no better than High's, despite roughly $2\times$ the cost and latency on Connecting Dots specifically. Overall, High offers the best accuracy–cost trade-off across all four tasks, and we adopt it as our default; none of the tasks in this ablation justify the added expense of Max.

D Qualitative Samples

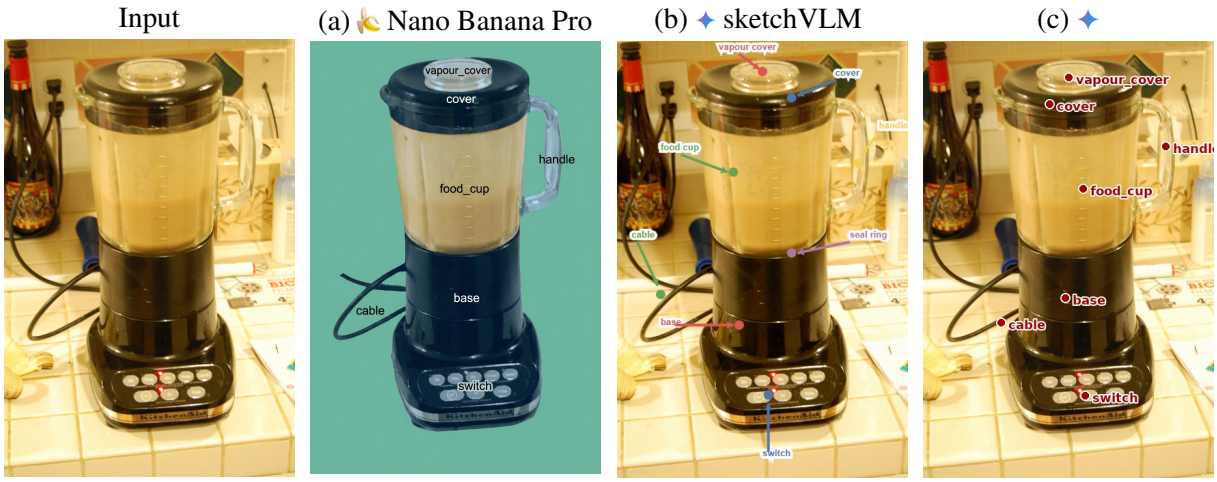


Figure D1: Qualitative comparison on the part labeling task. (b) sketchVLM places each part label directly on its corresponding region while preserving the original image, producing more interpretable part annotations than (a) Nano Banana Pro or (c) Gemini-3-Pro-Preview.

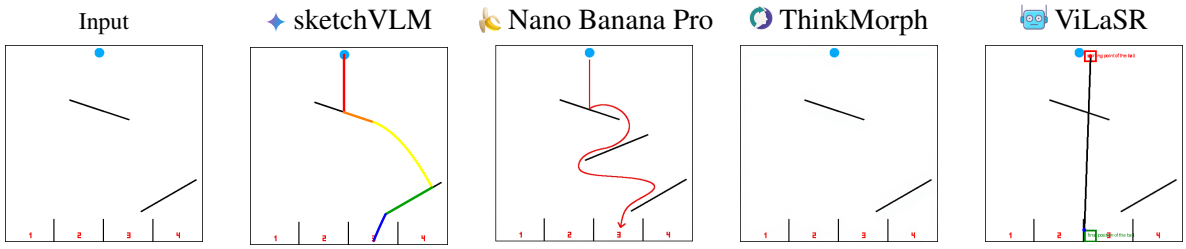


Figure D2: SketchVLM generates the most accurate Ball Drop images compared to other baselines.

D.1 Ball Drop



Figure D3: Qualitative examples of models on VPCT. Gemini-3-Pro-Preview in single-turn produces the most accurate annotations, while NanoBanana Pro, ThinkMorph and ViLaSR often draw paths that cross walls and provide the wrong answer.

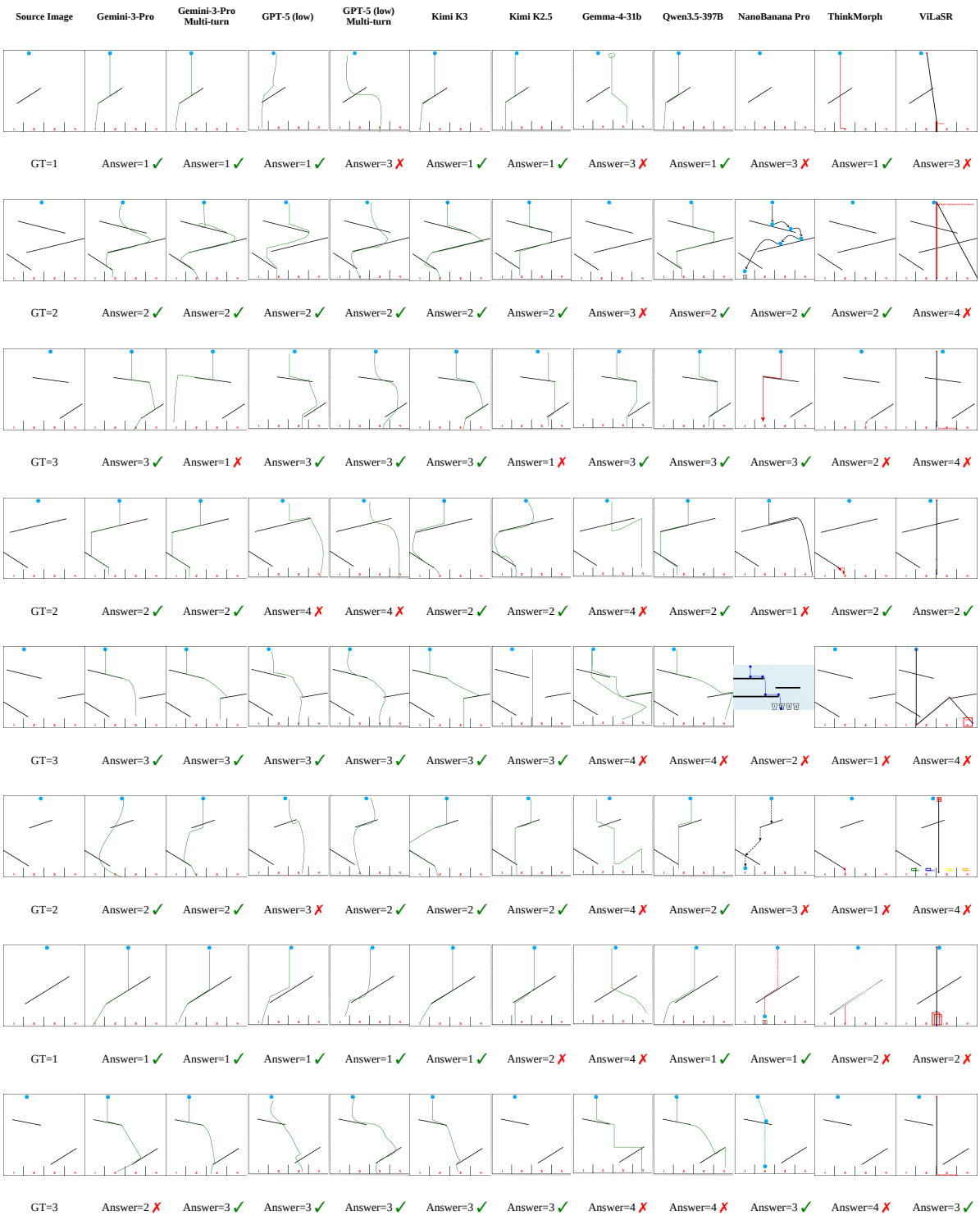


Figure D4: Qualitative examples of models on our Ball Drop dataset. Gemini-3-Pro-Preview in single-turn produces the most accurate annotations, while NanoBanana Pro, ThinkMorph and ViLaSR often draw paths that cross walls and provide the wrong answer.

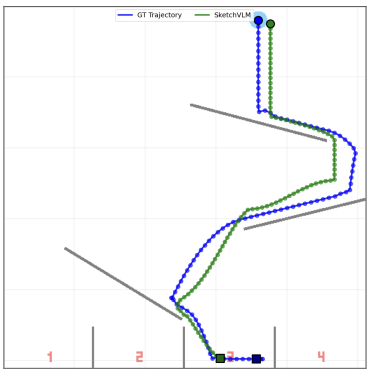


Figure D5: The SketchVLM framework boosts VLM accuracy on the Ball Drop task while also letting them draw ball trajectory paths that closely simulate the ground truth data.

D.2 Maze Navigation

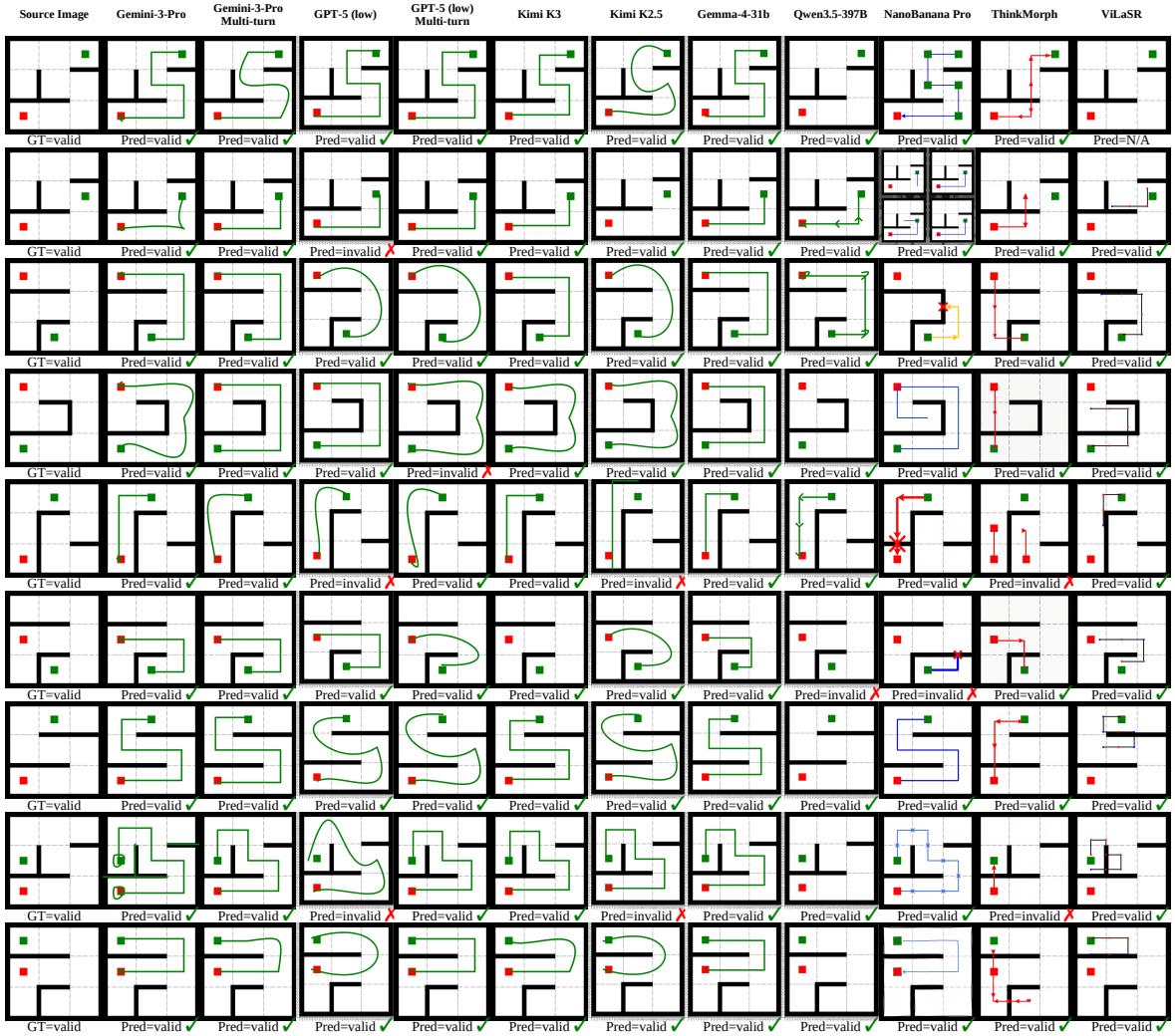


Figure D6: Qualitative examples of models on valid paths in Maze Navigation.

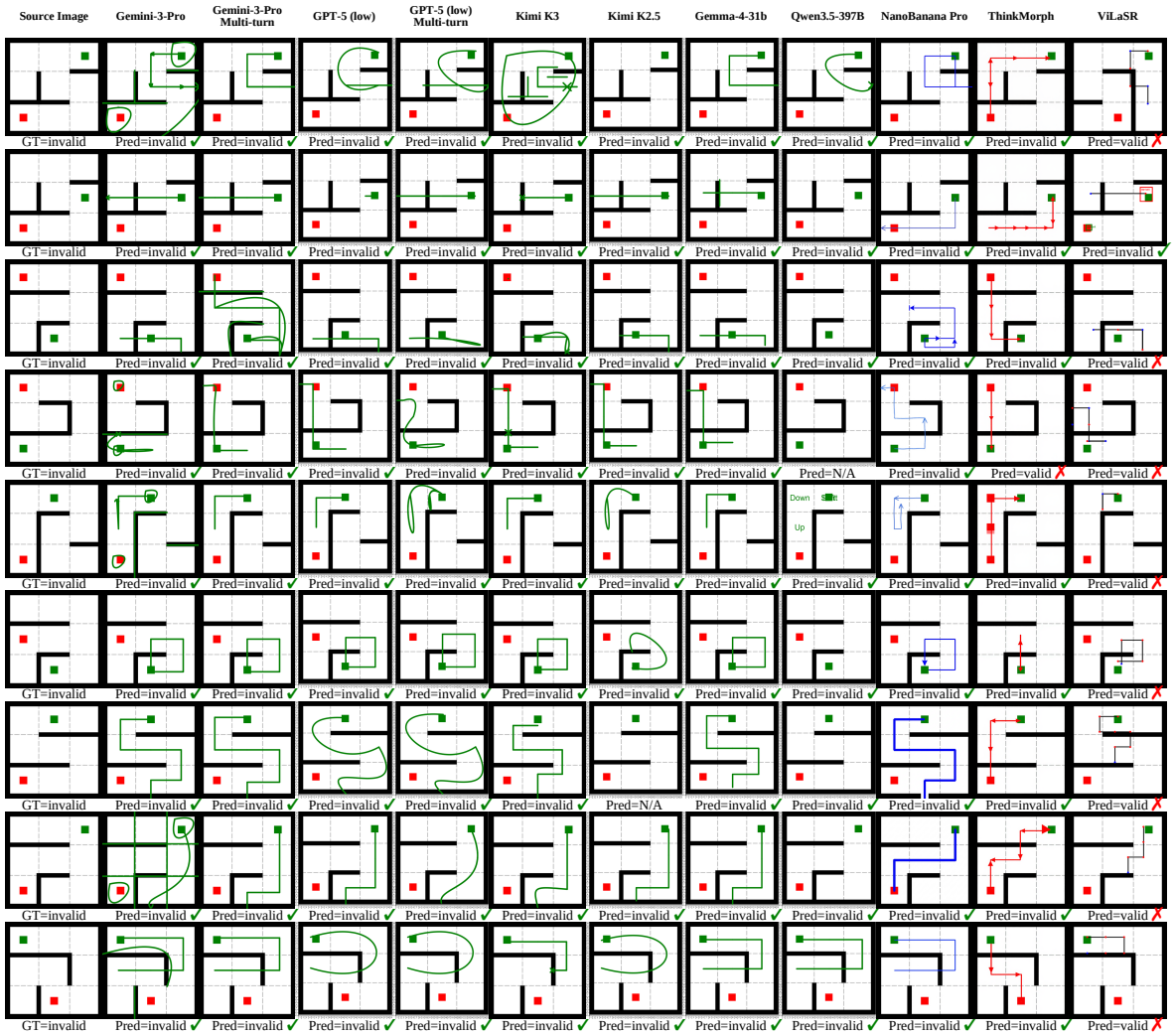


Figure D7: Qualitative examples of models on invalid paths in Maze Navigation.

D.3 Connect Dots

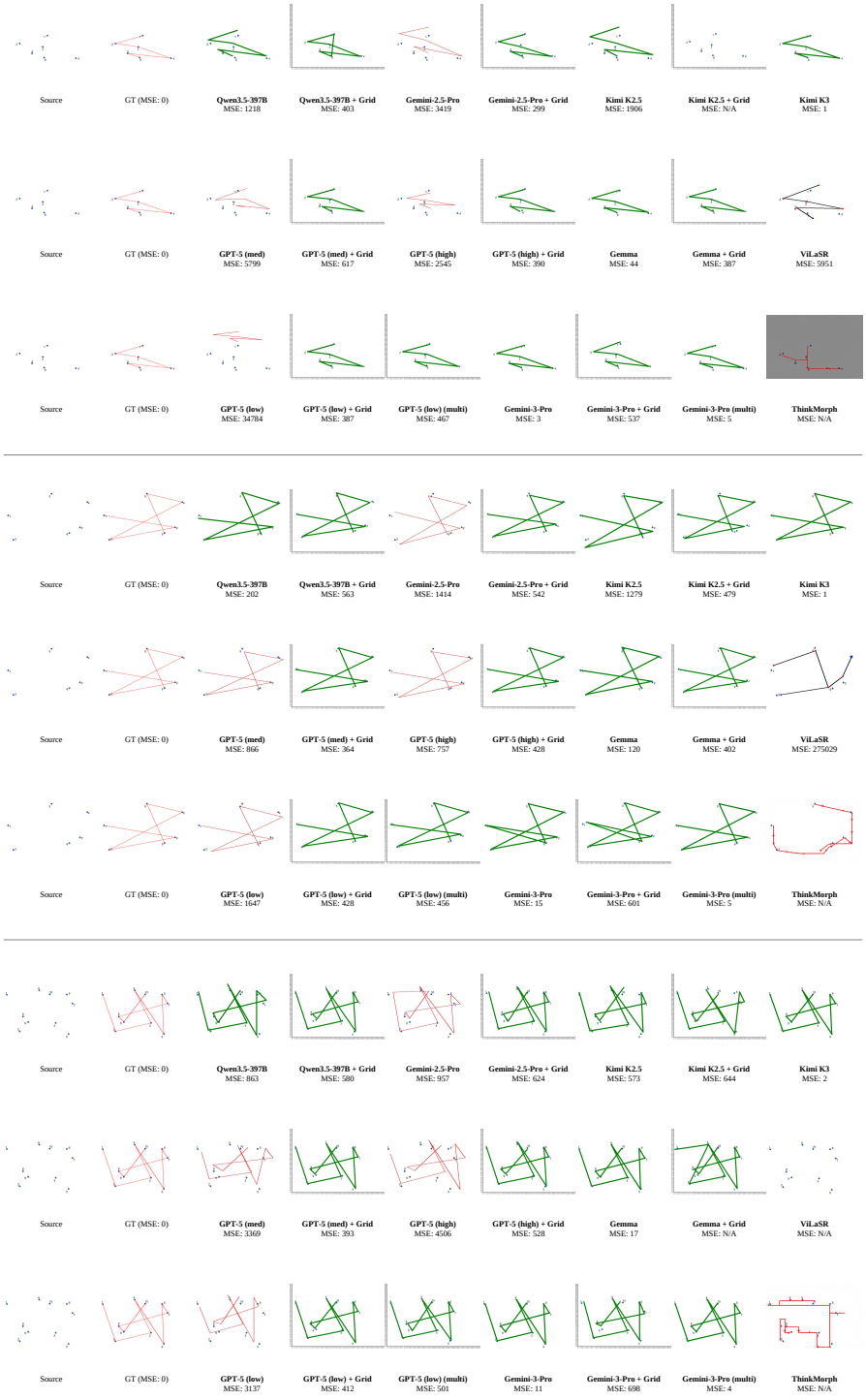


Figure D8: Connect-the-Dots qualitative comparisons for the random dots subset. Each cell shows the overlay and MSE.

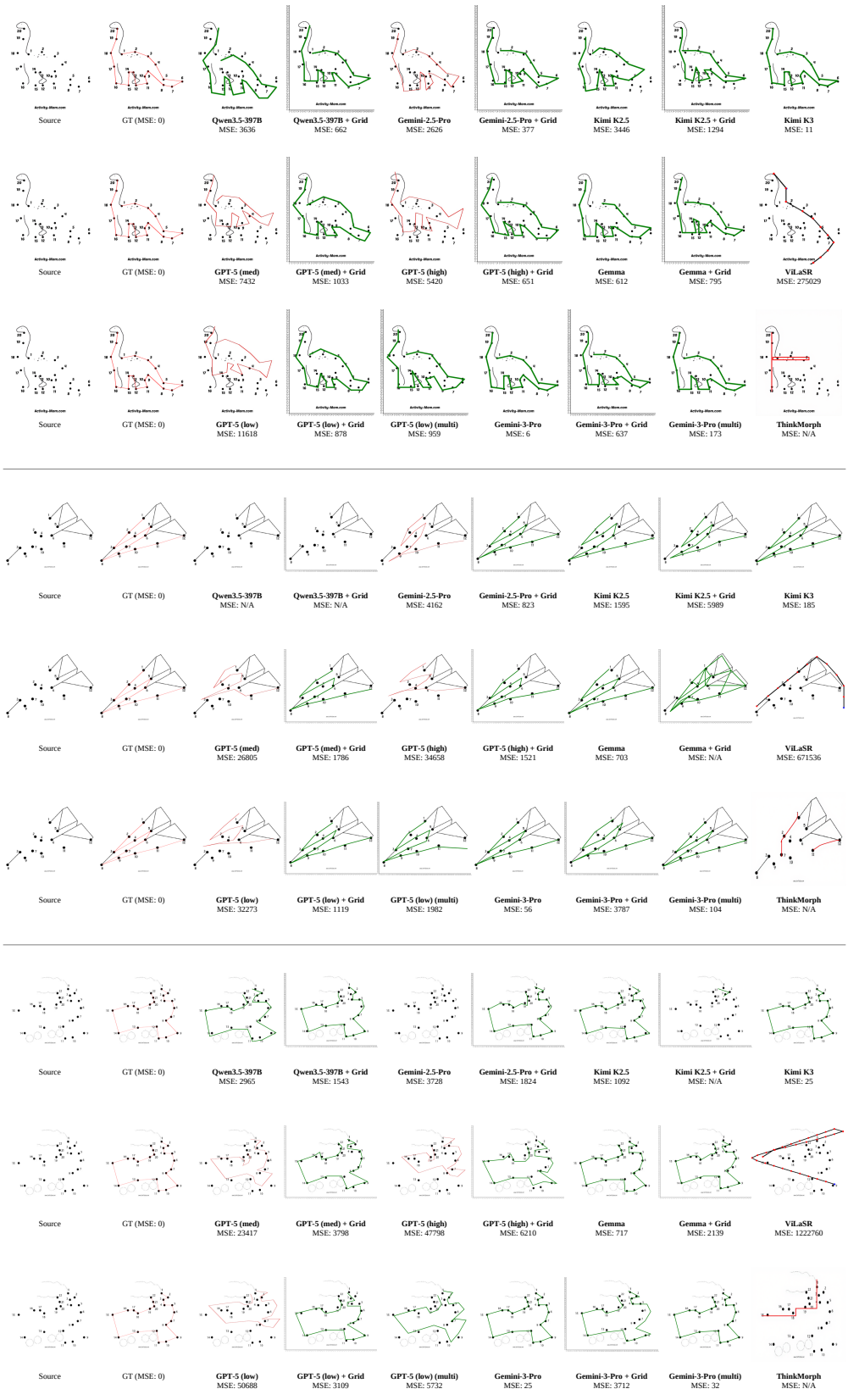


Figure D9: Connect-the-Dots qualitative comparisons for the worksheets subset.

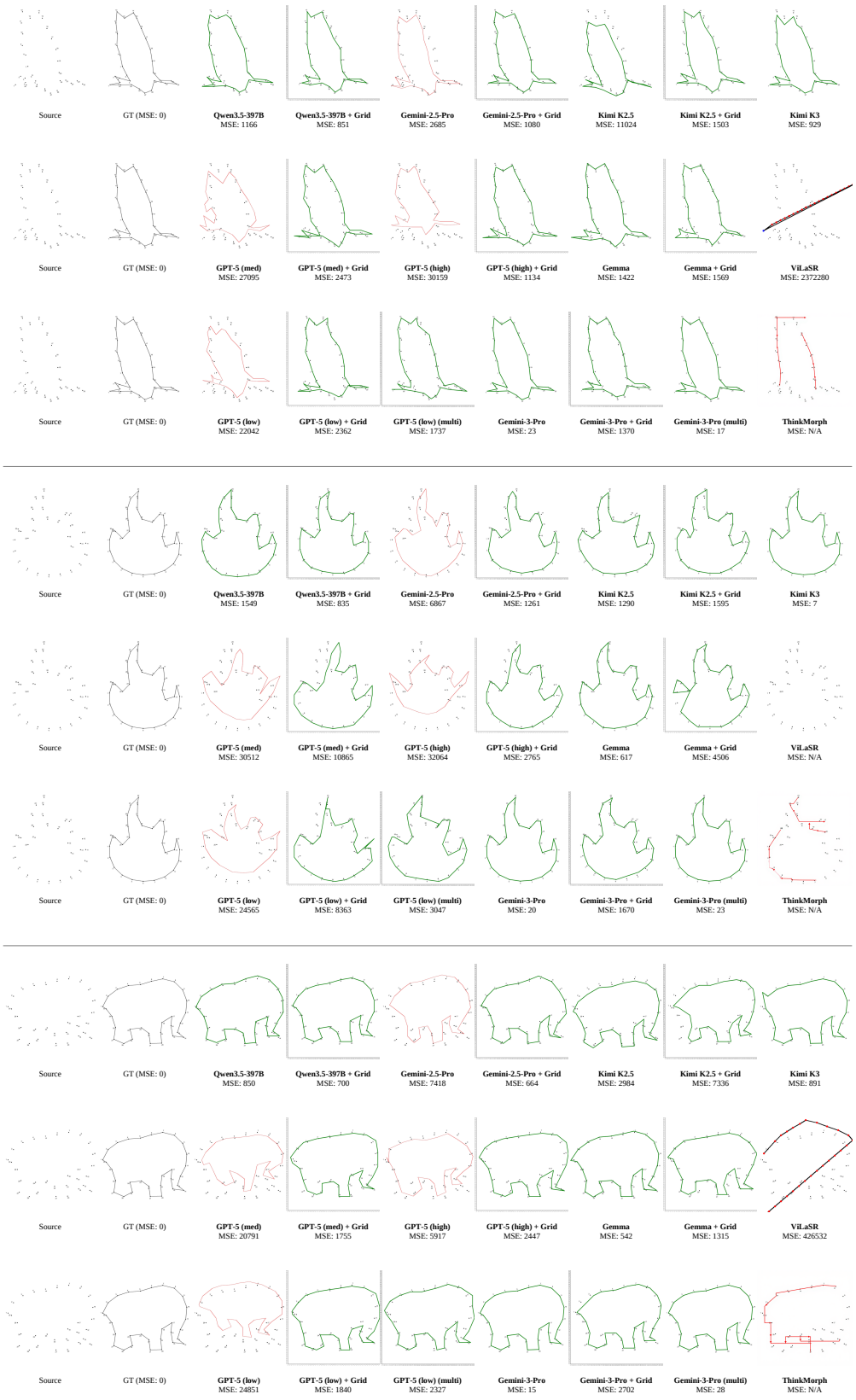


Figure D10: Connect-the-Dots qualitative comparisons for the outlines subset.

D.4 Counting

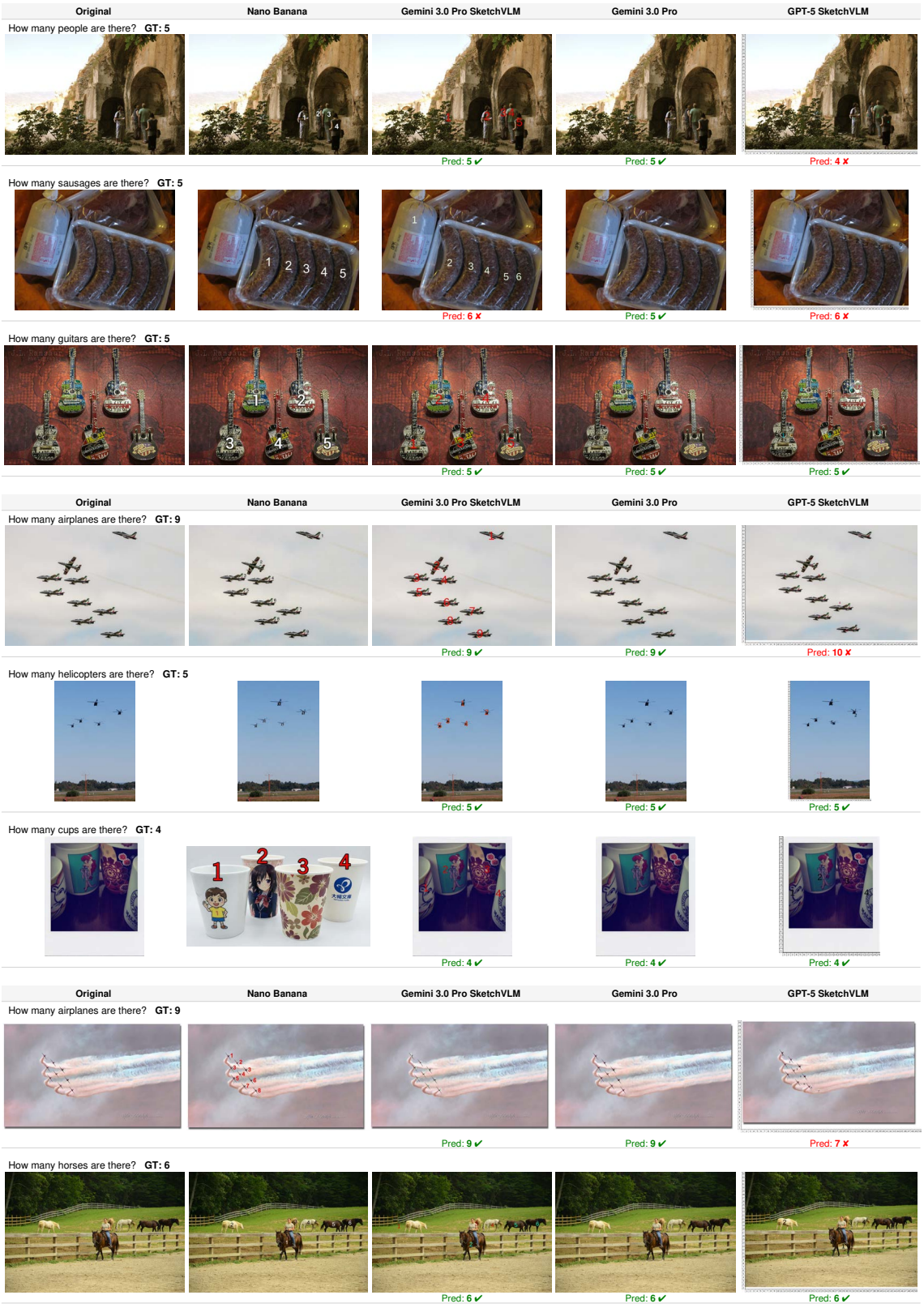


Figure D11: Qualitative comparison on counting task between Gemini-3-Pro-Preview, ✦ sketchVLM , sketchVLM, and Nano Banana Pro.

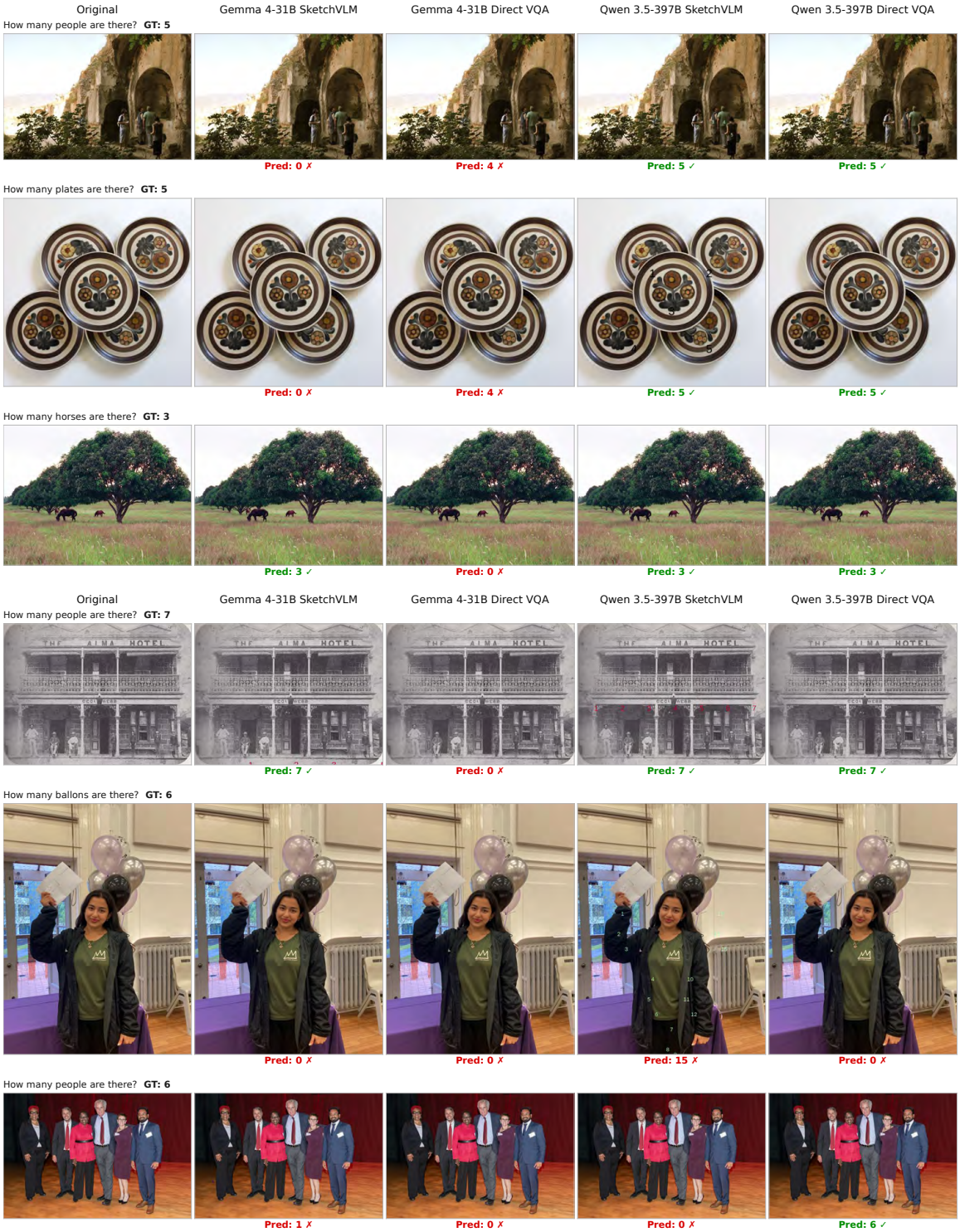


Figure D12: Qualitative comparison on the counting task between Gemma-4-31B and Qwen3.5-397B using SketchVLM and Direct VQA.

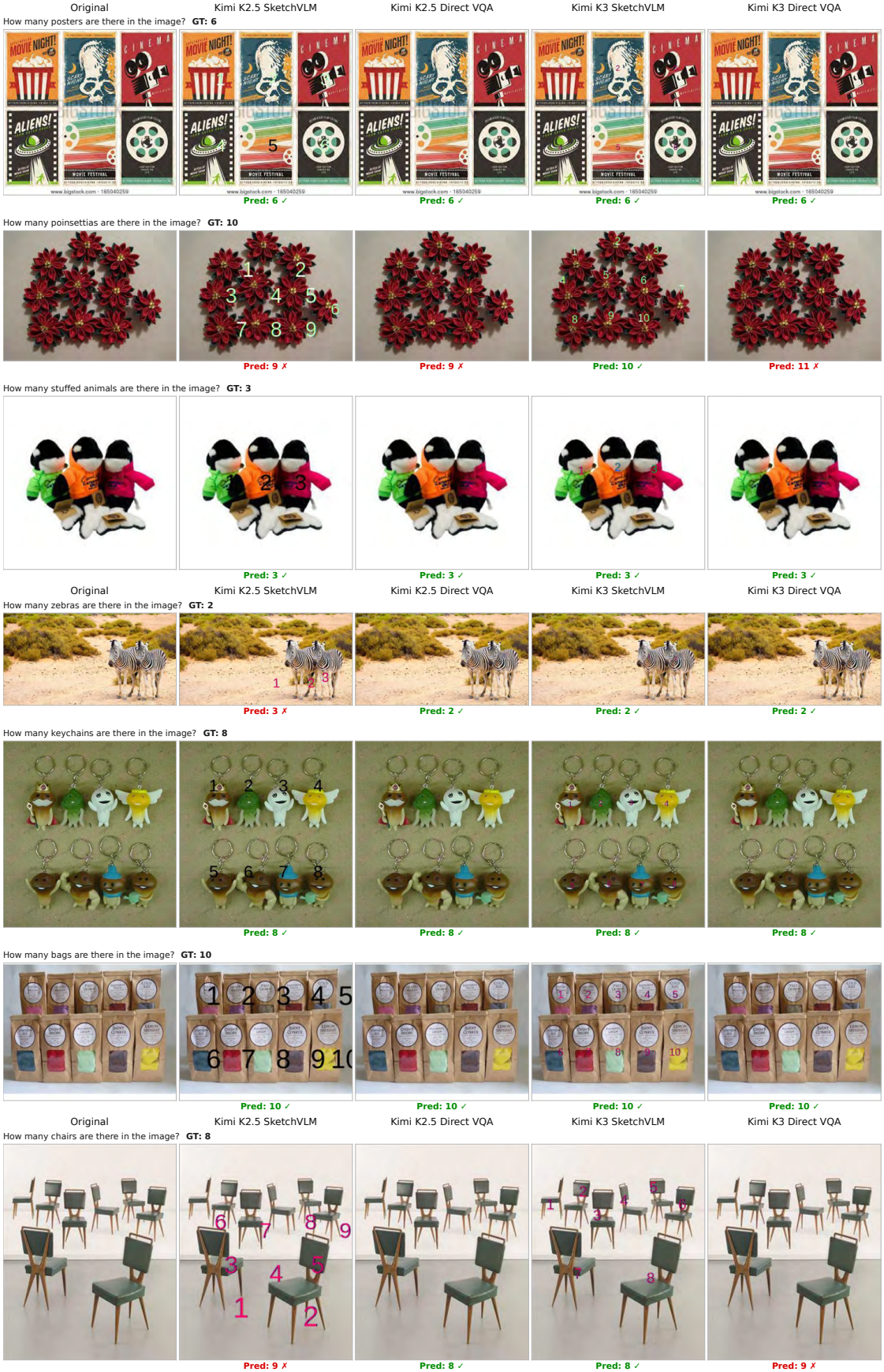


Figure D13: Qualitative comparison on the counting task between Kimi K2.5 and Kimi K3 using SketchVLM and Direct VQA.

D.5 Drawing Shape



Figure D14: Qualitative comparison on drawing shape tasks between Gemini-3-Pro-Preview, ✦ sketchVLM , and Nano Banana Pro.

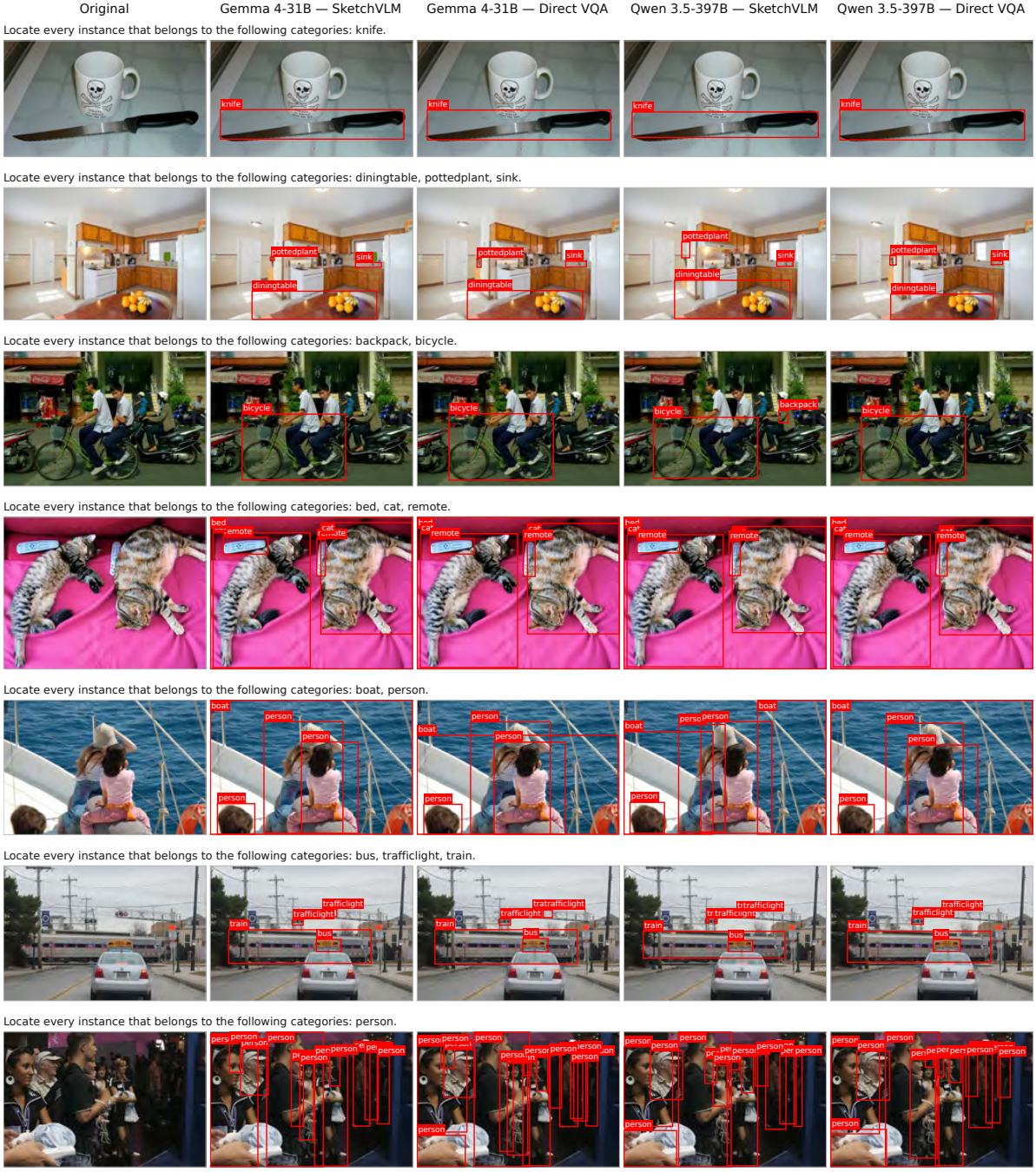


Figure D15: Qualitative comparison of SketchVLM and Direct VQA on the drawing shape task using Gemma 4 and Qwen3.5.

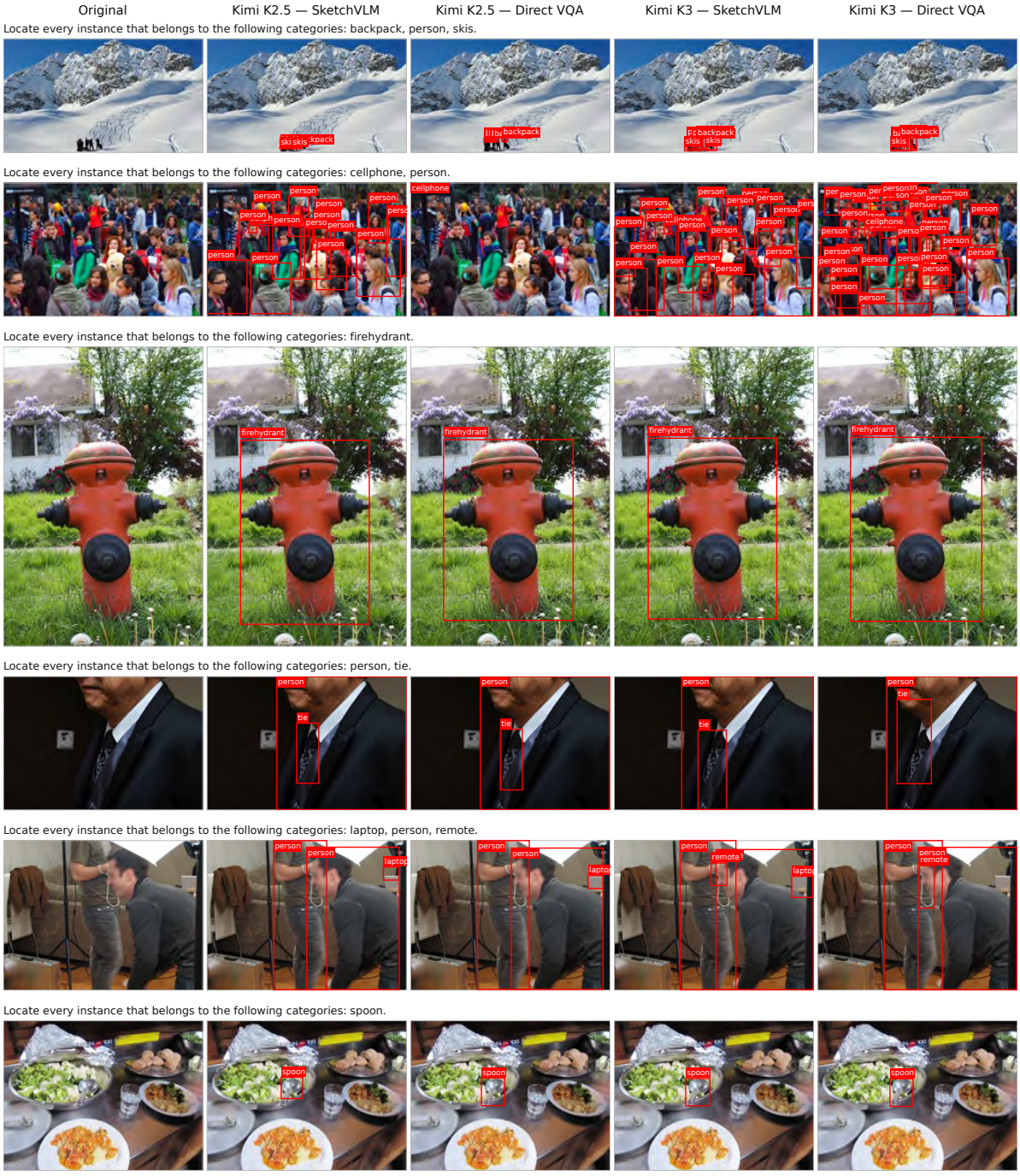


Figure D16: Qualitative comparison of SketchVLM and Direct VQA on the drawing shape task using Kimi K2.5 and Kimi K3.

D.6 Part Labeling

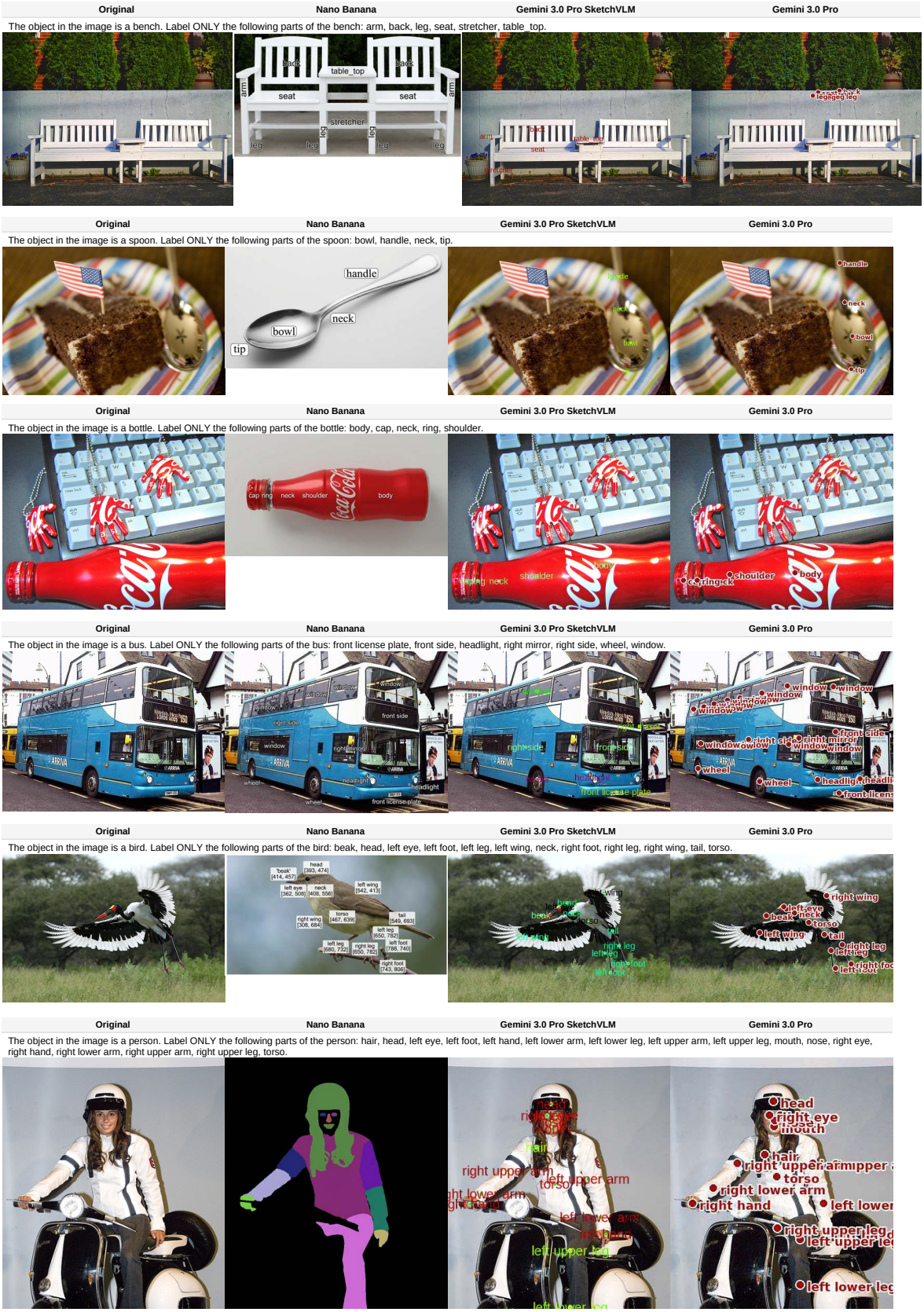


Figure D17: Qualitative comparison on part labeling tasks between Gemini-3-Pro-Preview, ✦ sketchVLM , and Nano Banana Pro.

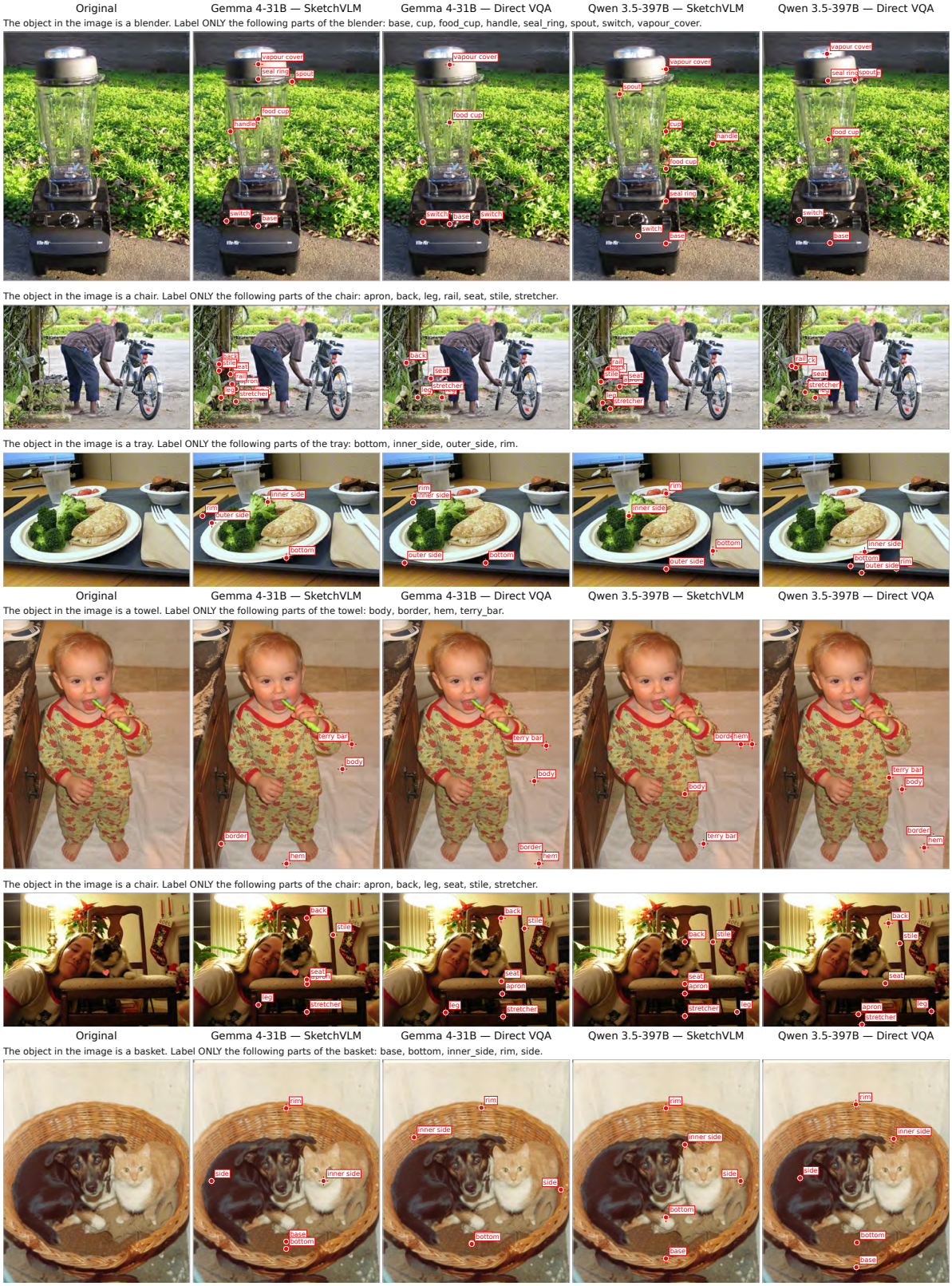


Figure D18: Qualitative comparison of SketchVLM and Direct VQA on the part labeling task using Gemma 4 and Qwen3.5-397B.

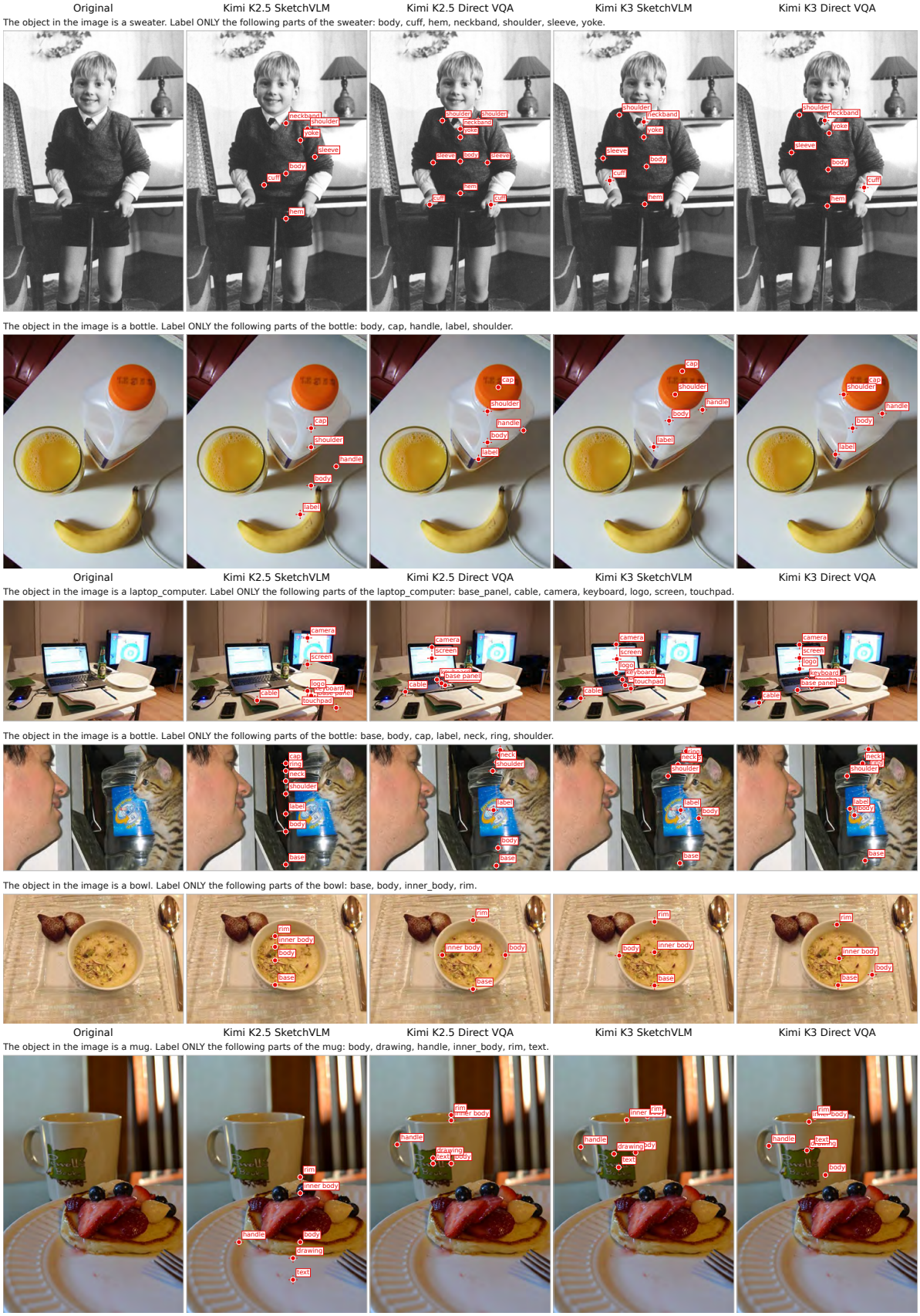


Figure D19: Qualitative comparison of SketchVLM and Direct VQA on the part labeling task using Kimi K2.5 and Kimi K3.

D.7 Error Analysis

As reported in Sec. 5.8, Gemini-3-Flash infers the SketchVLM text answer from the annotation alone in 95.5% of GPT-5 cases and 94.2% of Gemini-3-Pro-Preview cases. To better understand the remaining failures, we manually inspected 101 failed Maze Navigation examples and categorized them into three failure modes:

1. *Confusing sketch*: The sketch contains unnecessary strokes or exaggerated curves that make it difficult to interpret.
2. *Text contradiction*: The sketch directly contradicts the model's own text answer.
3. *Judge classification error*: The sketch is readily interpretable to a human, but the VLM judge makes a reasoning error and misclassifies it.

The failures of Gemini-3-Pro-Preview primarily consist of confusing sketches (72.4%), whereas those of GPT-5 primarily involve contradictions between the sketch and the model's own text answer (51.2%) (Tab. D1). Notably, 19.0% and 34.9% of the respective failures are attributable to judge classification errors rather than annotation errors. Thus, the reported alignment scores are conservative estimates of true annotation faithfulness.

We also randomly selected 100 VPCT examples and manually inspected the responses from both Gemini-3-Pro-Preview and GPT-5. The most common annotation error is crossing walls in the image. The second most common error is incorrect physical reasoning, such as predicting that a ball will change direction in midair (Tab. D2 and Figs. D20 and D21). Other examples of errors can be found throughout the qualitative samples Secs. D.2 to D.6 and Figs. D3, D4, D6 and D7.

Table D1: Distribution of annotation–text alignment failures on Maze Navigation. Confusing sketches account for most Gemini-3-Pro-Preview failures, whereas text contradictions account for most GPT-5 failures.

Model	Confusing sketch	Text contradiction	Judge classification error
Gemini-3-Pro-Preview	72.4%	8.6%	19.0%
GPT-5	14.0%	51.2%	34.9%

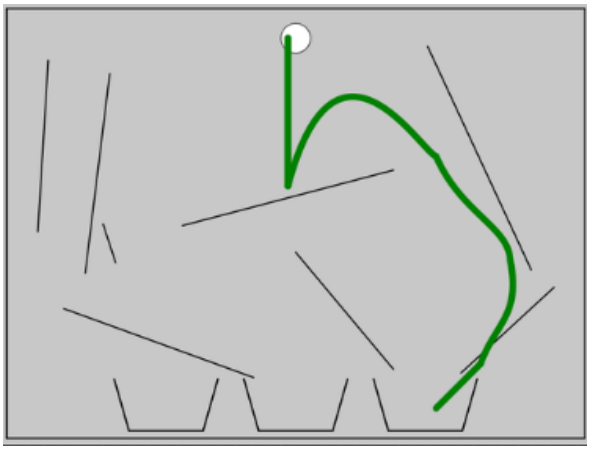


Figure D20: Example of sketching with bad physics understanding on VPCT. The predicted ball trajectory changes direction in midair.

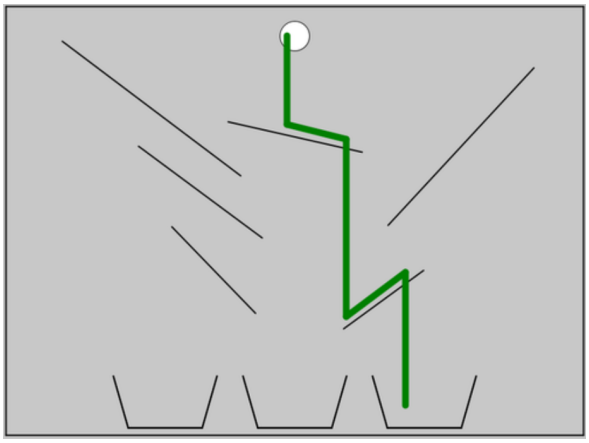


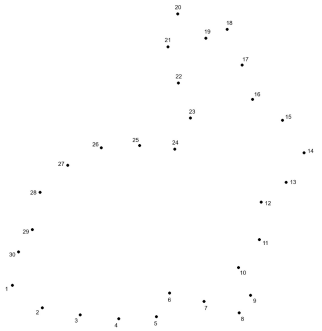
Figure D21: Example of sketching with both bad physics understanding and crossing walls on VPCT. The predicted ball trajectory is physically implausible and crosses a wall.

Table D2: Distribution of annotation errors on 100 randomly selected VPCT examples. Wall crossings are the most common annotation error for both models.

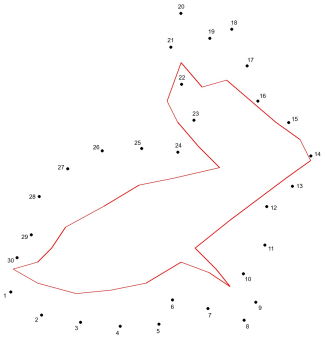
Model	No errors	Crosses wall	Incorrect physics	Both
Gemini-3-Pro-Preview	91.0%	7.0%	2.0%	0.0%
GPT-5	56.0%	28.0%	10.0%	6.0%

D.8 Connect-the-Dots: Grid versus No Grid

Source



GPT-5 (without Grid)



GPT-5 (with Grid)

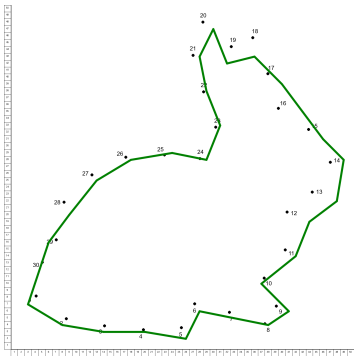
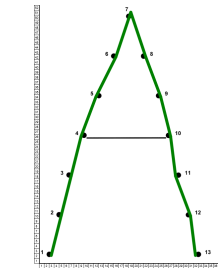
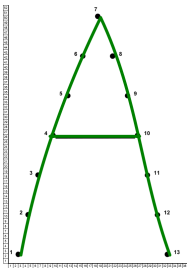


Figure D22: Appending a reference coordinate grid to the edge of the input image allows GPT-5 to be more precise, but does not help Gemini-3-Pro-Preview.

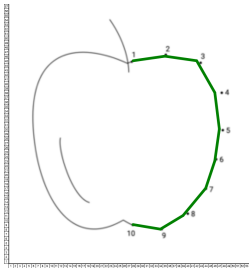
D.9 Connect-the-Dots: Bézier Curves versus Lines



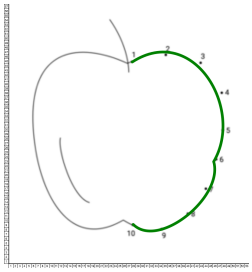
Straight Lines: 12 strokes



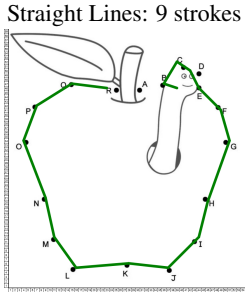
Bézier Curves: 2 strokes



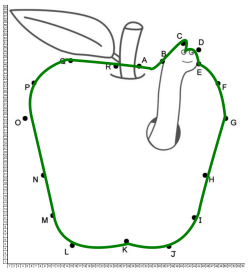
Straight Lines: 9 strokes



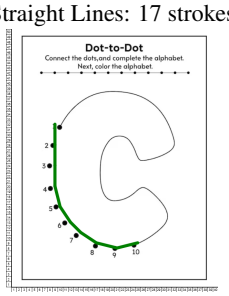
Bézier Curves: 1 stroke



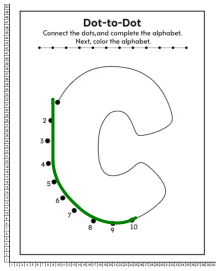
Straight Lines: 17 strokes



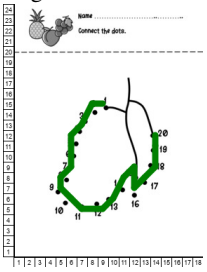
Bézier Curves: 9 strokes



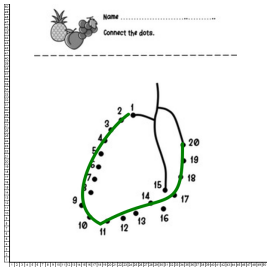
Straight Lines: 9 strokes



Bézier Curves: 3 strokes



Straight Lines: 19 strokes



Bézier Curves: 1 stroke

Figure D23: Using Bézier curves instead of straight lines allows the model to connect the dots in fewer strokes, and it can draw less jagged shapes, leading to a more aesthetically pleasing result.

D.10 Gemini-3-Pro-Preview Coordinate Systems

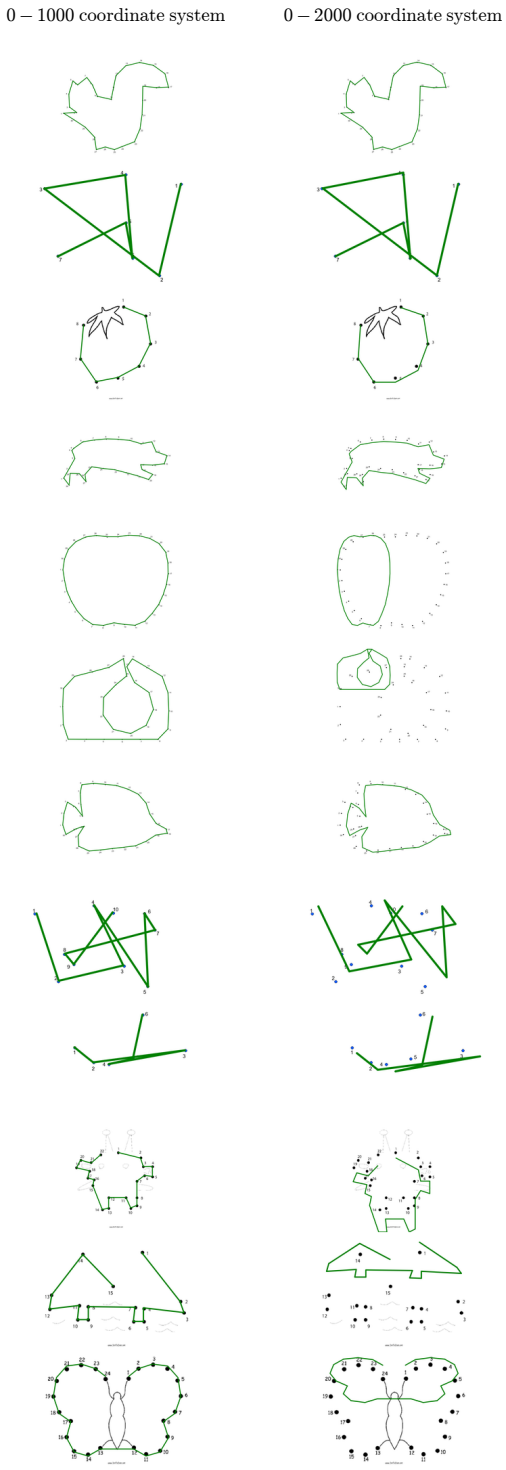


Figure D24: Comparing Gemini-3-Pro-Preview annotations under two coordinate systems. The 0–2000 system (vs. the 0–1000 system the model is typically used to) often preserves the overall shape but produces annotations that are compressed or shifted. In most cases the model adapts to the new coordinate system, but these failure cases lead to a noticeable decrease in performance. We hypothesize this change is what causes Gemini-3-Pro-Preview and Gemma-4-31B to also perform worse with the grid (which has a different coordinate system).

D.11 Multi-turn Ablation

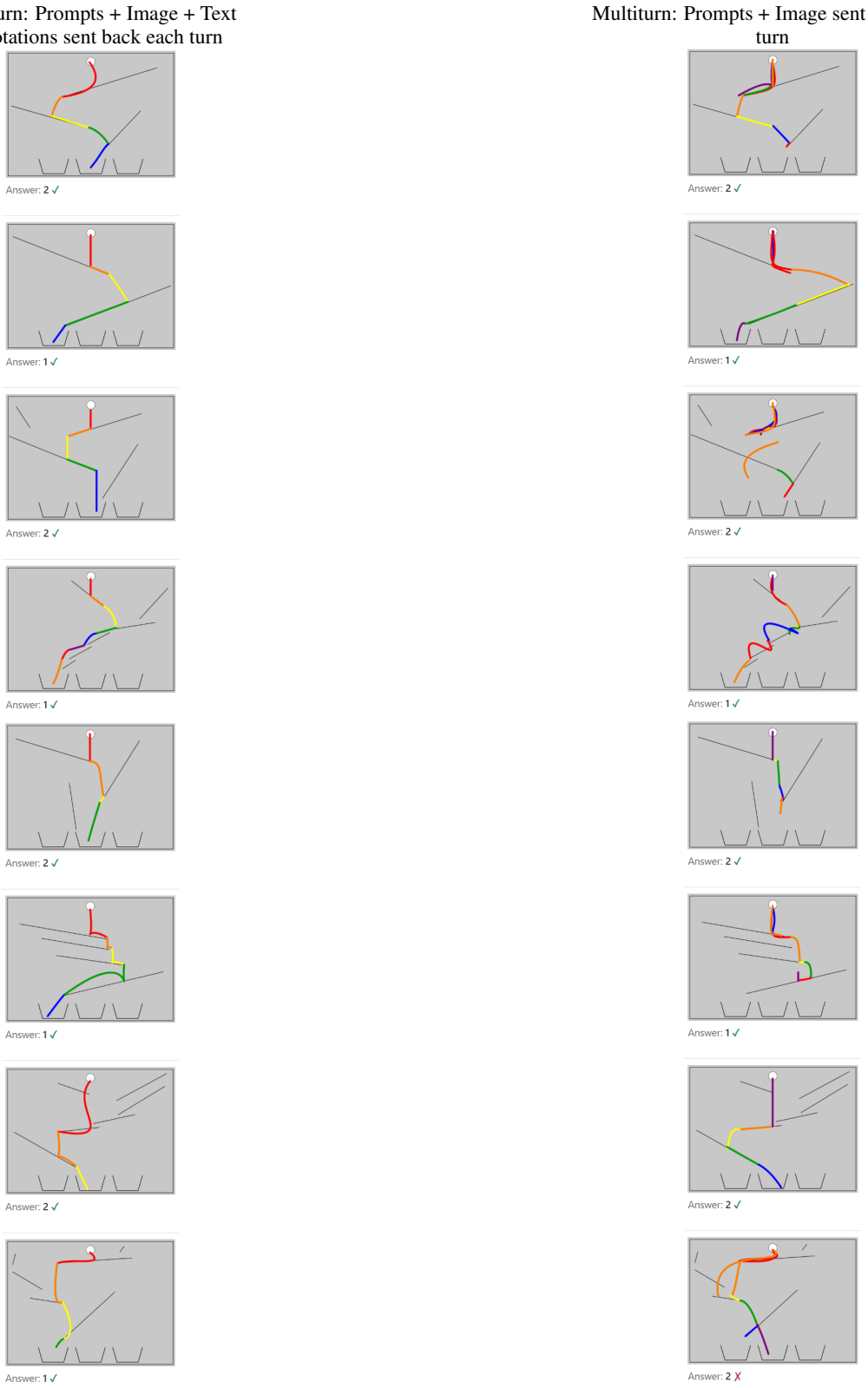


Figure D25: When testing Gemini-3-Pro-Preview in the multi-turn setting, not sending back the text representation of prior annotations degrades drawing quality: the model often attempts to redraw earlier strokes and fails to properly connect new strokes to the existing trajectory.

E VLM-Judge Details

We use a VLM judge (Gemini-3-Flash-Preview (DeepMind, 2025a)) to evaluate annotation quality and annotation–text alignment. To verify that this automated metric is meaningful, we measure how well VLM judge ratings agree with human annotators. The human annotators were recruited internally and were not asked to provide personal information. They were volunteers and not paid. Annotators were informed that their ratings would be used for research evaluation and reported anonymously.

Experiment Three independent human annotators each rate 50 annotations across the Ball Drop, VPCT, and Maze Navigation tasks from all 5 models (✦ Gemini-3-Pro-Preview, GPT-5, Nano Banana Pro, ViLaSR, and ThinkMorph), totaling 2,250 labeled images. Both human and VLM judges use the same 1–5 grading rubric (Sec. E.3). We report agreement using quadratically weighted Cohen's κ, which penalizes larger disagreements more heavily on ordinal scales (Latif et al., 2025), and Pearson correlation, following recent VLM-judge benchmarks (Li et al., 2026).

Results Human–human agreement is high, with $\kappa = 0.84 \pm 0.04$ and Pearson $r = 0.85 \pm 0.04$ (Tab. E1), indicating strong consistency between annotators. Detailed per-dataset agreement statistics are reported in Tab. E3.

Human–VLM agreement is moderate, with $\kappa = 0.51 \pm 0.02$ and Pearson $r = 0.52 \pm 0.01$, with task-level results shown in Tab. E2. The VLM judge's ratings are positively correlated with human judgments across all tasks and models, making it a useful proxy for large-scale evaluation. However, the gap between human–human and human–VLM agreement indicates that the VLM judge can miss subtle errors, particularly logical violations such as trajectories clipping through walls (Sec. E.2).

Finally, we test whether using Gemini-3-Flash-Preview as the VLM judge favors Gemini-3-Pro-Preview. We compare its scores against those of an independent judge, Kimi K2.6. Both judges assign Gemini-3-Pro-Preview lower scores relative to the human ratings than they do on average for the other four models, providing no evidence of bias toward Gemini-3-Pro-Preview (Tab. E5).

We therefore use the VLM judge for cost-effective large-scale comparison across models, while acknowledging that human annotation remains more reliable for fine-grained evaluation.

Table E1: Agreement is measured by quadratic κ and Pearson correlation (mean $\pm$ std). Human annotators are highly consistent, while the VLM judge shows moderate alignment with human judgment.

Comparison	Kappa (Quadratic)	Pearson
Human–Human	0.84 ± 0.04	0.85 ± 0.04
Human–VLM Judge	0.51 ± 0.02	0.52 ± 0.01

Table E2: Human–VLM agreement measured by quadratically weighted Cohen's κ and Pearson correlation. The VLM judge achieves consistent moderate agreement with all annotators, with higher reliability on valid trajectories and lower agreement when annotations violate maze constraints.

	Annotator 1		Annotator 2		Annotator 3	
Dataset	κ_{quad}	Pearson	κ_{quad}	Pearson	κ_{quad}	Pearson
ball-path	0.53	0.53	0.51	0.51	0.51	0.52
maze-invalid	0.40	0.41	0.44	0.46	0.42	0.44
maze-valid	0.53	0.58	0.60	0.64	0.63	0.65
vpct	0.42	0.49	0.41	0.44	0.46	0.50

Table E3: Human–human agreement across datasets. Annotators show consistently high agreement ($\kappa \approx 0.85$–0.92), indicating reliable human evaluation, with slightly lower agreement on *maze-invalid* cases.

	Annotator 1 vs Annotator 2		Annotator 1 vs Annotator 3		Annotator 2 vs Annotator 3	
Dataset	κ_{quad}	Pearson	κ_{quad}	Pearson	κ_{quad}	Pearson
ball-path	0.90	0.90	0.90	0.90	0.85	0.85
maze-invalid	0.80	0.82	0.50	0.52	0.53	0.53
maze-valid	0.91	0.92	0.91	0.92	0.92	0.93
vpct	0.89	0.90	0.88	0.89	0.86	0.87

Table E4: Human evaluation score (1–5, higher is better). We report mean $\pm$ standard deviation for each task and the overall mean across all tasks. Best mean in each column is bolded. SketchVLMs have higher mean annotation quality than other annotation models.

Model	Human Score (1–5)				
	VPCT	Ball Drop	Maze Navigation (Invalid)	Maze Navigation (Valid)	Mean
sketchVLM	3.18 ± 1.27	3.24 ± 1.08	$\mathbf{4.13 \pm 1.24}$	$\mathbf{4.45 \pm 1.22}$	3.70 ± 1.32
✦ sketchVLM	$\mathbf{4.56 \pm 0.85}$	$\mathbf{3.79 \pm 1.30}$	3.92 ± 1.27	4.36 ± 1.07	$\mathbf{4.14 \pm 1.18}$
Nano Banana Pro	2.44 ± 1.30	2.94 ± 1.56	2.77 ± 1.41	4.29 ± 1.36	3.08 ± 1.57
ViLaSR	1.26 ± 0.88	1.01 ± 0.12	3.02 ± 1.34	1.96 ± 1.05	1.74 ± 1.20
ThinkMorph	1.44 ± 1.01	1.10 ± 0.38	1.17 ± 0.45	1.25 ± 0.79	1.24 ± 0.72

Table E5: Comparison of human and VLM-judge scores. Here, Δ is the judge score minus the human score. Relative bias is Δ for Gemini-3-Pro-Preview minus the mean Δ across the other four models; negative values indicate that the judge rates Gemini-3-Pro-Preview less favorably relative to the other models.

Sketch model	**Human**	**Kimi K2.6 judge (Δ)**	**Gemini-3-Flash judge (Δ)**
✦ sketchVLM	**4.14**	3.93 (-0.21)	3.70 (-0.44)
sketchVLM	3.70	3.26 (-0.44)	2.26 (-1.44)
Nano Banana Pro	3.08	3.32 ($+0.24$)	2.60 (-0.48)
ViLaSR	1.74	1.81 ($+0.07$)	1.81 ($+0.07$)
ThinkMorph	1.24	1.87 ($+0.63$)	1.63 ($+0.39$)
Relative bias toward Gemini-3-Pro-Preview		-0.34	-0.08

E.1 Annotation Score Explanation Examples

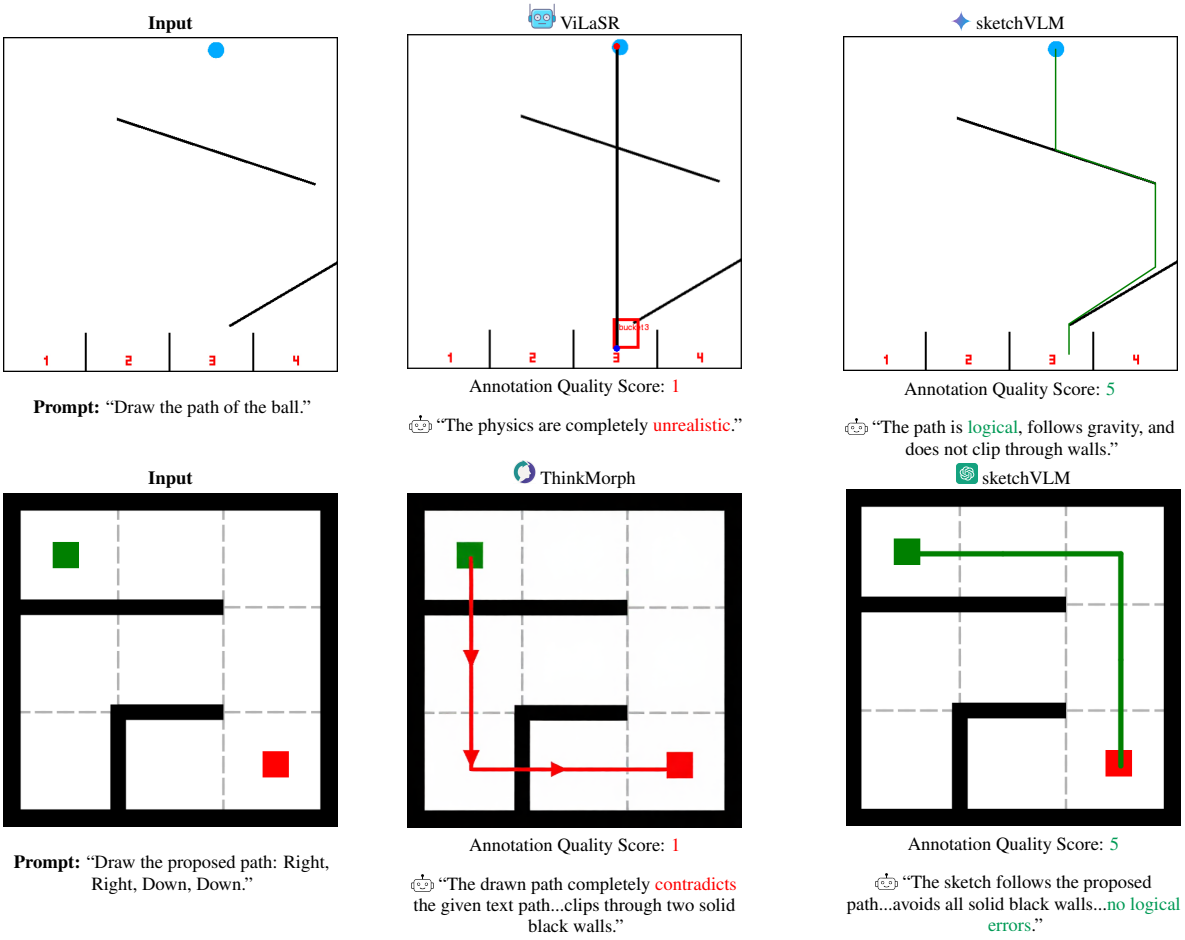


Figure E1: Low-quality annotations from ThinkMorph and ViLaSR may still lead to the correct final answer, but contain logical errors that are harder for users to verify than the high-quality annotations from SketchVLMs.

E.2 Qualitative Examples

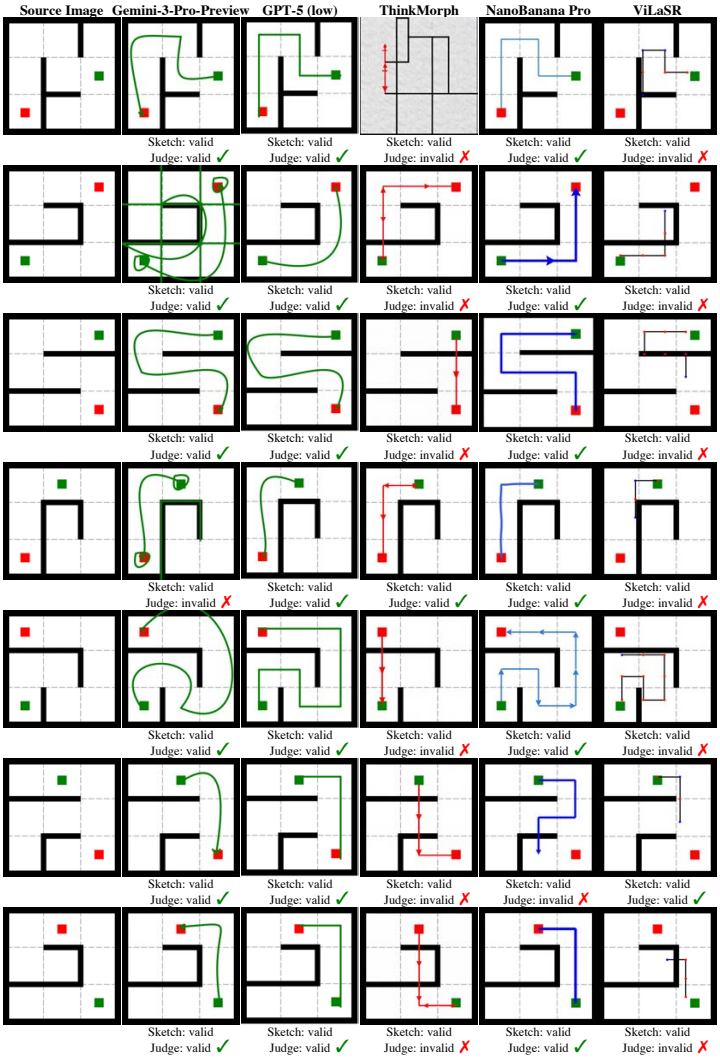


Figure E2: Comparison between the original sketching model's answer and the judge's answer for valid paths.

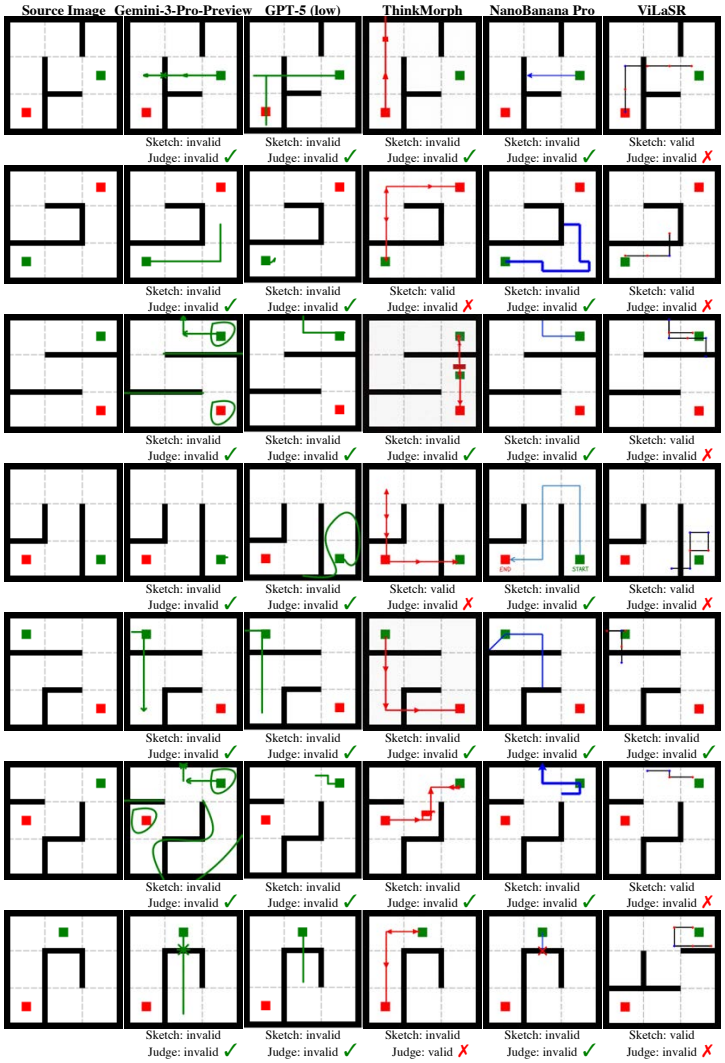


Figure E3: Comparison between the original sketching model's answer and the judge's answer for invalid paths.

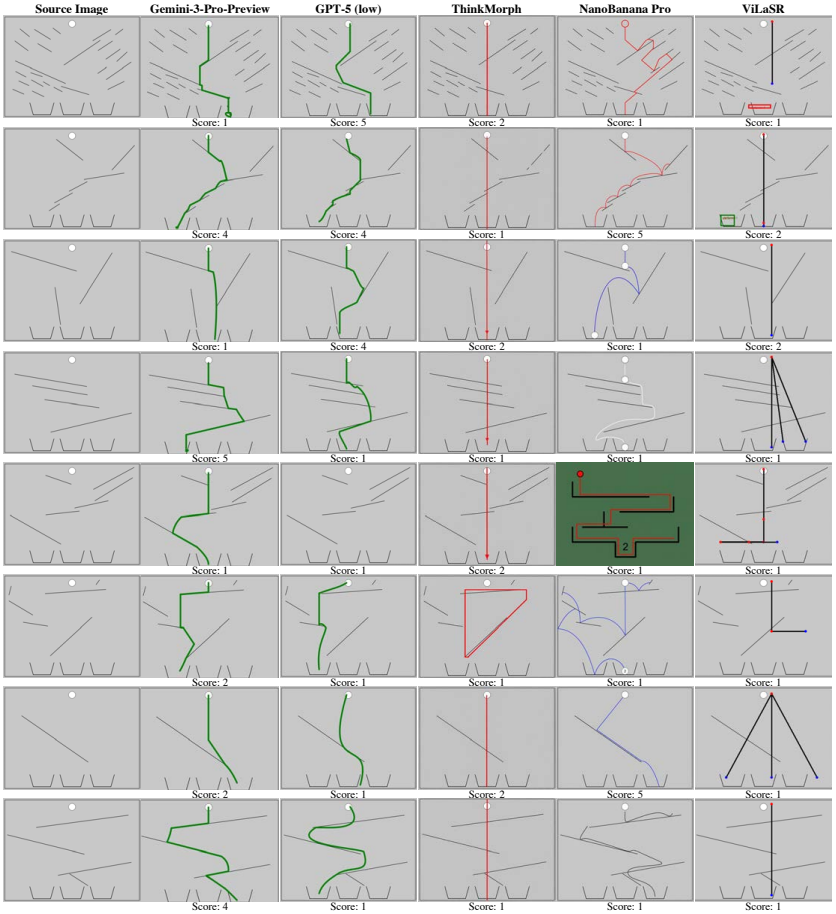


Figure E4: VLM-Judge quality scores for VPCT.

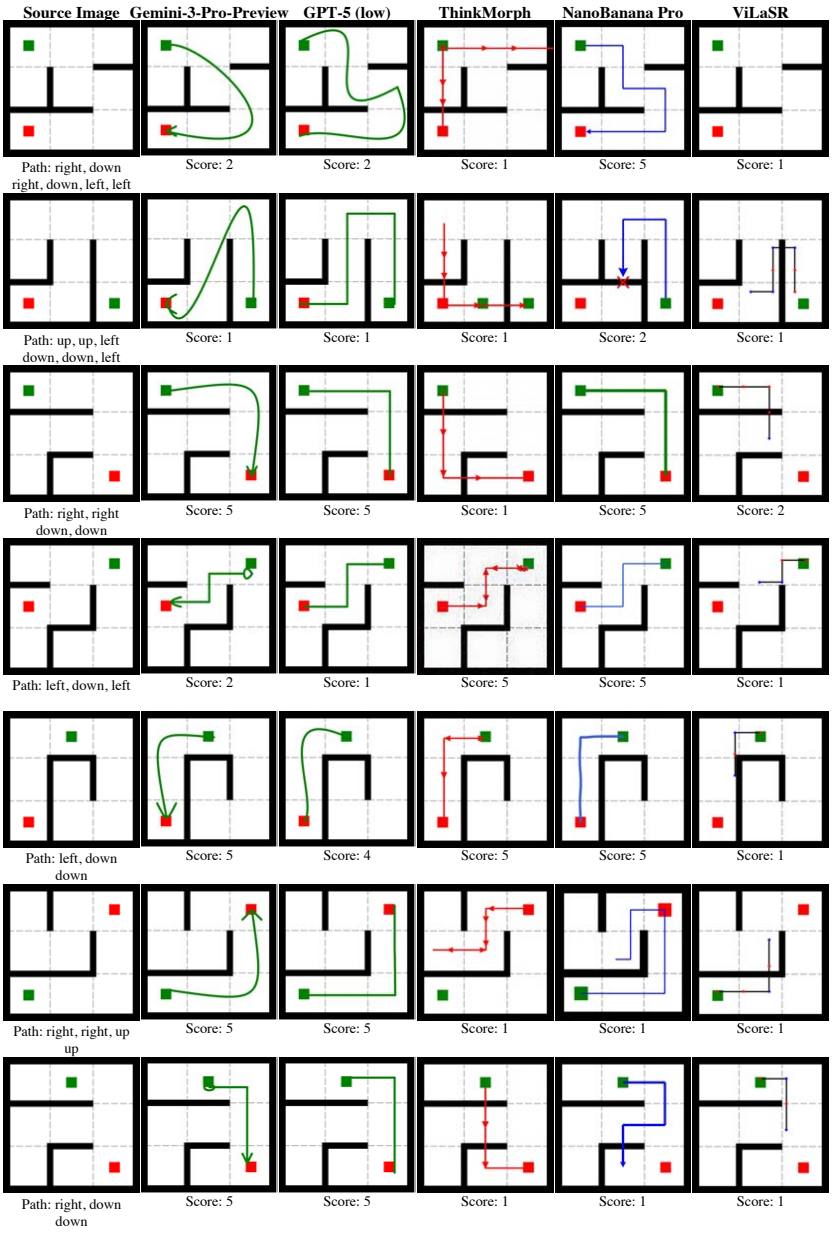


Figure E5: VLM-Judge quality scores for valid paths in the Maze Navigation task.

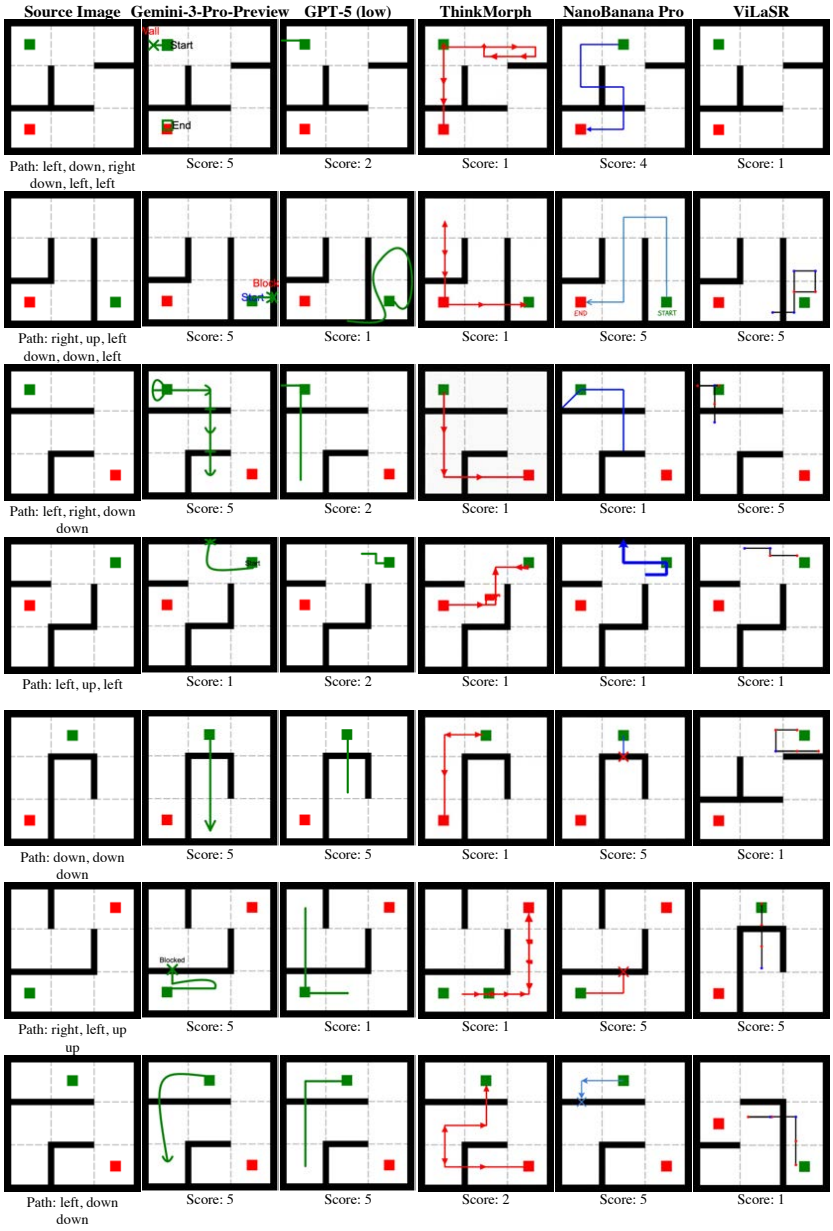


Figure E6: VLM-Judge quality scores for invalid paths in the Maze Navigation task.

E.3 Rubric Prompts

Rubric for VPCT and Ball Drop Quality

```
# Overall Instructions
You will be shown two images: the original image and an AI-annotated version. The annotation is supposed to show how
      the path the ball will take when it is dropped and where it will eventually end up. Your job is to grade the
      quality of the sketch using a rubric. You should grade only based on the following attributes that are given.

A minor issue is a violation that has little effect on the final answer of what bucket the ball will fall into. For
      example, if the ball path slightly clips through the walls, but the trajectory of the path still makes sense,
      then this is only a minor issue and not a major issue. A major issue is a violation that has a significant
      effect on the final answer of what bucket the ball will fall into. For example, if the ball path clips through
      the walls and the trajectory of the path is not logical, then this is a major issue. If there is no sketch
      provided, then the score should be a 1. The score should be based on the scale of 1 - 5

# Quality Checks

Below are more details:

## Things that are bad
1) The drawn path clips through any of the black walls.
2) The drawn path has very unrealistic physics. For example, the ball path momentarily disobeys gravity by moving
      upwards or moving in a direction that is not logical for gravity.
3) The drawn path substantially alters the original image (like adding or removing walls)
4) The drawn path contains multiple different paths instead of a single path.

## Things that are good
1) The drawn path is a single path that clearly shows where the final resting position of the ball will be.
2) The drawn path does not cut through any of the black walls.

## Things that are okay
1) If the path slightly clips through the walls, but the trajectory of the path still makes sense, then this is only
      a minor issue and not a major issue. We want to heavily penalize the ball going through walls and following a
      totally illogical path.
2) It's okay if the drawn path does not start exactly at the origin of the ball. As long as it is close, then this is
       not a minor or major issue.



# Scoring breakdown

1) The sketch has several critical flaws
2) The sketch has a critical flaw
3) The sketch contains some errors, overall direction of the path is valid and makes sense.
4) The sketch contains one minor logical error.
5) The sketch contains zero errors.

# Output Format
You should follow this output format EXACTLY with no other output:

{reasoning for logical consistency score}
Quality Score: {integer from 1 - 5}

# Example Output

<example_1>
The drawing contains multiple errors. The ball path barely clips through one of the platforms, but it is a minor
      issue as the path of the ball still makes sense. Additionally, the ball path defies gravity and begins to
      momentarily float straight upwards for no logical reason. The minor error combined with the critical error
      results in a logical score of 2/5.
Quality Score: 2
</example_1>

<example_2>
Overall, the path the ball takes is logical. It properly reflects off the first wall and then rolls to the right.
      Then, it falls down to the left and lands in bucket 2. The ball path slightly clips through one of the black
      edges, but this is a minor issue because the big picture trajectory of the ball is still logical. Given the
      overall path makes sense and there was just one minor issue, the quality score is 4/5.
Quality Score: 4
</example_2>
```

Figure E7: Rubric for VPCT and Ball Drop Quality

Rubric for Path Navigation Quality

```
# Overall Instructions
You will be shown two images: the original image and an AI-annotated version. Your job is to grade the quality of the
      sketch using a rubric. You should grade only based on the following attributes that are given.

# Sketch Instructions
The original sketch was drawn according to the following prompt:

"You are given an image of a maze where the green square marks the START cell and the red square marks the END cell
     of the maze. The walls of the maze are solid black lines. Dashed gray lines mark cell boundaries that can be
     crossed. You are given a proposed sequence of moves to reach the end of the maze starting from the green square
     and ending at the red square. Each move will move exactly one cell length in that direction. For example, "right
     " means move one cell in the maze to the right. A valid path must NOT cross any solid black walls and must end
     up in the red square cell. A valid path can also move through any of the dashed gray cell lines. The grid on the
      outside of the image is only there to help provide a reference for you. Moving one right means go one big cell
     in the maze right."

# Quality Checks

The main things that you should be looking for are:
1) Clipping through walls when it is not required to.
2) The drawn path does not go to the CENTER of each cell that it goes through.
3) The drawn path contradicts the given text path.

Below are more details:

## Things that are bad
1) The drawn path clips through any of the black walls when it is not required to. For example, even if the
     directions of the drawn path are correct, if the path touches or goes through a wall, then it is a bad sketch.
     That means that if the path goes through a wall even when it is not absolutely required to, then it is a bad
     sketch.
2) Each move in the drawn path should go to the **center** of the next cell in the path. If the drawn path is a
     curved path, then this does not apply. This is important! Look at each step in the path and make sure that the
     drawn path goes to the center of the next cell.
3) The sketch contains additional moves that are not in the path
4) The drawn sketch contradicts the given text path.
5) Even if the directions of the drawn path are correct, the end of the path does not end up touching the red square.
6) The drawn path does not start by touching the green square

## Things that are not an issue
1) If the proposed path is not valid, the drawn sketch shows exactly what the path should look like (even if it has
     to clip through walls or double back on itself).
2) If the proposed path is not valid, and the drawing ends as soon as there is an invalid move that is taken (such as
      requiring to go through a wall), then that is not an issue. It's okay for the drawing to not show all the steps
      of the path here since the sketch is emphasizing that the path is invalid.

## Score breakdown

1) The sketch makes absolutely no logical sense.
2) The sketch has some critical flaws that breaks the logic of the sketch.
3) The sketch contains multiple logical errors.
4) The sketch contains a minor error.
5) The sketch contains zero errors.

# Output Format
You should follow this output format EXACTLY with no other output:

{reasoning for quality score}
Quality Score: {integer from 1 - 5}

# Example Output
<example_1>
The drawing contains multiple errors. The drawn path goes up, up, left instead of up, up, right. This contradicts the
      given text path. Additionally, the end of the drawn path slightly clips through the solid black wall.
     Additionally, the draw path does NOT go to the center of each cell that it goes through. Instead, it only moves
     about 60 percent of the way across each cell and doesn't therefore doesn't follow the grid logic of moving from
     one center of each cell to the center of the next cell. The minor error combined with the multiple critical
     errors results in a score of 1/5.

Quality Score: 1
</example_1>

The original proposed path that the model should have followed is: {INSERT_ORIGINAL_PROMPT_PATH}
```

Figure E8: Rubric for Maze Navigation Quality

F Model Settings and Prompts

F.1 API Settings

Table F1: Model inference settings across providers. Temperature uses the provider default for all models. Reasoning uses the provider default unless otherwise specified.

Model	Model Identifier	Reasoning	API / Provider
GPT-5	`gpt-5`	Low, Medium, High	OpenAI Responses API
Gemini 2.5 Flash	`gemini-2.5-flash`	Default	Google AI Studio API
Gemini 2.5 Pro	`gemini-2.5-pro`	Default	Google AI Studio API
Gemini 3 Pro	`google/gemini-3-pro-preview`	Default	OpenRouter (Google AI Studio)
Gemini 3.1 Pro	`google/gemini-3.1-pro-preview-20260219`	Default	OpenRouter; latency/cost evaluation only
Gemini 3 Flash	`google/gemini-3-flash-preview`	Default	OpenRouter (Google AI Studio)
Nano Banana Pro	`google/gemini-3-pro-image-preview`	Default; image-only output	OpenRouter (Google AI Studio)
Gemma-4-31B	`google/gemma-4-31b-it`	Default	OpenRouter
Qwen3.5-397B-A17B	`qwen/qwen3.5-397b-a17b`	Default	OpenRouter
Kimi K2.5	`moonshotai/kimi-k2.5`	Default	OpenRouter
Kimi K3	`moonshotai/kimi-k3`	Low, High, Max	OpenRouter
Qwen 2.5 VL 7B	`qwen/qwen-2.5-vl-7b-instruct`	Default	OpenRouter (Hyperbolic)
ThinkMorph	`ThinkMorph/ThinkMorph-7B`	Default	Local; official code
ViLaSR	`inclusionAI/ViLaSR`	Default	Local; official code

F.2 SketchVLM System Prompt

System prompt (1/3)

```
# prompts.py

SYSTEM_PROMPT_BASE = """You are an expert artist specializing in drawing sketches that are visually
         appealing, expressive, and professional.
You will be provided with a blank grid. Your task is to specify where to place strokes on the grid to
         create a visually appealing sketch to complete the request.
The grid uses numbers (0 to {res_x}) along the bottom (x axis) and numbers (0 to {res_y}) along the left
         edge (y axis) to reference specific locations within the grid. The {origin_corner} is the origin
         . Each cell is uniquely identified by a combination of the corresponding x axis numbers and y
         axis number (\eg, the bottom-left cell is '{example_bottom_left}', the cell to its right is '{
         example_right_of_bottom_left}').
You can draw on this grid by specifying where to draw strokes. You can draw multiple strokes to depict the
          whole object, where different strokes compose different parts of the object.
To draw a stroke on the grid, you need to specify the following:
Starting Point: Specify the starting point by giving the grid location (\eg, 'x0y0' for column 0, row 0).
Ending Point: Specify the ending point in the same way (\eg, 'x{res_x}y{res_y}' for column {res_x}, row {
         res_y}).
Intermediate Points: Specify at least two intermediate points that the stroke should pass through. List
         these in the order the stroke should follow, using the same grid location format (\eg, 'x6y5', '
         x13y10' for points at column 6 row 5 and column 13 row 10).
Parameter Values (t): For each point (including the start and end points), specify a t value between 0 and
          1 that defines the position along the stroke's path. t=0 for the starting point. t=1 for the
         ending point.
Intermediate points should have t values between 0 and 1 (\eg, "0.3 for x6y5, 0.7 for x13y10").

Examples:
To draw a smooth curve that starts at x8y6, passes through x6y7 and x6y10, ending at x8y11:
Points = ['x8y6', 'x6y7', 'x6y10', 'x8y11']
t_values = [0.00,0.30,0.80,1.00]
To close this curve into an ellipse shape, you can add another curve:
Points = ['x8y11', 'x11y10', 'x11y7', 'x8y6']
t_values = [0.00,0.30,0.70,1.00]
To draw a large circle that starts at x25y44 and ends at x25y44, passing through the cells x32y41, x35y35,
          x31y29, x25y27, x19y29, x15y35, x18y41:
Points = ['x25y44', 'x32y41', 'x35y35', 'x31y29', 'x25y27', 'x19y29', 'x15y35', 'x18y41', 'x25y44']
t_values = [0.00, 0.125, 0.25, 0.375, 0.50, 0.625, 0.75, 0.875, 1.00]
To draw non-smooth shapes (with corners) like triangles or rectangles, you need to specify the corner
         points twice with adjacent corresponding t values.
For example, to draw an upside-down "V" shape that starts at x13y27, ends at x24y27, with a pick (corner)
         at x18y37:
Points = ['x13y27', 'x18y37','x18y37', 'x24y27']
t_values = [0.00,0.55,0.5,1.00]
To draw a triangle with corners at x10y29, x15y33, and x9y35, start with drawing a "V" shape that starts
         at x10y29, ends at x9y35, with a pick (corner) at x15y33:
Points = ['x10y29', 'x15y33', 'x15y33', 'x9y35']
t_values = [0.00,0.55,0.5,1.00]
and then close it with a straight line from x13y27 to x24y27 to form a triangle:
Points = ['x13y27', 'x24y27']
t_values = [0.00,1.00]
Note that for a triangle, the start and end points should be different from each other.
To draw a rectangle with four corners at x13y27, x24y27, x24y11, x13y11:
Points = ['x13y27', 'x24y27', 'x24y27', 'x24y11', 'x24y11', 'x13y11', 'x13y11', 'x13y27']
t_values = [0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00]
To draw a small square with four corners at x26y25, x29y25, x29y21, x26y21:
Points = ['x26y25', 'x29y25', 'x29y25', 'x29y21', 'x29y21', 'x26y21', 'x26y21', 'x26y25']
t_values = [0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00]
To draw a single dot at x15y31 use:
Points = ['x15y31']
t_values = [0.00]
To draw a straight linear line that starts at x18y31 and ends at x35y14 use:
Points = ['x18y31', 'x35y14']
t_values = [0.00, 1.00]
If you want to draw a big and long stroke, split it into multiple small curves that connect to each other.
"""
```

Figure F1: System Prompt (1/3). Used for all SketchVLM models in single-turn and multi-turn.

System prompt (2/3)

```
# Sketch Methods

Below are the different sketching methods you can use for your task.

## FREEHAND SKETCH
- Emit one or more stroke blocks with points on the grid, no <text>.
- Use multiple strokes to compose shapes; curves/lines are both fine.
- the <id> tag should describe the part being drawn.

<s1>
  <points>'x12y20','x13y20','x14y21','x15y22'</points>
  <t_values>0.00,0.33,0.66,1.00</t_values>
  <id>part_1</id>
</s1>
<s2>
  <points>'x20y18','x20y14','x24y14','x24y18','x20y18'</points>
  <t_values>0.00,0.25,0.50,0.75,1.00</t_values>
  <id>part_2</id>
</s2>

## STRAIGHT LINE
<sN>
  <points>'x10y19','x40y19'</points>
  <t_values>0.00,1.00</t_values>
  <id>line_1</id>
</sN>

## Arrow (draw the shaft, and the arrowhead as two separate parts)

<s1>
  <points>'x12y32','x6y32'</points>
  <t_values>0.00,1.00</t_values>
  <id>arrow_shaft</id>
</s1>
<s2>
  <points>'x7y33','x6y32'</points>
  <t_values>0.00,1.00</t_values>
  <id>arrowhead_top</id>
</s2>
<s3>
  <points>'x7y31','x6y32'</points>
  <t_values>0.00,1.00</t_values>
  <id>arrowhead_bottom</id>
</s3>

## BOX / RECTANGLE (list the 4 corners in order)
<sN>
  <points>'x12y12','x20y12','x20y18','x12y18','x12y12'</points>
  <t_values>0.00,0.25,0.50,0.75,1.00</t_values>
  <id>box_1</id>
</sN>


## COUNTING (place numerals near each instance; one stroke per number; change text size based on object size and image resolution, so can be text size="1.0" or "2.0" up to "32.0" etc)
<sN>
  <points>'x08y22'</points>
  <t_values>0.00</t_values>
  <text size="4.0" color="black">'1'</text>
  <id>count_1</id>
</sN>

## LABELING (anchor a text label to the cell closest to center of the object/part; change text size based on object/part size and image resolution, so can be text size="1.0" or "2.0" up to "32.0" etc)
<sN>
  <points>'x26y17'</points>
  <t_values>0.00</t_values>
  <text size="3.2" color="black">'handlebar'</text>
  <id>label_handlebar</id>
</sN>

# Rules
- Output only <answer>...</answer> with a single <strokes>...</strokes> section.
- For counting/labeling tasks, prefer <text> with short values ('1','2',... or 'wheel','seat',...).
- Use <points> with exactly one anchor cell for each text label/number (one per item/part).
- Do not mix patterns: if the user asks to label, do not draw boxes; if the user asks to count, do not label names.
- Keep each stroke in its own <sN>...</sN> block; increment N in order without gaps.
- If the question requires an answer (\eg, "How many?"), include it at the end of your response, after the </strokes> tag, in a new <final_answer> tag.
"""
```

Figure F2: System Prompt (2/3), continued from Fig. F1.

System prompt (3/3)

```
def build_system_prompt(res_x: int, res_y: int, origin: str = "bottom_left") -> str:
    if origin == "top_left":
        origin_corner = "top left"
        example_bottom_left = f"x0y{res_y}"
        example_right_of_bottom_left = f"x1y{res_y}"
    else:
        origin_corner = "bottom left"
        example_bottom_left = "x0y0"
        example_right_of_bottom_left = "x1y0"

    return SYSTEM_PROMPT_BASE.format(
        res_x=res_x,
        res_y=res_y,
        origin_corner=origin_corner,
        example_bottom_left=example_bottom_left,
        example_right_of_bottom_left=example_right_of_bottom_left,
    )

COUNTING_PROMPT = """
Task:
- COUNT all the {object} by placing numbered SVG text strokes on them (no curves).

Output example could be:
<answer>
<concept>Numbering each {object}</concept>
<strokes>
<s1>
    <text size="1.6" color="#ff0066">'1'</text>
    <points>'xAyB'</points>
    <t_values>0.00</t_values>
    <id>marker_{object}1</id>
</s1>
<!-- s2, s3, ... one per object -->
</strokes>

Rules:
- Use ONLY text strokes (no curves).
- Exactly one point per stroke ('xAyB') at the object's center-ish cell.
- You MAY style numbers: <text size="1.8" color="#0057ff"> or <style><font_size>...</font_size><color>...</color></style>.
- size is cells (multiplier) unless you suffix 'px'
- choose bigger text size for larger objects, smaller for tiny objects. use bigger size for higher resolution images, smaller for lower resolution.
- choose readable colors that will contrast well with the object that you are numbering and the background.
- If 0 objects, still return the full wrapper with an empty <strokes> block.
- Do not write anything outside <answer>...</strokes>.
"""

GENERIC_LABEL_PROMPT = """
Task:
- The object in the image is a {concept}.
- Label ONLY the following parts of the {concept}: {labels_hint}.
- Do not invent or add any new part names beyond this list.

- Use SVG text strokes (no curves) to place each label.

Output EXACTLY this XML shape:
<answer>
<concept>Labeling: {concept}</concept>
<strokes>
  <!-- one <sN> per label -->
  <s1>
    <text size="1.6" color="#ff0066">'head'</text>   <!-- size: cells or 'px' -->
    <points>'xAyB'</points>
    <t_values>0.00</t_values>
    <id>label_head</id>
  </s1>
</strokes>

Rules:
- Use ONLY text strokes (no curves).
- Anchor each label at the center of the corresponding part ('xAyB').
- You MAY style the text labels: <text size="1.8" color="#0057ff"> or <style><font_size>...</font_size><color>...</color></style>
- Use one <sN> per label name in the list above.
- Choose bigger text size for larger objects, smaller for tiny objects. use bigger size for higher resolution images, smaller for lower resolution.
- Choose readable colors that will contrast well with the object/part that you are labeling and the background.
- Do not write anything outside <answer>...</strokes>.
"""
```

Figure F3: System Prompt (3/3), continued from Fig. F2. Parameters are res_x, res_y, and origin. res_x controls how many x coordinates there are. res_y controls how many y coordinates there are. origin controls whether the origin is at the top left or bottom left.

One-stroke guard (multiturn)

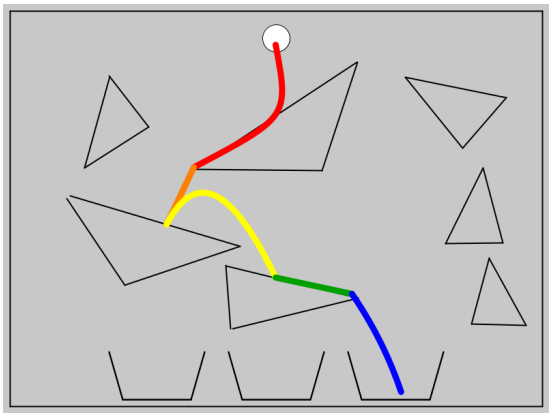
```
[Mode: stepwise]
You are in stepwise mode.
      On this turn you
      output EXACTLY ONE
      stroke block:
<answer>
  <strokes>
    <sN>...</sN>
  </strokes>
</answer>
Do NOT output any other
      <sM> blocks, no
      <final_answer>, no
      explanations.
If the drawing is already
      complete and no
      further stroke is
      needed, output an
      empty <answer> with
      NO <strokes> block.
Stop immediately after
      </answer>.
```

Final-answer guard (multiturn)

```
[Mode: stepwise]
All strokes have already
      been provided. On
      this turn output ONLY:
<final_answer> ...
      </final_answer>
Do not output the previous
      strokes again. Stop
      immediately after
      </final_answer>.
```

Figure F4: During multi-turn, when the model must draw a stroke, we append the "One-stroke guard" to clarify to the model that it should only draw one stroke with no final answer. After the model has decided that no more strokes are needed (it does this by outputting no strokes on a turn), we run one more turn where we prompt the model with "Final-answer guard."

F.3 SketchVLM Output Example



Stroke annotations (XML)

```
<s1>
  <points>'x500y100','x500y270','x500y270','x350y400'</points>
  <t_values>0.00,0.50,0.50,1.00</t_values>
  <id>path_1</id>
</s1>
<s2>
  <points>'x350y400','x325y470','x300y540'</points>
  <t_values>0.00,0.50,1.00</t_values>
  <id>drop_1</id>
</s2>
<s3>
  <points>'x300y540','x390y470','x500y670'</points>
  <t_values>0.00,0.50,1.00</t_values>
  <id>path_2</id>
</s3>
<s4>
  <points>'x500y670','x570y690','x640y710'</points>
  <t_values>0.00,0.50,1.00</t_values>
  <id>path_bounce</id>
</s4>
<s5>
  <points>'x640y710','x700y850','x730y950'</points>
  <t_values>0.00,0.60,1.00</t_values>
  <id>drop_2</id>
</s5>
<final_answer>3</final_answer>
```

Figure F5: Example sketch output and annotations for prompting ✦ sketchVLM on VPCT. Each stroke is colored differently for viewing purposes.

G Other Baselines

G.1 Baselines

Image-Editing Models. To benchmark the image-editing model Nano Banana Pro we first prompt it to generate a sketch for the input image. We then provide the resulting edited image to Gemini-3-Pro-Preview to produce the final task answer. For Connect-the-Dots, we additionally report a manual evaluation of Nano Banana Pro, since its outputs do not always map cleanly to our structured sketch representation.

Fine-tuned Sketching Models. For fine-tuned sketching models such as ViLaSR and ThinkMorph, we use the same task prompts as for SketchVLM. Because these models are trained to always output a sketch, they do not have a meaningful "no-sketch" baseline mode; therefore, we omit baseline VQA accuracy for these models in the main results and report only their sketch-conditioned performance.

H AI Use

We utilized AI assistants for grammar editing, LaTeX formatting, and code debugging. All code, experiments, scientific claims, analyses, and final writing decisions were reviewed and verified by the authors.