SPERRY: MULTIMODAL REASONING WITH UNIFIED MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Current test-time inference in multimodal models largely relies on textual thinking traces, which are insufficient for reasoning processes that are hard to verbalize, such as spatial reasoning and visual geometry. Existing alternatives use external tools or latent-space reasoning to obtain visual thoughts, but they either lack flexible pixel-level generation or fail to expose interpretable intermediate reasoning. Unified multimodal models that generate both text and images offer a promising foundation, yet their post-training for rigorous multimodal reasoning remains underexplored due to the lack of high-quality interleaved text-visual reasoning data and effective training recipes. We introduce Sperry-43K, a dataset of interleaved text-visual reasoning traces across diverse visual reasoning tasks, together with SperryBench, a benchmark for evaluating models' visual reasoning performance. We further propose a complete post-training pipeline combining two-stage supervised fine-tuning with reinforcement learning. To reduce training-inference mismatch during RL, we introduce Generative Self-Prompted Policy Optimization (GSPPO). Our 14B model improves the base unified multimodal model from 19.1% to 51.9% on SperryBench, outperforming much larger public models such as Qwen3.5-397B and Gemma-4-31B, as well as a latent-reasoning baseline trained on the same data. Code and data will be released.

1 INTRODUCTION

Recent reasoning models have achieved significant breakthroughs in tasks like mathematical problem-solving (Guo et al., 2025a; OpenAI, 2024; Hendrycks et al., 2021; Cobbe et al., 2021; Shao et al., 2024b; Gemini Team, 2025; Qwen Team, 2025) and reasoning-heavy coding (Chen et al., 2021; Jain et al., 2024; Jimenez et al., 2024; OpenAI, 2026a; Anthropic, 2026; Kimi Team, 2026) through test-time scaling (Snell et al., 2024). However, their reliance on purely textual reasoning processes (Wei et al., 2023) presents a fundamental limitation for visual reasoning domains (e.g., spatial reasoning and visual geometry (Chen et al., 2026; Li et al., 2025a; Xu et al., 2025a)). Unlike text-based reasoning, resolving these tasks often requires tracking intermediate states via updated visual states rather than sentences. Therefore, a multimodal reasoning system must transcend text-only states to accommodate visual intermediate steps.

Two paradigms have emerged to bring multimodal information into the reasoning process. Agentic vision methods (OpenAI, 2025; Kimi Team, 2026; Zheng et al., 2025b; Zhang et al., 2025e) use external tools such as cropping, zooming, highlighting, and code execution to extract visual information from the source image. Latent visual reasoning methods (Li et al., 2025b; Wang et al., 2026b) train a VLM to generate tokens that carry visual information to assist the reasoning process. Tool-based methods can draw new visual states (Menon et al., 2024), but rely on external operations; latent methods encode visual information without explicitly rendering it. We instead train a native image generator to produce the intermediate states used for reasoning.

To overcome the limitations of non-generative paradigms, a multimodal reasoning system must inherently possess the ability to model and predict visual state transitions. Traditional VLMs are structurally misaligned for this purpose as they are "read-only" visual processors. In response to this disconnect, unified multimodal models (UMMs) have recently emerged. By jointly pretraining on visual understanding and generation within a backbone, UMMs can seamlessly process and emit interleaved text and image tokens. Recent work trains UMMs for visual reasoning (Zhang et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

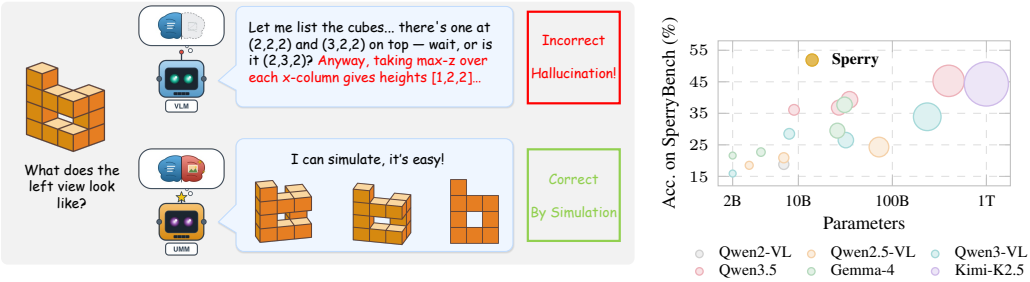

Figure 1: **Sperry at a glance. (a)** UMMs can reason by simulation and imagination. **(b)** With only 14B parameters, Sperry outperforms much larger open-source models on SperryBench.

2025f; Li et al., 2026; Hu et al., 2026). We study a complete post-training recipe that combines simulator-grounded traces, separate expert training, and programmatic image rewards.

As we aim to tackle visual problems whose intermediate states are difficult to express in text alone, and existing interleaved datasets such as Zebra-CoT (Li et al., 2025a) cover only part of this setting, we first curate Sperry-43K. Its training split contains 42,682 interleaved text-and-image reasoning traces across six tasks chosen to cover two capabilities, mental simulation/imagination and 3D/scene-level spatial intelligence, that are difficult to express in text alone. Alongside training data, we release SperryBench, a 2,249-sample benchmark for evaluating models' visual reasoning performance. Its samples are excluded from every training stage.

In addition, we propose Sperry, a post-training framework that adapts a UMM into a multimodal reasoner for visual reasoning. We first apply SFT on interleaved reasoning traces, where the understanding branch learns to produce the next text segment and decide when to generate an intermediate image, while the generation expert learns to render the corresponding visual state from the multimodal prefix. A straightforward joint SFT schedule already improves the model, but it does not fully fit the generated reasoning images: text behavior stabilizes quickly, whereas visual generation requires more updates to learn continuous state transitions. We therefore propose a two-stage SFT, training the understanding expert first and then fitting the generation expert with the understanding expert fixed.

SFT establishes the overall reasoning format, but errors remain concentrated in intermediate image generation (e.g., geometric distortions during structural rotation). Thus, we further introduce RL to enhance the accuracy of intermediate visual state generation, thereby improving end-to-end accuracy. While a naive RL algorithm yielded substantial performance gains for the model, a remaining source of error is a training-inference mismatch in the generator's prompts: during SFT the generator sees dataset prompts, whereas at inference its prompts come from the model's own text policy. Therefore, we propose Generative Self-Prompted Policy Optimization (GSPPO), which trains the model on self-generated prompts and further improves the model's performance during the RL phase.

Our experiments show that using UMMs for multimodal reasoning yields substantial performance gains. On SperryBench, Sperry improves Bagel from 19.1% to 51.9%. The gain runs through the generated images: replacing them at inference, or training without them, costs 10–18 points (Sec. 5.3). Moreover, with only a 14B model, Sperry outperforms Qwen3.5-397B (Qwen Team, 2026), Kimi-K2.5-1T (Kimi Team, 2026), and Gemma-4-31B (Farabet & Lacombe, 2026), all evaluated zero-shot. Sperry also surpasses Monet (Wang et al., 2026b), a latent visual reasoning baseline retrained on Sperry-43K, by 16.1 points (7.7 points after SFT alone), suggesting that explicitly generated images carry signal that latent visual tokens do not capture; Sec. 5.2 discusses the confounds of this comparison. Our contribution can be summarized as follows:

- We curate Sperry-43K, a dataset of interleaved text-and-image reasoning traces with simulator-grounded ground truth, along with SperryBench, a benchmark for evaluating models' visual reasoning performance.
- We propose Sperry, a post-training framework that turns UMMs into multimodal reasoners via a branch-aware two-stage SFT and a generation-branch RL stage with GSPPO.

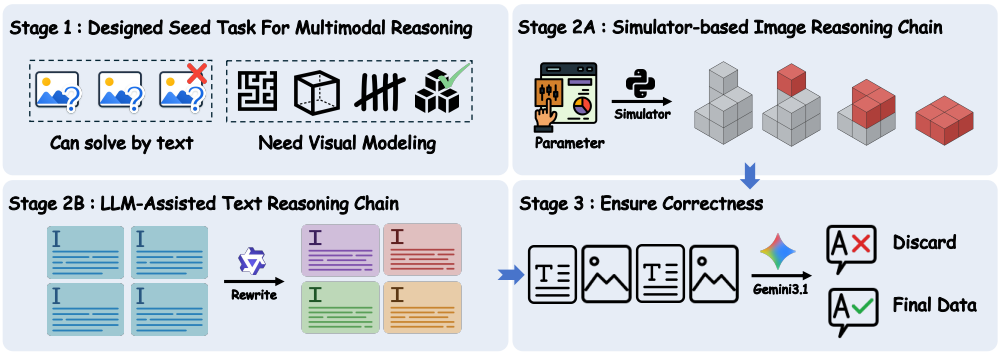

Figure 2: Overview of our data collection pipeline.

Table 1: Per-subset statistics of Sperry-43K and SperryBench.

Subset	#Train	#Bench	Total
Maze	4,548	240	4,788
BlockCount	5,611	296	5,907
CubeView	8,379	441	8,820
LineTrack	11,394	600	11,994
Counting	3,315	175	3,490
Jigsaw	9,435	497	9,932
Total	42,682	2,249	44,931

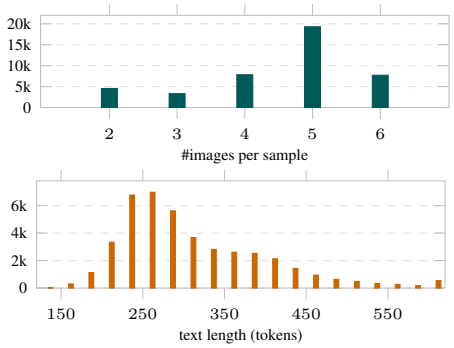

Figure 3: Sperry-43K training-split histograms.

2 Related Work

2.1 Multimodal CoT

Multimodal CoT (Zhang et al., 2024; Shao et al., 2024a; Su et al., 2025) differs in how visual information enters reasoning. External tools inspect or edit images (Wu et al., 2025b; Wang et al., 2025; Fu et al., 2025; Zhang et al., 2025c;a), or draw new representations (Hu et al., 2024; Wu et al., 2025a; Meng et al., 2023; Menon et al., 2024; Ning et al., 2026). InterleaveThinker (Zheng et al., 2026) coordinates separate planning, generation, and critique agents. Latent methods (Li et al., 2025b; Wang et al., 2026b; Yang et al., 2025; Zhang et al., 2025b; Chen et al., 2025a) use visual tokens; SpecFlow (Shen et al., 2026) instead updates a bounded spectral visual workspace.

Native visual reasoning includes image-token planning (Li et al., 2025c; Xu et al., 2025b), image-to-image solutions (He et al., 2025), self-critique (Chern et al., 2025), and interleaved SFT (Li et al., 2025a; Qin et al., 2025; Gu et al., 2025). VisWorld (Wu et al., 2026) studies generated visual states; COOPER (Zhang et al., 2025f) generates depth and segmentation for spatial reasoning. MentisOculi (Zeller et al., 2026) finds that visual thoughts do not consistently help, motivating checks of both generation fidelity and image use.

Preference optimization and RL improve reasoning-driven and interleaved generation (Guo et al., 2025b; Nie et al., 2026; Liu et al., 2026). MoTiF (Li et al., 2026) combines solver-rendered traces, reflective SFT, and Flow-GRPO with VLM-judged transition rewards; BRAID (Hu et al., 2026) jointly optimizes text and image policies using trajectory and visual-utility feedback. Sperry combines simulator-grounded supervision with separate expert training and generation-only RL on self-prompts.

2.2 Unified Multimodal Models

Unified multimodal models (Zhang et al., 2025d) support visual understanding and generation in one architecture: autoregressive models with a shared discrete token space (Chameleon Team, 2024; Wang et al., 2024b; Wu et al., 2025c; Meituan LongCat Team, 2026), decoupled designs with separate visual pathways (Wu et al., 2024; Chen et al., 2025b), and hybrid autoregressive-diffusion models (Zhou et al., 2024; Xie et al., 2024; 2025; Deng et al., 2025; Shi et al., 2026; Ma et al., 2025).

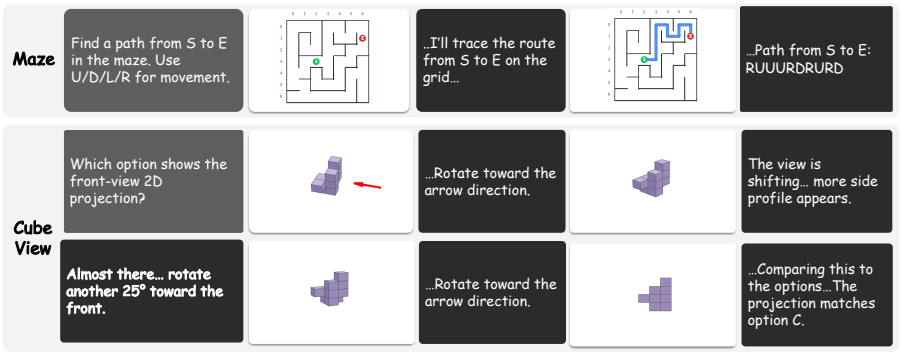

Figure 4: Representative Sperry-43K samples from *Maze* and *CubeView*.

3 Dataset and Benchmark

We focus on tasks where an intermediate reasoning step corresponds to a visual transformation, that is, updating a source image to reveal a new state. Such steps have an updated image as their natural representation; expressing them in text is possible but indirect. This setting covers two types of capabilities. The first is simulation and imagination, where the model mentally executes a sequence of edits on an image. The second is spatial intelligence (Wang et al., 2026a), where solving the task requires a 3D or scene representation that image generation models can represent more naturally than pure VLMs. Following this criterion, we construct Sperry-43K for training and SperryBench for evaluating these visual reasoning capabilities.

Sperry-43K. Sperry-43K contains 42,682 training samples across six subsets; together with the 2,249 SperryBench evaluation samples, the full release contains 44,931 samples. Each sample consists of a problem input, an interleaved text-and-image reasoning trace, and a final answer. Four are hand-designed: *Maze*, *BlockCount*, *CubeView*, and *LineTrack*, chosen because they specifically require the simulation/imagination and spatial-intelligence capabilities discussed above. The other two, *Counting* and *Jigsaw*, are filtered from Zebra-CoT (Li et al., 2025a), an open-source interleaved reasoning dataset. Figs. 4 and 5 show example traces from the four hand-designed subsets, with some intermediate steps omitted for space; details can be found in the Appendix.

Fig. 2 shows the four-stage pipeline used to construct the dataset. **Stage 1** filters the candidate task pool to those that require visual modeling, rejecting tasks whose intermediate states are naturally expressed in text or code (for example, arithmetic or symbolic manipulation). **Stage 2A** samples a problem instance and uses a task-specific simulator to render the deterministic sequence of visual states that constitutes its solution; because rendering is rule-based, every intermediate image and the final answer of the four hand-designed subsets are correct by construction. **Stage 2B** samples from a fixed pool of templated step instructions (e.g., "Highlight the first layer") and uses an LLM to paraphrase each into a diverse reasoning process, which reduces verbatim memorization of the templates. **Stage 3** interleaves the two chains, with each reasoning segment preceding the image it describes, mirroring the alternation between text and image generation at inference, and verifies each assembled trace by feeding it to Gemini 3.1 Pro as a multi-turn conversation, keeping only samples for which the verifier recovers the simulator's ground-truth answer. Tab. 1 reports the per-subset counts; SFT and RL use at most 6,000 traces per subset (31,474 in total; Sec. A.1), and Fig. 3 shows the distributions of images-per-sample and text length.

SperryBench. SperryBench evaluates models' visual reasoning performance across six subtasks. Constructed with the same pipeline as Sperry-43K, it contains 2,249 samples stratified across these subtasks and excluded from every training stage (SFT, RL, and the Monet retraining baseline). We report per-subset accuracy and an overall score.

4 Method

4.1 Preliminaries

DiffusionNFT. DiffusionNFT (Zheng et al., 2025a) is an online RL method for aligning flow-matching models with reward signals. For a rectified-flow model (Lipman et al., 2023; Liu et al.,

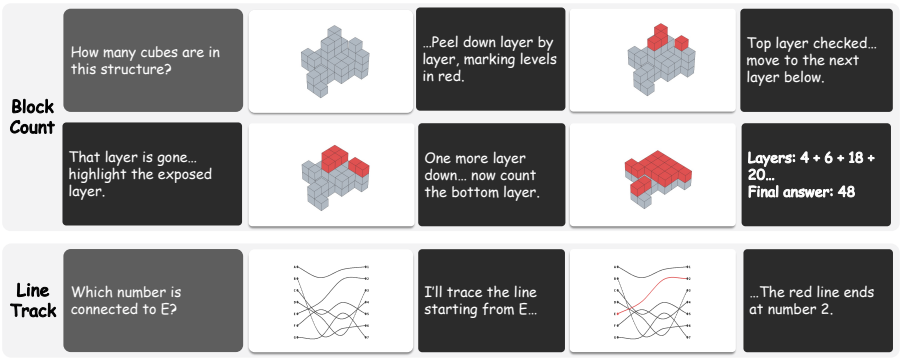

Figure 5: Representative Sperry-43K samples from *BlockCount* and *LineTrack*.

2022) with forward process $x_t = (1-t)\,x_0 + t\,\epsilon$ and velocity target $v = \epsilon - x_0$, it samples a group of G images per prompt from π^{old}, scores each with a reward $r(x_0, c) \in [0, 1]$, and implicitly decomposes π^{old} into positive/negative distributions whose velocities are $v_\theta^+ = (1-\beta)\,v^{\text{old}} + \beta\,v_\theta$ and $v_\theta^- = (1+\beta)\,v^{\text{old}} - \beta\,v_\theta$. The model is updated via a contrastive loss on the *forward* process:

$$\mathcal{L}_{\text{NFT}} = \mathbb{E}_{c,\,x_0\sim\pi^{\text{old}},\,t,\,\epsilon}\Big[r\,\|v_\theta^+(x_t,c,t) - v\|^2 + (1-r)\,\|v_\theta^-(x_t,c,t) - v\|^2\Big]. \tag{1}$$

It makes DiffusionNFT solver-agnostic and likelihood-free, avoiding the trajectory storage overhead of reverse-process alternatives (Black et al., 2024; Fan et al., 2023; Liu et al., 2025).

4.2 Model

Sperry is built on Bagel (Deng et al., 2025), a unified multimodal model with a Mixture-of-Transformers (MoT) backbone (Liang et al., 2025) composed of a 7B understanding branch and a 7B generation branch. Text tokens are decoded autoregressively, while image tokens are sampled by iterative rectified-flow matching. Both branches share the same multimodal context.

A multimodal reasoning instance consists of a problem input $Q = (Q_T,\, Q_{I_1}, \ldots, Q_{I_N})$, namely question text Q_T paired with N input images, and a final answer A, as shown in Fig. 6. Sperry produces an interleaved chain-of-thought

$$T \;=\; \big(R_{t,1},\, R_{i,1},\, R_{t,2},\, R_{i,2},\, \ldots,\, R_{t,K},\, R_{i,K},\, A\big), \tag{2}$$

where each $R_{t,k}$ is a text segment ending with a special `<abs_vis_token>` that marks the call to image generation, and $R_{i,k}$ is the intermediate reasoning image then sampled by the generation branch conditioned on $(Q,\, R_{<k})$. After K such alternations, the understanding branch decodes a closing text segment and the final answer A.

4.3 Supervised Fine-Tuning

SFT trains the unified model to produce the interleaved reasoning traces defined in Eq. 2: the understanding branch learns when to invoke the generator and what prompt to issue, while the generation branch learns to produce the target image given the multimodal context. The two branches optimize different training objectives:

$$\begin{aligned} \mathcal{L}_{\text{txt}}(\phi) &= -\mathbb{E}_{T,\,u}\,\log q_\phi\big(u \,\big|\, Q,\, T_{<u}\big), \\ \mathcal{L}_{\text{img}}(\theta) &= \mathbb{E}_{T,\,k,\,t,\,\epsilon}\,\big\|\, v_\theta\big(x_t,\, Q,\, R_{<k},\, t\big) - v \,\big\|^2, \end{aligned} \tag{3}$$

where u and k index the text tokens and image positions in trace T, $T_{<u}$ is the multimodal prefix (text *and* image tokens) preceding token u, $R_{<k} = (R_{t,1}, R_{i,1}, \ldots, R_{i,k-1}, R_{t,k})$ is the prefix preceding the k-th image, including the text segment $R_{t,k}$ that requests it, $x_0 = R_{i,k}$, and x_t, v follow Eq. 1. The two losses solve different modeling problems: probability modeling over discrete language tokens, and continuous density modeling in visual latent space. The two problems have inherently different optimization landscapes, and the branches accordingly need very different training durations: the understanding branch is best trained for one to two epochs, whereas the generation branch typically requires an order of magnitude more updates. Under a single schedule, stopping early leaves

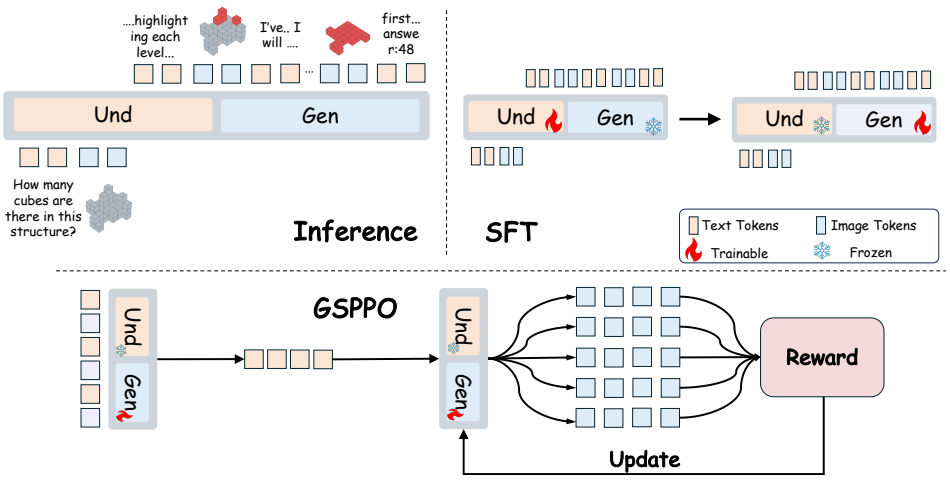

Figure 6: Overview of Sperry's training pipeline: two-stage SFT followed by GSPPO.

images undertrained, whereas training long enough for images to converge causes the text policy to memorize Sperry-43K and lose test-time generalization.

As shown in Fig. 6, we therefore train the two branches sequentially. In Stage 1, we freeze the generation branch and train the understanding branch on the full reasoning trace. In Stage 2, we freeze the understanding branch and train the generation branch for an order of magnitude more updates. The order follows the dependency between the two experts: in Bagel's mixture-of-transformers, image tokens attend to the keys and values that the understanding expert computes for the text, so fixing the understanding expert first gives the generator a stationary conditioning signal to fit.

4.4 Reinforcement Learning

After SFT, the reasoning text follows the supervised format closely: traces are well formed, image calls occur at the expected steps, and per-step prompts read like the training data. However, the intermediate images still have room for improvement: generated states sometimes deviate from the required one, for example a path through the wrong cells in Maze or an overshooting rotation in CubeView. We therefore introduce Generative Self-Prompted Policy Optimization (GSPPO), an RL stage that targets only the generation branch and leaves the SFT text policy fixed.

GSPPO instantiates DiffusionNFT inside the unified model. During SFT, the generation branch is trained with prompts drawn from Sperry-43K. At inference, however, those prompts are produced by the model's own text policy q_ϕ, which after SFT does not reproduce the dataset's prompt distribution exactly. Applying DiffusionNFT with dataset prompts would therefore train the generation branch on prompts it does not encounter at inference. We close this gap by drawing the prompt from the model itself. Each RL example is one image-generation step of a training trace: the problem Q, the interleaved prefix s up to that step, and the ground-truth image for the step. Given (Q, s), the frozen understanding branch regenerates the text segment $y \sim q_\phi(\cdot|Q, s)$ that requests the image, so the generator is conditioned on a self-written prompt rather than the dataset's. Conditioned on this shared (s, y) (the prefix s includes the problem Q), we then sample a group of G candidate images $\{x_0^g\}_{g=1}^G \sim \pi_{\text{img,old}}(\cdot|y, s)$, score each with a reward $r_g = r(x_0^g; I^\star, P) \in [0, 1]$ computed from the image alone against the ground-truth image $I^\star$ stored for this step and the problem image P (the reward does not read y, so the regenerated prompt itself is not checked), and update only the generation branch on this group:

$$\mathcal{L}_{\text{GSPPO}}(\theta;\phi) = \mathbb{E}_{Q,s\sim\mathcal{D},\, y\sim q_\phi(\cdot|Q,s),\, x_0^g\sim\pi_{\text{img,old}}(\cdot|y,s),\, t,\, \epsilon} \frac{1}{G}\sum_{g=1}^{G}\Big[w_g\, \ell_t\big(v_\theta^+(x_t^g, y, s, t),\, v^g\big) + (1-w_g)\, \ell_t\big(v_\theta^-(x_t^g, y, s, t),\, v^g\big)\Big], \qquad \ell_t(\hat{v}, v) = \frac{t\,\|\hat{v}-v\|_2^2}{\text{sg}\big(\|\hat{v}-v\|_1\big)}, \quad (4)$$

where $x_t^g = (1-t)\, x_0^g + t\,\epsilon$ and $v^g = \epsilon - x_0^g$ follow Eq. 1, and G (the RL group size) is distinct from the trace length K in Eq. 2. Following the DiffusionNFT implementation, the weight $w_g \in [0, 1]$ is

Table 2: SperryBench accuracy (%) with thinking enabled for all evaluated model sizes. **Bold**/underline: best/second-best open-source result per column; sizes are total parameters.

Model	Size	Maze	BlockCount	CubeView	LineTrack	Counting	Jigsaw	Overall
Baselines								
Random Guess	–	0.0	3.6	25.0	14.3	5.9	25.0	12.3
Blind	35B	0.0	3.0	13.6	13.3	30.3	21.3	13.6
Proprietary Models								
GPT-5.4 (medium)	–	72.1	9.1	42.9	53.0	84.6	76.3	56.3
GPT-5.4 (high)	–	74.6	4.7	39.7	55.8	87.4	79.3	56.9
Open-source Models								
Qwen2-VL	7B	0.0	9.1	19.4	14.1	47.3	22.6	18.8
Qwen2.5-VL	7B	0.0	5.5	28.6	15.5	53.4	22.2	20.9
Qwen3-VL	2B	0.1	2.1	7.1	14.2	49.3	22.7	15.9
	8B	0.8	8.3	26.4	16.5	78.6	40.2	28.5
	32B	0.0	6.1	5.9	15.8	83.4	47.5	26.5
	235B	2.8	12.0	34.7	19.1	<u>85.4</u>	49.3	33.9
Qwen3.5	9B	6.7	13.5	36.7	19.0	83.4	57.3	36.1
	27B	16.3	10.1	15.4	28.0	82.3	<u>68.8</u>	36.8
	35B	12.1	11.5	42.0	22.7	**86.9**	60.6	39.3
	397B	32.9	14.5	44.7	25.0	84.0	**70.8**	<u>45.3</u>
Gemma-4	E2B	0.8	8.1	26.1	12.7	50.9	31.0	21.6
	E4B	1.2	4.0	24.7	13.3	58.3	34.6	22.7
	26B	3.8	2.7	33.8	19.0	69.7	47.9	29.5
	31B	15.0	9.5	30.8	24.0	79.4	67.2	37.7
Kimi-K2.5	1T	14.6	4.0	48.8	<u>54.0</u>	84.6	59.6	44.3
Latent Reasoning Models								
Qwen2.5-VL	7B	0.0	5.5	28.6	15.5	53.4	22.2	20.9
+ Monet		2.5	16.2	51.0	34.5	61.6	48.9	35.8
Ours								
Bagel (base)		0.1	8.4	19.4	23.3	46.6	17.1	19.1
+ SFT	14B	<u>41.8</u>	<u>26.8</u>	<u>51.8</u>	42.3	62.1	36.2	43.5
+ GSPPO		**70.0**	**31.4**	**54.0**	**56.7**	59.2	40.2	**51.9**

an affine map of the clipped advantage: $\hat{A}_g = \mathrm{clip}\big((r_g - \bar{r}_{\text{group}})/\sigma_{\text{all}},\ -5,\ 5\big)$ with the mean taken within the group and the standard deviation over all rewards of the epoch, and $w_g = (\hat{A}_g + 5)/10$, so above-average candidates weight the positive branch and below-average candidates the negative branch. The per-branch loss ℓ_t is the DiffusionNFT implementation's squared error of the implied clean latent $\hat{x}_0 = x_t^g - t\,\hat{v}$ against x_0^g, normalized by its detached (sg) ℓ_1 error, with norms taken as means over latent dimensions. The old policy $\pi_{\text{img,old}}$ is an exponential moving average of the trainable weights, and only LoRA adapters (Hu et al., 2021) on the generation branch are trained. Full settings are in Sec. A.1. The understanding branch participates only as a sampler; gradients flow through v_θ alone, never through q_ϕ. Its parameters are fixed, although its conditional text distribution can still shift slightly, because text tokens attend to keys and values that the generation expert computes for VAE-encoded context images (Sec. 5.2). This is intentional: SFT has already produced a text policy we trust, and exposing it to a noisy image-derived reward would risk degrading the when-to-call and what-to-ask behaviors that two-stage SFT was structured to install.

Every subset in Sperry-43K admits a programmatic reward: each candidate image is scored by a few lines of image-processing code against the ground-truth intermediate image stored for that step of the trace, without decoding any text and without a learned reward model. The rules are simple and task-shaped: Maze compares the set of grid cells covered by the drawn path; BlockCount and LineTrack compare the red highlight (top layer or traced segment) by intersection-over-union; CubeView compares object silhouettes by IoU; Counting compares the multiset of colored objects detected by HSV segmentation; each of these returns $r \in \{0, 1\}$. Jigsaw, whose state is a filled photograph, receives a continuous $r \in [0, 1]$ from the per-region pixel error of the filled tiles. Because the rules are deterministic, the same pair of generated and reference images always receives the same score. Sec. A.3 gives the exact thresholds.

Table 3: BabyVision (Chen et al., 2026) accuracy (%). 2D/3D PC: 2D/3D pattern completion; 3D V.: 3D visualization; Rot.: rotation.

Model	2D PC	3D PC	3D V.	Rot.	Maze
Qwen3-VL-Plus	30.0	27.8	27.8	30.0	2.5
Qwen3-VL-235B	25.0	29.6	29.6	40.0	11.7
Bagel	60.0	55.6	**44.4**	**50.0**	20.0
Sperry (Ours)	**75.0**	**77.8**	**44.4**	40.0	**30.0**

Table 4: VisuLogic (Xu et al., 2025a) accuracy (%). Pos.: positional; Sty.: stylistic; Spa.: spatial; Oth.: other; Qua.: quantitative; Att.: attribute reasoning.

Model	Pos.	Sty.	Spa.	Oth.	Qua.	Att.	Overall
Bagel	13.2	15.6	16.5	11.1	12.0	7.7	13.1
Sperry (Ours)	**28.4**	**26.3**	**24.8**	**24.1**	**23.3**	**20.7**	**24.5**

Table 5: Accuracy (%) per backbone

Backbone	Base	SFT
Bagel	19.1	43.5
SenseNova-U1	17.8	**63.8**

Table 6: Transfer and retention (%; GenEval 0–1);

Evaluation	Bagel	SFT	RL
VisWorld Maze-I	–	37.1	52.7
MMBench v1.1	82.4	85.1	83.5
GenEval	0.82	–	0.78

Table 7: DiffusionNFT vs. GSPPO

RL Variant	Maze	Blk.	Cube	Line	Cnt.	Jig.	Avg.
SFT only	41.8	26.8	51.8	42.3	62.1	36.2	43.5
+ DiffusionNFT	59.7	27.9	**54.6**	55.9	**64.3**	**41.0**	50.6
+ GSPPO (Ours)	**70.0**	**31.4**	54.0	**56.7**	59.2	40.2	**51.9**

5 Experiments

5.1 Setup

Benchmark and baselines. SperryBench accuracy is the unweighted mean over six subsets. All baselines except the retrained Monet are evaluated zero-shot. Baselines are Qwen2-VL (Wang et al., 2024a), Qwen2.5-VL (Bai et al., 2025b), Qwen3-VL (Bai et al., 2025a), Qwen3.5 (Qwen Team, 2026), Gemma-4 (Farabet & Lacombe, 2026), and Kimi-K2.5 (Kimi Team, 2026) (thinking mode in Table 2, instruct mode in the Appendix), GPT-5.4 (OpenAI, 2026b), and Monet (Wang et al., 2026b), a latent visual reasoning method; Random Guess and Blind (Qwen3.5-35B, question text only) rows bound Table 2. Sperry builds on Bagel-7B-MoT; settings are in Secs. A.1–A.2 and all models share one grader (Sec. B.1).

5.2 Main Results

We compare Sperry with open-source VLMs, GPT-5.4, and Monet (Table 2). At 14B total parameters, Sperry reaches **51.9%** overall, **+6.6** points over Qwen3.5-397B. SFT lifts Bagel from 19.1% to 43.5%, and RL adds +8.4. Monet, retrained on Sperry-43K with its official recipe including RL, reaches 35.8%. On a second UMM architecture, SenseNova-U1 (Diao et al., 2026), the same SFT recipe improves accuracy from 17.8% to 63.8% (Table 5), demonstrating transfer across backbones.

Also, to test transfer and retention, we evaluate on BabyVision (Chen et al., 2026), VisuLogic (Xu et al., 2025a), VisWorld-Eval Maze-I (Wu et al., 2026), MMBench (Liu et al., 2023), and GenEval (Ghosh et al., 2023). Sperry improves over Bagel on most BabyVision subtypes but not Rotation (Tab. 3) and all six VisuLogic categories (Tab. 4); Maze-I image accuracy rises from 37.1% to 52.7%, with MMBench and GenEval essentially retained (Table 6). Thus, the learned visual operations transfer across renderings while preserving general ability.

5.3 Other Experiments

Two-stage vs. joint SFT. The understanding and generation branches converge at different rates: the text policy learns the reasoning format quickly, while the generator needs more updates to reproduce visual state transitions (Sec. 4.3). We therefore test whether training the branches separately improves intermediate-image correctness over jointly updating both experts. Tab. 8 reports image reward, the percentage of traces whose intermediate images all pass the task-specific checks in Sec. A.3. Two-stage SFT raises this score from 35.1 to 41.2, a gain of 6.1 points. Joint-SFT answer accuracy is not reported in this comparison. The image-reward gain supports separating the training schedules to better fit the visual states used in reasoning.

GSPPO vs. vanilla DiffusionNFT. At inference, the generator receives prompts written by the understanding branch, whereas vanilla DiffusionNFT trains it on the dataset's text segments. To test the effect of this mismatch, we keep the RL pipeline fixed and replace dataset prompts with prompts sampled from the frozen understanding branch (Sec. A.1). As shown in Tab. 7, GSPPO improves overall accuracy from 50.6% to 51.9%, with gains of 10.3 points on Maze and 3.5 points on

Table 8: Same-backbone SFT controls: answer accuracy (%) and image reward (traces passing every intermediate-image check in Sec. A.3, $\times 100$). Joint SFT reports reward only.

Model	Maze	BlockCount	CubeView	LineTrack	Counting	Jigsaw	Overall	Reward
Sperry SFT	**41.8**	**26.8**	**51.8**	**42.3**	62.1	**36.2**	**43.5**	41.2
image → problem image	0.9	7.2	42.4	23.7	59.2	19.8	25.5	–
image → blank canvas	0.7	12.4	44.0	16.9	49.6	28.1	25.3	–
Stage-1 only (base generator)	0.8	3.2	42.7	20.5	56.5	27.6	25.2	1.7
Text-only SFT	3.6	26.0	46.4	31.6	**63.9**	30.0	33.6	–
Joint SFT	–	–	–	–	–	–	–	35.1

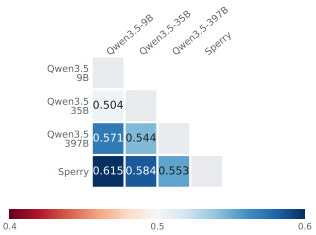

Figure 7: Pairwise win rate.

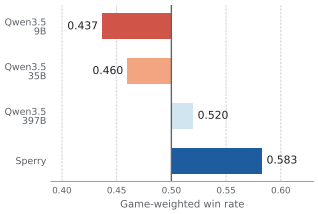

Figure 8: Overall win rate.

Table 9: Oracle context (%).

Subset	GPT-5.4	Q3.5-397B	Sperry
Maze	**93.3**	74.2	61.7
Blk.	5.7	26.0	**32.1**
Cube	**91.4**	81.2	66.0
Line	98.2	99.3	**99.8**
Cnt.	**97.7**	**97.7**	81.7
Jig.	**96.8**	94.8	44.9
Avg.	**80.5**	78.9	64.4

BlockCount. The effect varies across tasks: Counting decreases from 64.3% to 59.2%, and CubeView and Jigsaw also decline slightly. These results support training the generator on its own text policy's prompts, with the overall benefit concentrated in Maze and BlockCount.

Sperry relies on the images it generates. Task-specific supervision can improve the text policy as well as the generator. We therefore test both whether the trained model uses its intermediate images and whether learning to generate them improves answer accuracy. At inference, replacing each generated image with the problem image or a blank canvas lowers SFT accuracy from 43.5% to 25.5% and 25.3%, respectively (Tab. 8). Maze falls from 41.8% to below 1% under either substitution. Counting changes little when the problem image is reused (62.1% to 59.2%), consistent with its intermediate images largely repeating the input, but falls to 49.6% with a blank canvas. For training controls, Stage-1-only SFT with the base generator reaches 25.2%, and epoch-matched text-only SFT reaches 33.6% (Sec. A.1). Together, these results show that the model uses generated visual states and that training the generator contributes to accuracy beyond text-only task supervision.

Sperry learns effective multimodal reasoning traces. Successful reasoning requires both constructing useful intermediate states and interpreting them to obtain an answer. We examine these abilities with two complementary evaluations. First, we provide reference reasoning traces as oracle context, so each model predicts the answer from an already constructed trace (Tab. 9). GPT-5.4 and Qwen3.5-397B reach 80.5% and 78.9%, respectively, compared with Sperry's 64.4%. Yet in standard evaluation, where models construct their own traces, Sperry exceeds Qwen3.5-397B by 6.6 points. Second, GPT-5.4 judges the helpfulness of model-generated traces pairwise on the four hand-designed subsets (Sec. B.3). Sperry ranks first overall with a 58.3% win rate, compared with 52.0% for Qwen3.5-397B (Figs. 7–8), and leads on three of the four subsets. Together, the oracle-context comparison and helpfulness judgments suggest that constructing informative visual traces contributes to Sperry's advantage over Qwen3.5-397B.

Inference efficiency. Generating intermediate images introduces iterative sampling, so we measure whether this cost makes complete visual reasoning trajectories slower than text reasoning. We time both models on the same problems under their evaluation decoding settings (Sec. A.2). Sperry averages 46.0 s per problem on one H100, including image generation, while Qwen3.5-9B served with vLLM (Kwon et al., 2023) takes about three minutes. Under these configurations, Sperry achieves lower end-to-end latency while also attaining higher SperryBench accuracy (51.9% versus 36.1%), showing that explicit visual reasoning can offer a favorable accuracy–latency trade-off.

6 Conclusion

We presented Sperry, which post-trains a unified multimodal model to reason over interleaved text-and-image traces via two-stage SFT and generation-branch RL with simulator-derived rewards (GSPPO). At 14B parameters it reaches 51.9% on SperryBench, above Qwen3.5-397B and Kimi-K2.5, and removing its generated images costs 10–18 points: it reasons over the states it draws.

AI USE STATEMENT

We used generative AI tools to assist with code implementation, benchmark baselines, and polishing the manuscript's language and presentation. AI models also participate in the data and evaluation pipelines described in this work. Qwen3.5-27B rewrites reasoning text for the simulator-generated traces (Sec. C.3); Gemini 3.1 Pro verifies these traces against simulator-derived answers (Sec. C.4); and Qwen3-VL-235B-A22B-Instruct-FP8 filters the Counting and Jigsaw subsets for intermediate-image helpfulness (Sec. C.5). Qwen3.5-397B-A17B assists with converting traces into the Monet baseline's training format (Sec. C.7), and GPT-5.4 judges the helpfulness of generated reasoning traces (Sec. B.3). The appendix documents the models, prompts, settings, and automated checks used in these stages. We take responsibility for the manuscript, code, data, and reported results, including the components produced with AI assistance.

ETHICS STATEMENT

This work studies visual reasoning through generated intermediate images. Four task subsets use synthetic scenes rendered by deterministic simulators; Counting and Jigsaw are derived from Zebra-CoT (Li et al., 2025a) and include natural photographs, some depicting people in public scenes. Reuse of these data and the pretrained models should respect their source licenses and privacy constraints. Generated reasoning images can be incorrect, and visual plausibility alone does not establish the correctness of a reasoning trace. Downstream applications should retain the base model's safety safeguards and evaluate errors in their intended setting.

REPRODUCIBILITY STATEMENT

Sections 4.3 and 4.4 describe the training objectives, while Sections A.1–A.3 provide training and inference settings, hardware, sampling configurations, and reward definitions. Sections B.1 and B.3 specify answer grading and the helpfulness evaluation. Sections C.1–C.5 document the simulators, text templates, rewriting prompts, verification, filtering, and separation of training and evaluation splits; Section C.7 details the Monet data conversion. The data pipeline distinguishes deterministic simulator stages from sampled, API-dependent model stages and describes the use of cached outputs to reproduce the released split (Sec. C.5).

REFERENCES

Anthropic. Introducing claude opus 4.7. https://www.anthropic.com/news/claude-opus-4-7, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning, 2024. URL https://arxiv.org/abs/2305.13301.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. doi: 10.48550/arXiv.2405.09818. URL https://github.com/facebookresearch/chameleon.

Chao Chen, Zhixin Ma, Yongqi Li, Yupeng Hu, Yinwei Wei, Wenjie Li, and Liqiang Nie. Reasoning in the dark: Interleaved vision-text reasoning in latent space, 2025a. URL https://arxiv.org/abs/2510.12603.

Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Y. charles, Yiping Bao, Yuantao Fan, Guopeng Li, Haiyang Shen, Xuanzhong Chen, Wendong Xu, Shuzheng Si, Zefan Cai, Wenhao Chai, Ziqi Huang, Fangfu Liu, Tianyu Liu, Baobao Chang, Xiaobo Hu, Kaiyuan Chen, Yixin Ren, Yang Liu, Yuan Gong, and Kuan Li. BabyVision: Visual reasoning beyond language, 2026. URL https://arxiv.org/abs/2601.06521v1.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025b.

Ethan Chern, Zhulin Hu, Steffi Chern, Siqi Kou, Jiadi Su, Yan Ma, Zhijie Deng, and Pengfei Liu. Thinking with generated images. *arXiv preprint arXiv:2505.22525*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

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. Emerging properties in unified multimodal pretraining, 2025. URL https://arxiv.org/abs/2505.14683.

Haiwen Diao, Penghao Wu, Hanming Deng, Jiahao Wang, Shihao Bai, et al. SenseNova-U1: Unifying multimodal understanding and generation with NEO-unify architecture, 2026. URL https://arxiv.org/abs/2605.12500.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2305.16381.

Clement Farabet and Olivier Lacombe. Gemma 4: Byte for byte, the most capable open models. https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/, 2026.

Xingyu Fu, Minqian Liu, Zhengyuan Yang, John Corring, Yijuan Lu, Jianwei Yang, Dan Roth, Dinei Florencio, and Cha Zhang. Refocus: Visual editing as a chain of thought for structured image understanding, 2025. URL https://arxiv.org/abs/2501.05452.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment, 2023. URL https://arxiv.org/abs/2310.11513.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025a. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Ziyu Guo, Renrui Zhang, Chengzhuo Tong, Zhizheng Zhao, Rui Huang, Haoquan Zhang, Manyuan Zhang, Jiaming Liu, Shanghang Zhang, Peng Gao, Hongsheng Li, and Pheng-Ann Heng. Can we generate images with cot? let's verify and reinforce image generation step by step, 2025b. URL https://arxiv.org/abs/2501.13926.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy, 2020. URL https://arxiv.org/abs/2006.10256.

Zefeng He, Xiaoye Qu, Yafu Li, Tong Zhu, Siyuan Huang, and Yu Cheng. Diffthinker: Towards generative multimodal reasoning with diffusion models, 2025. URL https://arxiv.org/abs/2512.24165.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models, 2024. URL https://arxiv.org/abs/2406.09403.

Zican Hu, Xuyang Hu, Yiming Liu, Zuwei Long, Wei Liu, Yunzhuo Hao, Jiawei Gu, Linjie Li, Yu Cheng, Zhenhong Sun, Weibo Gu, Xing Sun, and Zhi Wang. Bridging interleaved multimodal reasoning as a unified decision process, 2026. URL https://arxiv.org/abs/2607.03748.

Michael Igorevich Ivanitskiy, Rusheb Shah, Alex F. Spies, Tilman Räuker, Dan Valentine, Can Rager, Lucia Quirke, Chris Mathwin, Guillaume Corlouer, Cecilia Diniz Behn, and Samy Wu Fung. A configurable library for generating and manipulating maze datasets, 2023. URL https://arxiv.org/abs/2309.10498.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *ICLR*, 2024. URL https://arxiv.org/abs/2310.06770.

Kimi Team. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Ang Li, Charles Wang, Deqing Fu, Kaiyu Yue, Zikui Cai, Wang Bill Zhu, Ollie Liu, Peng Guo, Willie Neiswanger, Furong Huang, Tom Goldstein, and Micah Goldblum. Zebra-cot: A dataset for interleaved vision language reasoning, 2025a. URL https://arxiv.org/abs/2507.16746.

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Hao Chen, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning, 2025b. URL https://arxiv.org/abs/2509.24251.

Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. Imagine while reasoning in space: Multimodal visualization-of-thought, 2025c. URL https://arxiv.org/abs/2501.07542.

Tingyu Li, Le Zhou, Siyuan Li, Yujun Wu, Xinglong Xu, Jingxuan Wei, Conghui He, and Cheng Tan. Bridging modal isolation in interleaved thinking: Supervising modality transitions via stepwise reinforcement, 2026. URL https://arxiv.org/abs/2606.12886.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models, 2025. URL https://arxiv.org/abs/2411.04996.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl, 2025. URL https://arxiv.org/abs/2505.05470.

Jie Liu, Zilyu Ye, Linxiao Yuan, Shenhan Zhu, Yu Gao, Jie Wu, Kunchang Li, Xionghui Wang, Xiaonan Nie, Weilin Huang, and Wanli Ouyang. Unigrpo: Unified policy optimization for reasoning-driven visual generation, 2026. URL https://arxiv.org/abs/2603.23500.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player?, 2023. URL https://arxiv.org/abs/2307.06281.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Yiyang Ma, Xingchao Liu, Xiaokang Chen, Wen Liu, Chengyue Wu, Zhiyu Wu, Zizheng Pan, Zhenda Xie, Haowei Zhang, Xingkai yu, Liang Zhao, Yisong Wang, Jiaying Liu, and Chong Ruan. Janusflow: Harmonizing autoregression and rectified flow for unified multimodal understanding and generation, 2025. URL https://arxiv.org/abs/2411.07975.

Meituan LongCat Team. Longcat-next: Lexicalizing modalities as discrete tokens, 2026. URL https://arxiv.org/abs/2603.27538.

Fanxu Meng, Haotong Yang, Yiding Wang, and Muhan Zhang. Chain of images for intuitively reasoning, 2023. URL https://arxiv.org/abs/2311.09241.

Sachit Menon, Richard Zemel, and Carl Vondrick. Whiteboard-of-thought: Thinking step-by-step across modalities, 2024. URL https://arxiv.org/abs/2406.14562.

Ming Nie, Chunwei Wang, Jianhua Han, Hang Xu, and Li Zhang. Towards unified multimodal interleaved generation via group relative policy optimization, 2026. URL https://arxiv.org/abs/2603.09538.

Zhiwei Ning, Wenwen Tong, Xiangli Kong, Shengnan Ma, Ziyi Shang, Jingcheng Ni, Tao Hu, Yong Xien Chng, Jixuan Ying, Zehuan Wu, Hanming Deng, Jie Yang, Yuanjie Zheng, Wei Liu, and Lewei Lu. Intersketch: An interleaved reasoning model with self-correcting visual sketch and stepwise reward, 2026. URL https://arxiv.org/abs/2605.26520.

OpenAI. OpenAI o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

OpenAI. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, 2025.

OpenAI. Gpt-5.5 system card. https://openai.com/index/gpt-5-5-system-card/, 2026a.

OpenAI. GPT-5.4 thinking system card. https://openai.com/index/gpt-5-4-thinking-system-card/, 2026b.

Luozheng Qin, Jia Gong, Yuqing Sun, Tianjiao Li, Mengping Yang, Xiaomeng Yang, Chao Qu, Zhiyu Tan, and Hao Li. Uni-CoT: Towards unified chain-of-thought reasoning across text and vision. *arXiv preprint arXiv:2508.05606*, 2025.

Qwen Team. Qwen3: Think deeper, act faster. https://qwenlm.github.io/blog/qwen3/, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026. Accessed 2026.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning, 2024a. URL https://arxiv.org/abs/2403.16999.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024b. URL https://arxiv.org/abs/2402.03300.

Yixian Shen, Zhiheng Yang, Qi Bi, Changshuo Wang, Shuai Wang, Jia-Hong Huang, George Floros, Prayag Tiwari, and Anuj Pathania. Spectral-progressive thought flow for lightweight multimodal reasoning, 2026. URL https://arxiv.org/abs/2606.02842.

Min Shi, Xiaohui Zeng, Jiannan Huang, Yin Cui, Francesco Ferroni, Jialuo Li, Shubham Pachori, Zhaoshuo Li, Yogesh Balaji, Haoxiang Wang, Tsung-Yi Lin, Xiao Fu, Yue Zhao, Chieh-Yun Chen, Ming-Yu Liu, and Humphrey Shi. Duogen: Towards general purpose interleaved multimodal generation, 2026. URL https://arxiv.org/abs/2602.00508.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, Linjie Li, Yu Cheng, Heng Ji, Junxian He, and Yi R. Fung. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers, 2025. URL https://arxiv.org/abs/2506.23918.

Haozhe Wang, Alex Su, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning, 2025. URL https://arxiv.org/abs/2505.15966.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.

Qineng Wang, Baiqiao Yin, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, Saining Xie, Jiajun Wu, Li Fei-Fei, and Manling Li. Mindcube: Spatial mental modeling from limited views, 2026a. URL https://arxiv.org/abs/2506.21458.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond image and language. In *CVPR*, 2026b. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_Monet_Reasoning_in_Latent_Visual_Space_Beyond_Image_and_Language_CVPR_2026_paper.html.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. *arXiv preprint arXiv:2410.13848*, 2024.

Jialong Wu, Xiaoying Zhang, Hongyi Yuan, Xiangcheng Zhang, Tianhao Huang, Changjing He, Chaoyi Deng, Renrui Zhang, Youbin Wu, and Mingsheng Long. Visual generation unlocks human-like reasoning through multimodal world models, 2026. URL https://arxiv.org/abs/2601.19834.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing, 2025a. URL https://arxiv.org/abs/2506.09965.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use, 2025b. URL https://arxiv.org/abs/2505.19255.

Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, Song Han, and Yao Lu. VILA-U: A unified foundation model integrating visual understanding and generation. In *International Conference on Learning Representations*, 2025c. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e9e140df6de01afb672cb859d203c307-Abstract-Conference.html.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024.

Jinheng Xie, Zhenheng Yang, and Mike Zheng Shou. Show-o2: Improved native unified multimodal models. *arXiv preprint arXiv:2506.15564*, 2025.

Weiye Xu, Jiahao Wang, Weiyun Wang, Zhe Chen, Wengang Zhou, Aijun Yang, Lewei Lu, Houqiang Li, Xiaohua Wang, Xizhou Zhu, Wenhai Wang, Jifeng Dai, and Jinguo Zhu. Visulogic: A benchmark for evaluating visual reasoning in multimodal large language models, 2025a. URL https://arxiv.org/abs/2504.15279.

Yi Xu, Chengzu Li, Han Zhou, Xingchen Wan, Caiqi Zhang, Anna Korhonen, and Ivan Vulić. Visual planning: Let's think only with images, 2025b. URL https://arxiv.org/abs/2505.11409.

Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens, 2025. URL https://arxiv.org/abs/2506.17218.

Jana Zeller, Thaddäus Wiedemer, Fanfei Li, Thomas Klein, Prasanna Mayilvahanan, Matthias Bethge, Felix Wichmann, Ryan Cotterell, and Wieland Brendel. Mentisoculi: Revealing the limits of reasoning with mental imagery, 2026. URL https://arxiv.org/abs/2602.02465.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

Chi Zhang, Haibo Qiu, Qiming Zhang, Zhixiong Zeng, Lin Ma, and Jing Zhang. Deepsketcher: Internalizing visual manipulation for multimodal reasoning, 2025a. URL https://arxiv.org/abs/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. Latent sketchpad: Sketching visual thoughts to elicit multimodal reasoning in mllms, 2025b. URL https://arxiv.org/abs/2510.24514.

Shuoshuo Zhang, Zijian Li, Yizhen Zhang, Jingjing Fu, Lei Song, Jiang Bian, Jun Zhang, Yujiu Yang, and Rui Wang. Pixelcraft: A multi-agent system for high-fidelity visual reasoning on structured images, 2025c. URL https://arxiv.org/abs/2509.25185.

Xinjie Zhang, Jintao Guo, Shanshan Zhao, Minghao Fu, Lunhao Duan, Guo-Hua Wang, Qing-Guo Chen, Zhao Xu, Weihua Luo, and Kaifu Zhang. Unified multimodal understanding and generation models: Advances, challenges, and opportunities, 2025d. URL https://arxiv.org/abs/2505.02567v1.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, et al. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*, 2025e.

Zefeng Zhang, Xiangzhao Hao, Hengzhu Tang, Zhenyu Zhang, Jiawei Sheng, Xiaodong Li, Zhenyang Li, Li Gao, Daiting Shi, Dawei Yin, and Tingwen Liu. Cooper: A unified model for cooperative perception and reasoning in spatial intelligence, 2025f. URL https://arxiv.org/abs/2512.04563.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models, 2024. URL https://arxiv.org/abs/2302.00923.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023. URL https://arxiv.org/abs/2304.11277.

Dian Zheng, Harry Lee, Manyuan Zhang, Kaituo Feng, Zoey Guo, Ray Zhang, and Hongsheng Li. Interleavethinker: Reinforcing agentic interleaved generation, 2026. URL https://arxiv.org/abs/2606.13679.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025a.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deepeyes: Incentivizing "thinking with images" via reinforcement learning, 2025b. URL https://arxiv.org/abs/2505.14362.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model, 2024. URL https://arxiv.org/abs/2408.11039.

Supplementary Material for "Sperry: Multimodal Reasoning with Unified Model"

This supplementary material is organized into three parts. **Part A (Training and inference details)** reports the hyperparameters of SFT and GSPPO, the inference configuration used at evaluation time, and the simulator-derived reward function. **Part B (Evaluation details)** provides non-thinking baseline results and the chain-of-thought helpfulness study. **Part C (Dataset)** documents the Sperry-43K image simulator, the fixed text pool, the LLM rewrite, the correctness checks and filtering pipeline, sample traces, and the conversion from Sperry-43K to the Monet training format.

CONTENTS

A Training details

A.1 Training Hyperparameters 19

A.2 Inference Configs 20

A.3 Reward Function 21

B Evaluation details

B.1 Answer Extraction and Grading 22

B.2 Non-thinking Results 23

B.3 Chain-of-Thought Helpfulness Evaluation 24

C Dataset

C.1 Sperry-43K Image Simulator 25

C.2 Sperry-43K Fixed Text Pool 26

C.3 LLM Rewrite Details 26

C.4 Correctness Verification Details 28

C.5 Data Filtering Pipeline 28

C.6 Data Samples 29

C.7 Monet Data Conversion 37

A.1 TRAINING HYPERPARAMETERS

We initialize from the public Bagel-7B-MoT checkpoint and post-train in three sub-stages: SFT Stage 1 (understanding), SFT Stage 2 (generation), and GSPPO. All sub-stages run in bf16 mixed precision on a single node of 8 NVIDIA H100 80 GB GPUs; SFT shards parameters with FSDP (Zhao et al., 2023) HYBRID_SHARD across the 8 GPUs, while GSPPO trains LoRA adapters on the same hardware. The training mixture did not include general-purpose image-generation replay.

SFT. Both SFT stages use AdamW (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\varepsilon = 10^{-15}$, weight decay 0, a 2,000-step linear warm-up to 2×10^{-5} followed by a constant learning rate; gradients are clipped at L2 norm 1.0. Training packs interleaved samples up to a budget of 32,768 tokens per micro-batch per GPU. Stage 1 trains the understanding branch only (ce_weight=1, mse_weight=0, freeze_gen=True) for 1,117 steps, about 1.5 epochs over the 31,474-trace mixture with token packing, after which the text loss has flattened (Sec. 4.3); this ends inside the warm-up, so the Stage-1 learning rate rises linearly to 1.1×10^{-5} and never reaches the plateau. Stage 2 resumes from the Stage-1 checkpoint with a fresh optimizer and schedule, swaps to the generation branch (ce_weight=0, mse_weight=1, freeze_und=True), and runs for 9,000 steps; rectified-flow timesteps use the standard shift 1.0 during SFT. The VAE is kept frozen throughout. Both stages draw from a capped mixture of the training split: all Maze (4,548), BlockCount (5,611), and Counting (3,315) traces and 6,000 traces each from CubeView, LineTrack, and Jigsaw, i.e. 31,474 of the 42,682 released training traces; the full split is released. Text-conditioning dropout is applied with probability 0.1; VAE and ViT conditioning dropout are configured at 0.3 but are inactive for interleaved samples. We track an exponential moving average of the trainable parameters with decay 0.9999 and use the EMA weights as the initialization for GSPPO.

GSPPO. The GSPPO stage updates the generation branch on top of the Stage-2 SFT checkpoint with LoRA (rank 64, $\alpha = 128$) inserted into the attention (q, k, v, o) and MLP (gate, up, down) projections of the generation expert in every transformer block; the understanding branch, the ViT, and the VAE remain frozen. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, weight decay 10^{-4}, learning rate 10^{-4}, gradient clipping at 1.0, and the same bf16 mixed-precision setup as SFT. The RL pool is built from the SFT traces: every intermediate image of a trace becomes one sub-problem consisting of the prefix up to that image and the image itself as reference, with the same per-subset caps as SFT and Maze and Counting sub-problems duplicated twice (about 120K sub-problems); each epoch draws sub-problems with per-subset sampling weights of 3 (Maze, LineTrack), 2 (Counting), and 0.5 (CubeView, BlockCount, Jigsaw). For each sub-problem we sample a group of $G = 16$ candidate images with one denoising trajectory each, score them with the reward of Sec. A.3, and form advantages by subtracting the group mean and dividing by the standard deviation of all rewards in the epoch, clipped to $[-5, 5]$ and mapped affinely to the branch weight $w = (\hat{A} + 5)/10$; the per-sample loss is scaled by a constant 5, which only rescales the effective learning rate. We follow the off-policy schedule of DiffusionNFT: rollouts and old-policy velocities come from an EMA copy of the LoRA weights whose decay at epoch e is $\min(10^{-3}e, 0.5)$, so the reference tracks the current policy almost exactly in early epochs and reaches decay 0.1 at epoch 100; it is updated every epoch and held on GPU. Each training epoch consists of 384 unique sub-problems (6,144 rollouts) and 1 gradient step, accumulated over all rollouts and both sampled timesteps; following DiffusionNFT's terminology, an "epoch" here is one such batch and update, not a pass over the pool. Diffusion training uses the rectified-flow ODE formulation with shift 3.0, 10 inference steps per rollout, and two timesteps per sample drawn from the seven highest-noise steps of the 10-step schedule ($t \in [0.67, 1]$); classifier-free guidance is disabled during rollout sampling (cfg_text=cfg_img=1.0). We disable a separate KL penalty (kl_beta=0) since the EMA reference already constrains policy drift, and set the NFT weight $\beta_{\mathrm{NFT}} = 1$. All images, both training conditions and generations, are at resolution 512×512 after resizing. The vanilla DiffusionNFT baseline of Tab. 7 uses the same pipeline but conditions on the dataset's own text segment instead of a regenerated one. The regenerated segment is sampled from the frozen understanding branch at temperature 0.7 with a 500-token cap.

Same-backbone controls (Tab. 8). *Image-disabled inference* runs the two-stage SFT model through the standard interleaved loop but skips the diffusion step whenever an image is requested, feeding back either the problem image or a blank canvas through the same VAE and ViT encoders; no weights change. *Stage-1 only* evaluates the checkpoint after SFT Stage 1 with Bagel's original generation branch. *Text-only SFT* trains the understanding branch on the same 31,474 traces with

Table A.1: Training hyperparameters for SFT Stage 1, SFT Stage 2, and GSPPO. "—" marks parameters that do not apply to that stage; "=S1" means the same value as Stage 1.

	SFT Stage 1	**SFT Stage 2**	**GSPPO**
Modules			
trainable branch	understanding	generation	generation (LoRA)
frozen modules	gen branch, VAE	und branch, VAE	und branch, VAE, base gen weights
Optimizer			
optimizer	AdamW	AdamW	AdamW
(β_1, β_2)	(0.9, 0.95)	=S1	(0.9, 0.999)
ε	10^{-15}	=S1	10^{-8}
weight decay	0	0	10^{-4}
peak learning rate	2×10^{-5} (1.1×10^{-5} reached)	2×10^{-5}	10^{-4}
LR schedule	2k linear warm-up (stopped at 1,117)	2k warm-up, then const.	constant
gradient clip (ℓ_2)	1.0	1.0	1.0
Batching & schedule			
batch budget	32,768 tokens/GPU	32,768 tokens/GPU	384 sub-problems/epoch, G=16
#updates	1,117 steps	9,000 steps	$\approx$ 100 epochs (1 update each)
text loss weight (CE)	1.0	0.0	—
image loss weight (MSE / RF-NFT)	0.0	1.0	rectified-flow NFT
EMA decay	0.9999	0.9999	$\min(10^{-3}e,\ 0.5)$ at epoch e
modality dropout (text/vae/vit)	0.1 / inactive / inactive	0.1 / inactive / inactive	—
LoRA (GSPPO only)			
LoRA rank r / scale α	—	—	64 / 128
LoRA target modules	—	—	gen-expert attn. (q, k, v, o) + MLP (gate, up, down)
LoRA dropout	—	—	0
GSPPO-specific			
group size G	—	—	16
advantage clip / branch weight	—	—	± 5 / $w = (\hat{A} + 5)/10$
KL coef. (kl_beta)	—	—	0 (EMA reference instead)
NFT weight β_{NFT}	—	—	1
RF timestep shift	1.0	1.0	3.0
training timesteps	full schedule	full schedule	2 of the 7 highest-noise steps
System			
hardware	8×H100 80 GB	8×H100 80 GB	8×H100 80 GB
precision	bf16 mixed	bf16 mixed	bf16 mixed
parallelism	FSDP HYBRID_SHARD	FSDP HYBRID_SHARD	DDP + LoRA

every intermediate image removed and the rationale text kept, matched to Stage 1 by epochs rather than optimizer steps: text-only traces pack about 3.9× denser into the 32,768-token packed sequences, so 1.46 epochs correspond to 290 packed steps instead of 1,117, and the linear warm-up is rescaled in proportion (520 steps) so that both runs end at the same fraction of the peak learning rate. All other settings follow the Stage-1 column of Tab. A.1.

A.2 Inference Configs

We use a single inference configuration for Sperry on SperryBench and on the chain-of-thought helpfulness evaluation of Sec. B.3. Each problem is decoded $n = 8$ independent times; the SperryBench accuracies reported in the paper are the mean accuracy over these eight attempts (an eight-sample estimate of pass@1), and the helpfulness study uses the first trace. The full configuration is listed in Tab. A.2. Text is sampled autoregressively with temperature 0.5, the model's default nucleus parameter top-$p = 1.0$, and a budget of 2,000 new tokens per text segment. Each intermediate image is generated by 50 steps of rectified-flow ODE integration with timestep shift 3.0; classifier-free guidance (Ho & Salimans, 2022) uses $\text{cfg_text_scale} = 3.0$, $\text{cfg_img_scale} = 1.5$, applied over the interval $[0.4, 1.0]$ of the schedule, with global CFG renormalization and $\text{cfg_renorm_min} = 0.0$. Images are decoded by the Bagel VAE at a fixed 512×512 resolution; condition images are passed through the SigLIP encoder (Zhai et al., 2023) at native resolution clamped to the multi-scale range $[224,\ 518]$ with stride 14. We do not use TaylorSeer caching at inference. Each n-th trial uses seed $\text{idx} \cdot 10^4 + n$ to keep diversity across trials reproducible.

Table A.2: Inference configuration used for Sperry on SperryBench and the CoT helpfulness study. The same setting is applied to all subsets and to all $n = 8$ trials per question.

Component	Value
Sampling protocol	
trials per question n	8
seed for trial i on sample idx	$\text{idx} \cdot 10^4 + i$
Text decoding	
strategy	autoregressive sampling
temperature	0.5
top-p	1.0 (no nucleus filtering)
top-k	disabled
max new tokens	2,000 per text segment
Image generation (rectified-flow ODE)	
denoising steps	50
timestep shift	3.0
cfg_text_scale	3.0
cfg_img_scale	1.5
cfg_interval	[0.4, 1.0]
cfg_renorm_type	global
cfg_renorm_min	0.0
output resolution	512×512
TaylorSeer caching	disabled
Image conditioning	
VAE input size (problem / generated context)	512×512, stride 16
SigLIP ViT input size	[224, 518] multi-scale, stride 14
System	
precision	bf16
hardware	1×H100 80 GB per inference process

Baseline decoding. All open-source baselines are sampled with temperature 0.7 and at most 8,192 new tokens in non-thinking mode and 32,768 in thinking mode; GPT-5.4 is queried through the official OpenAI API at `medium` and `high` reasoning effort. The inference profile in Sec. 5.3 compares Sperry under the configuration of Tab. A.2 (2,000 tokens per text segment) with Qwen3.5-9B under its 32,768-token thinking budget served with vLLM.

A.3 Reward Function

Design. The GSPPO reward is computed by code from the generated image alone; no text is decoded and no learned reward model is used. It has three properties. First, *reference-grounded*: every candidate image is compared with the ground-truth intermediate image for the same step of the trace. Second, *rule-based and cheap*: each rule is a few lines of NumPy (Harris et al., 2020) and OpenCV that run at every rollout. Third, *shape-matched*: the five subsets whose intermediate state is a discrete structure (a path, a highlighted layer, a traced line, a projected view, a set of objects) receive a binary reward, while Jigsaw, whose state is a filled photograph, receives continuous per-region credit. The rules are deliberately coarse proxies for state correctness: each checks the task-relevant structure and ignores the rest of the scene, and none inspects the regenerated text prompt.

Inputs and alignment. Each RL sample is one image-generation step of a training trace, i.e. a sub-problem consisting of the problem image(s) P, the interleaved prefix up to that step, and the ground-truth intermediate image $I^\star$ for that step, which was rendered by the simulator (or, for Counting and Jigsaw, comes from Zebra-CoT (Li et al., 2025a)) when the trace was built. GSPPO regenerates the text segment that requests the image with the frozen understanding branch (Sec. 4.4), samples G candidate images $\hat{I}$, and scores each against the $I^\star$ stored with the sub-problem; alignment to the reference is by construction of the sub-problem, not by parsing the generated text. All images are resized to a common resolution before comparison, and each rule returns $r \in [0, 1]$.

Maze: cell-level path match (binary). Grid lines are detected in P as image rows and columns in which more than 30% of pixels are near-black (all channels < 50), clustered with a 5-pixel gap.

Blue path pixels ($B > 200$, $B > R + 50$, $B > G + 50$) are extracted from $I^\star$ and $\hat{I}$, pixels that are already blue in P are removed, and each mask is mapped to the set of grid cells it touches. $r = 1$ iff the two cell sets are identical.

BlockCount and LineTrack: highlight IoU (binary). The task-relevant highlight (the red top layer in BlockCount, the red traced segment in LineTrack) is isolated as the pixels that differ from P (channel-sum difference > 30) and are red ($R > 180$, $G < 120$, $B < 120$). $r = 1$ iff the intersection-over-union of the highlight masks of $I^\star$ and $\hat{I}$ is at least 0.5; $r = 0$ if the reference mask is empty.

CubeView: silhouette IoU (binary). Both images are resized to 128×128 and thresholded to a foreground silhouette (grayscale < 220). $r = 1$ iff the silhouette IoU is at least 0.5.

Counting: color-cluster match (binary). Both images are resized to 512×384 and converted to HSV. Eight color ranges (red, orange, yellow, green, cyan, blue, purple, and dark) are segmented, cleaned by morphological closing and opening (5×5 ellipse, two iterations each), and their contours with area of at least 300 pixels are counted, giving a sorted multiset of colored-object labels. $r = 1$ iff the multisets of $I^\star$ and $\hat{I}$ are identical.

Jigsaw: per-region fill similarity (continuous). The fill region is the set of pixels where $I^\star$ differs from the source image P (channel-sum difference > 30), split into connected components of at least 50 pixels. Each component c receives $1 - \min(\mathrm{MSE}_c/\rho,\, 1)$ with $\rho = 5000$, and r is the mean over components (Alg. 1). A binary variant (whole-region $\mathrm{MSE} \le 100$) exists in the code but is not used for GSPPO.

Reward scale in Tab. 8. The "image reward" reported there is the fraction of benchmark traces in which every generated intermediate image passes its step's check (the binary Jigsaw variant is used for this purpose), multiplied by 100. Counting demands an exact match of the detected object multiset and Jigsaw compares natural-photo tiles pixelwise, so their absolute values are low for every checkpoint; the comparison between schedules is still meaningful because both are scored by the same rule.

Algorithm 1 Continuous per-region reward (Jigsaw, GSPPO).

function JIGSAWREGIONREWARD($\hat{I}$, $I^\star$, P, ρ)
  Input: generated image $\hat{I}$, ground-truth filled image $I^\star$, source image P; clip constant ρ=5000.
  $M \leftarrow \mathbb{1}\big[\sum_{\mathrm{ch}} |I^\star - P| > 30\big]$ ▷ fill mask
  $\{c_1, \ldots, c_n\} \leftarrow$ connected components of M with $|c_j| \ge 50$
  if $n = 0$ **then return** 1
  for $j \leftarrow 1$ **to** n **do**
    $\mathrm{MSE}_j \leftarrow \frac{1}{|c_j|} \sum_{p \in c_j} \|\hat{I}_p - I^\star_p\|_2^2$
    $r_j \leftarrow 1 - \min(\mathrm{MSE}_j/\rho,\, 1)$
  end for
  return $\frac{1}{n} \sum_{j=1}^{n} r_j$
end function

B.1 ANSWER EXTRACTION AND GRADING

All SperryBench accuracies, for Sperry and for every baseline, are computed by the same grader: a rule-based matcher applied to the model's boxed answer, with a small language model used only to extract an answer when no box is present. No partial credit is given.

Extraction. We take the inner content of the last balanced `\boxed{...}` span in the model's output. If no such span exists, Qwen3.5-4B is prompted with the question and the model's full output and asked to extract the final answer in the subset's answer format; the extracted string then enters the same matching rules below. We write $\textsc{Extract}(s)$ for this two-step procedure (boxed span if present, otherwise the Qwen3.5-4B extraction); it returns $\perp$ only when neither yields an answer. Grading therefore does not depend on a model's willingness to emit a box; low Maze scores for several baselines reflect wrong or incomplete move sequences rather than formatting.

Matching. Counting, BlockCount, and LineTrack answers are integers or short canonical strings and are compared with the ground truth exactly after stripping whitespace (Alg. 2). Maze answers are

move sequences such as `DDDLLUL` and are compared after stripping whitespace and uppercasing (Alg. 3); a shorter prefix or a sequence that exits at the wrong cell is incorrect. CubeView and Jigsaw answers are option labels and are canonicalized before comparison: `A`, `a`, `Option1`, and `1` all denote the first option, with the analogous mapping for options 2–4 (Alg. 4).

Aggregation. Per-subset accuracy is the fraction of correctly graded attempts, averaged over the eight decoding attempts per question (Sec. A.2); Overall is the unweighted mean over the six subsets (Sec. 5.1). The Random Guess row of Tab. 2 is the expected accuracy of uniform guessing over each subset's answer space under the same grader.

Algorithm 2 Exact-match grading (Counting, BlockCount, LineTrack).

function EXACTGRADE(s, g)
  $\hat{a} \leftarrow$ EXTRACT(s); **if** $\hat{a} = \bot$ **then return** 0
  return $\mathbb{1}[\,\mathrm{strip}(\hat{a}) \equiv \mathrm{strip}(g)\,]$
end function

Algorithm 3 Sequence-equality grading (Maze).

function MAZEGRADE(s, g)
  $\hat{a} \leftarrow$ EXTRACT(s); **if** $\hat{a} = \bot$ **then return** 0
  return $\mathbb{1}[\,\mathrm{upper}(\mathrm{strip}(\hat{a})) \equiv \mathrm{upper}(\mathrm{strip}(g))\,]$
end function

Algorithm 4 Option-letter grading (CubeView, Jigsaw).

function OPTIONGRADE(s, g)
  $\hat{a} \leftarrow$ EXTRACT(s); **if** $\hat{a} = \bot$ **then return** 0
  $p \leftarrow \mathrm{canon}(\mathrm{upper}(\mathrm{strip}(\hat{a})))$; $q \leftarrow \mathrm{canon}(\mathrm{upper}(\mathrm{strip}(g)))$ ▷ $\{\texttt{OPTION}n, n\} \mapsto \texttt{A–D}$
  return $\mathbb{1}[\,p \equiv q\,]$
end function

B.2 NON-THINKING RESULTS

The main paper reports thinking-enabled results, where every reasoning-capable baseline is run with its native chain-of-thought mode on. To rule out the possibility that those numbers reflect prompting overhead rather than visual reasoning, we additionally evaluate every Qwen3-VL, Qwen3.5, and Gemma-4 baseline in its default (non-thinking) instruct mode, with the same prompt template as in the main results and only the thinking-mode flag flipped. Decoding hyperparameters (apart from the maximum generation length, Sec. A.2) and the SperryBench grading protocol are identical to the thinking runs. Tab. A.3 reports the per-subset accuracies. Two patterns are visible. First, without thinking, scale no longer helps monotonically: Qwen3.5-27B (32.8) matches or exceeds the 35B (32.0) and 397B (32.4) variants; under non-thinking decoding, the extra capacity of a much larger reasoning model collapses toward its instruct-mode capacity. Second, the relative ordering across subsets matches the thinking-enabled setting: the simulation-heavy subsets (Maze, BlockCount, CubeView, LineTrack) remain the bottleneck, while Counting and Jigsaw remain tractable. Both observations indicate that the gap to Sperry is not explained by absent test-time chain-of-thought alone.

Table A.3: Non-thinking results on SperryBench. Qwen3-VL-2B and Gemma-4-26B were not evaluated in this mode. All reasoning-capable models are evaluated in their default (instruct) mode without thinking. This complements Table 2 which reports thinking-enabled results. **Bold** / underline mark best / second-best per column.

Model	Size	Maze	BlockCount	CubeView	LineTrack	Counting	Jigsaw	Overall
Open-source Models								
Qwen3-VL	8B	0.0	5.7	3.9	14.5	80.6	38.4	23.9
	32B	0.8	10.1	3.0	16.5	81.1	42.5	25.7
	235B	2.1	7.8	11.6	17.8	84.6	45.9	28.3
Qwen3.5	9B	4.6	7.8	15.0	20.0	**85.7**	50.1	30.5
	27B	4.6	6.4	23.6	**25.3**	**85.7**	51.3	32.8
	35B	5.4	7.1	15.7	23.0	85.1	**55.5**	32.0
	397B	5.4	**12.2**	17.2	22.8	84.6	52.1	32.4
Gemma-4	E2B	0.0	6.4	14.5	13.8	42.3	24.1	16.9
	E4B	0.4	6.1	25.4	13.3	58.3	32.4	22.7
	31B	**7.5**	6.8	**34.5**	20.7	78.3	55.1	**33.8**

B.3 Chain-of-Thought Helpfulness Evaluation

Pairwise design. Likert-style scoring (e.g., asking the judge for an integer in $[1, 5]$) suffers from severe scale collapse on long reasoning traces, with the judge tending to assign the same midrange score regardless of model. We therefore use a pairwise win-rate design: the judge sees two reasoning traces side-by-side on the same problem and picks the more helpful one. To mitigate position bias (Zheng et al., 2023), the A/B presentation order is randomized per call. To bound the budget, we sample 3 of the $\binom{4}{2} = 6$ possible model pairs per question, with the sample and order seeded deterministically by question id.

Judge. We use GPT-5.4 with `high` reasoning effort. For each pair, the judge sees (i) the problem image, (ii) Trace A, and (iii) Trace B. Each trace is presented in the format the model actually produces, with interleaved text and intermediate generated images for Sperry and text-only output for the Qwen3.5 baselines. The judge prompt is shown in Prompt A.1.

Prompt: Pairwise CoT-helpfulness judge (GPT-5.4, high reasoning)

```
You are a judge.  Below is a problem (with image) and two model
reasoning traces (Trace A and Trace B). Each trace may consist of
text, intermediate images the model generated during reasoning, or
both.
Decide which trace is MORE HELPFUL for solving the problem
(helpfulness only, not stylistic preference).
Output ONLY one of these tokens, with no other text:
A - Trace A is more helpful.
B - Trace B is more helpful.
Tie - Both are roughly equally helpful (or equally bad).
```

Subsets and aggregation. We restrict the evaluation to the four hand-designed SperryBench subsets (Maze, BlockCount, CubeView, LineTrack), where rule-based ground-truth solution traces exist by construction, so "helpfulness" is well defined and judgments can be audited against a reference (the judge itself sees only the problem and the two traces, Prompt A.1); the two filtered open-source subsets (Counting, Jigsaw) lack such references and are omitted. Across 1,577 questions (240 Maze, 296 BlockCount, 441 CubeView, 600 LineTrack), we collect 4,731 pairwise judgments. The win rate of model m over model m' is $(w + 0.5\,t)/n$, where w counts wins, t ties, and n total games between them. The overall win rate of m is the same quantity averaged over all opponents (game-weighted).

Per-subset breakdown. The headline pairwise matrix and overall win rates are shown in Fig. 7 and Fig. 8 of the main text. Tab. A.4 additionally reports each model's average win rate on each of the

Table A.4: GPT-5.4 pairwise helpfulness win rates on the four hand-designed SperryBench subsets, broken down by subset. Per-subset cells average over the row model's three opponents; `Overall` is the game-weighted win rate across all opponents and subsets, matching the bar chart in Fig. 8.

Model	Maze	BlockCount	CubeView	LineTrack	Overall
Qwen3.5-9B	0.416	0.441	0.392	0.476	0.437
Qwen3.5-35B	0.396	0.478	0.437	0.493	0.460
Qwen3.5-397B	0.552	0.494	0.501	**0.535**	0.520
Sperry (Ours)	**0.637**	**0.586**	**0.671**	0.496	**0.583**

four hand-designed subsets, averaged over its three opponents; Sperry wins three of four subsets and trails Qwen3.5-397B on LineTrack (0.496 vs. 0.535), which is dominated by short textual reasoning.

C.1 SPERRY-43K IMAGE SIMULATOR

For each of the four hand-designed subsets we implement a deterministic Python simulator that draws every image in the trace, namely the problem image Q_{I_1} and every intermediate state $R_{i,k}$, with fixed rendering parameters per subset, so the visual style of an example is decided entirely by its random seed. The simulator-rendered intermediate images stored in each trace serve as the reference images for the GSPPO reward (Sec. A.3), and the simulator's answers serve as ground truth for the correctness check (Sec. C.4). The simulators are summarized in Tab. A.5; details follow.

Table A.5: Image simulators for the four hand-designed subsets of Sperry-43K. "Trace length" counts intermediate images per sample (excluding the problem image).

Subset	Renderer	Randomized factors	Fixed view / palette	Trace length	Answer space
Maze	maze-dataset + matplotlib	grid $5-7$, path length $4-10$, render style {pixel, plot}	S=green dot, E=red star, path=blue	1	UDLR string
BlockCount	matplotlib 3D voxel	grid $4-7$, max height 4, fill $0.35-0.85$	azim 225°, elev 40°, gray cubes, red top-layer	H $(1-4)$	integer count
CubeView	matplotlib 3D voxel	grid $\{3,4,5\}$, $4-15$ cubes, target view $\in$ {6 axis-aligned}	purple-palette faces, red arrow	3 rotations	A/B/C/D
LineTrack	PIL + Catmull–Rom curves	$4-7$ lines, ≥ 2 random crossings, query letter	dark-gray lines, red trace, 640×480 canvas	#cross+1 $(3-6)$	digit $1-7$

Maze. We sample lattice mazes with the `maze-dataset` library (Ivanitskiy et al., 2023) (`LatticeMazeGenerators.gen_dfs`) at three difficulty bands (grid 5×5, 6×6, 7×7 with path lengths $4-8$, $5-9$, $6-10$, respectively, sampled with equal probability). Each sample is rendered in one of two styles, picked uniformly: a pixel-art style that upscales the maze-dataset bitmap by $32\times$ and overlays `S`/`E` markers via matplotlib, and a plot style that uses `MazePlot` at dpi=80 with a 40 px unit length. Row indices (0 at the top) and column indices (0 at the left) are drawn as axis labels in both styles. The trace contains a single intermediate image: the same maze with the unique solution path overlaid in blue. The ground-truth answer is the UDLR move sequence emitted by the maze-dataset solver.

BlockCount. We sample a height map $h \in \mathbb{Z}^{N\times N}$ with $N \in \{4,5,6,7\}$, columns of height $\in \{1,\ldots,4\}$, and a fill ratio in $[0.35, 0.85]$. To guarantee that every occupied column's top face is visible from the fixed camera (`azim`$=225^\circ$, `elev`$=40^\circ$), we reject any height map that violates either of the two visibility constraints $h[c-1,r-1] \leq h[c,r]+1$ (diagonal blocker) and $\min(h[c-1,r],\, h[c,r-1]) \leq h[c,r]$ (dual side-blockers). The problem image renders the full structure with default cube color `#B0B8C1` and edge color `#333333`; intermediate images render the structure with the topmost remaining layer recolored to `#E05252`, then with that layer removed, recursively, until only the base layer remains. The trace length is therefore H where H is the maximum column height (so $1-4$ images), the last image highlighting the base layer. The ground-truth answer is the total cube count, $\sum_{r,c} h[r,c]$.

CubeView. We sample a connected voxel structure on a grid in $\{3,4,5\}^3$ by frontier-expanded random growth from a seed at the bottom center, with cube counts in $\{(4,8),(6,12),(8,15)\}$ depending on the grid size; we deduplicate by translation-canonical form and reject structures that span fewer than two units along ≥ 2 axes. Faces are painted with a fixed purple palette ($\#C5B8DB$ top, $\#9B89B9$ front, $\#A094C0$ right, $\#7B6B9E$ back, $\#8578A8$ left, $\#6B5B8E$ bottom); the

problem image is a perspective view with a red arrow indicating the target observation direction (one of {front, back, left, right, top, bottom}, sampled with weights $\{2200, 2200, 2200, 2200, 2200, 1000\}$). The four answer options are drawn as 224×224 2D grids of filled cells in medium purple ($\#9B89B9$); the correct option is the orthographic projection of the structure along the queried direction, distractors are random structurally distinct projections. Intermediate images interpolate the camera linearly from the problem-image perspective to the target view in three equal angular steps (rounded to the nearest 5° in the rewrite, Sec. C.3). The ground-truth answer is the option letter $\in \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$.

LineTrack. We draw $N \in [4, 7]$ same-color tangled curves on a 640×480 canvas, with endpoints distributed on a left vertical line of letters and a right vertical line of digits. Each curve is a Catmull–Rom spline through two random interior waypoints constrained to a horizontal band $[200, 440] \times [60, 420]$ with vertical jitter 160 px, sampled at 400 points per curve. We reject scenes whose query line has fewer than two crossings with other lines. The problem image renders all lines in dark gray ($\#444444$); intermediate images recolor the query line in red ($\#E05252$) from its starting letter up to the k-th crossing, with all non-query lines faded to $\#CCCCCC$ in CoT images. The trace contains $C+1$ images for C crossings (so $3-6$ images), the last showing the full red trace. The ground-truth answer is the digit at the right endpoint of the query line.

C.2 SPERRY-43K FIXED TEXT POOL

Each of the four simulators emits a sample as a tuple $(Q, T_0, R_{1:K})$, where Q is the question, T_0 is a placeholder for the model's first reasoning text segment, and $R_{1:K}$ are the simulator-rendered intermediate images. To produce a full interleaved trace we also need, for each image R_k, a text segment T_k that connects it to R_{k+1} (or, for the last image, to the boxed answer); in the notation of Eq. 2, T_k is the text segment $R_{t,k+1}$ that requests image $k+1$, and $T_0 = R_{t,1}$. Rather than inventing these segments at training time, we maintain two hand-written pools per subset and sample from them deterministically by trace seed.

Question pool. Each subset has a small set of paraphrased question templates (4 for Maze; analogous for BlockCount, CubeView, LineTrack). The templates carry identical semantics, in that they describe the input image, the task, and the answer format, but vary in surface form ("Find a path from S to E" vs. "Provide your answer as a UDLR string"). The aim is to prevent the SFT model from latching onto a single phrasing as a task identifier. The full templates are released with the dataset.

Reasoning-announcement pool. Each subset additionally carries a pool of "announcement" sentences for T_0, used to introduce the upcoming intermediate image. For Maze the pool has 20 paraphrases of "I will first generate a maze image with the path from the start to the end marked, and then reason from that path image to give the final answer". Analogous pools exist for the other three subsets. These are also paraphrased only; they never reveal the answer or the simulator's intermediate state.

Per-step caption stubs. For each T_k with $k \geq 1$ the simulator emits a short mechanical caption (e.g. for BlockCount: "Now I remove the highlighted layer and look at the next one"; for CubeView: "Rotate the camera by 30°"). These stubs are deliberately templated and do not yet read like first-person reasoning; turning them into natural prose is the job of Sec. C.3.

Why fixed pools instead of generation. The pools provide three properties that LLM-generated text cannot: (i) they are guaranteed to be answer-leak-free by construction (a human curator wrote each line), (ii) they are cheap, with no API cost when scaling the pool by an order of magnitude, and (iii) they are reproducible; rerunning the simulator on the same seed produces byte-identical traces. The diversity that the LLM rewrite adds in Sec. C.3 sits on top of, not instead of, the pools.

C.3 LLM REWRITE DETAILS

This section and the next describe Stages 2B and 3 of the pipeline in Fig. 2: the caption rewrite (here) and the correctness verification (Sec. C.4). The filters F1–F5 of Sec. C.5 are applied around these two stages.

Goal. The simulator emits captions like "Rotate 90°." or "Now look at layer 3.", which are correct but mechanical. Models trained directly on these captions inherit the cadence and learn "caption then

image" rather than "think then image". Stage 1 of the rewrite pipeline therefore replaces every T_k with a paraphrase produced by a strong LLM, whose prompt includes the simulator-rendered images so that the rewrite is grounded in the actual visual state.

Model and infrastructure. We use Qwen3.5-27B served locally via vLLM, with 16 concurrent requests, temperature 0.6, top-p 0.95, thinking enabled (the LLM's own ⟨think⟩ block is stripped from its output). Each subset has its own system prompt and per-step output template; we resume on failure by skipping ids already present in the output JSONL.

Per-subset prompts. The system prompts (selected excerpts in Prompt A.2 for BlockCount and Prompt A.3 for LineTrack) all share three rules and differ only in subset-specific style guidance:

1. *Strict no-leak:* the rewrite must not state any number, count, option letter, move, or partial result that would let a downstream learner shortcut the visual reasoning.
2. *Open-with-think:* the first segment T_0 opens with `<think>` but does not close it; the closing ⟨/think⟩ comes after the final image and just before $\boxed{\text{answer}}$.
3. *No conclusion:* no rewrite produces a sentence that summarizes or commits to an answer; the answer lives only in the final $\boxed{\text{answer}}$.

Prompt: Stage-1 rewrite system prompt for BlockCount (excerpt)

```
You are rewriting the text labels in a visual reasoning
chain-of-thought for a 3D cube-counting task.
The model being trained can generate images as intermediate reasoning
steps.  It counts all unit cubes in a 3D structure by removing the
top layer one at a time.  In each CoT image, the current top layer is
highlighted in RED. After that layer is examined, it is removed and
the next image highlights the new top layer.
Desired style:
- TEXT 1:  observe the structure and its height.  Announce the plan to
peel it layer by layer from the top.  Do NOT state any numbers.
- TEXT 2 ...  TEXT (N-1):  acknowledge the previous highlighted layer,
then describe removing it and highlighting the next one down.  Do NOT
state any cube counts or running totals.
- TEXT N (last, before the base-layer image):  acknowledge the
previous layer, remove it, and note that only the base layer remains -
now highlighted in red.  Do NOT state any counts or the final answer.
Rules:
- Do not write <think> or </think> anywhere
- Do NOT state any numbers, counts, or totals anywhere
- Do not write a conclusion or final answer - the chain continues
after the last image
- Keep each piece concise (2-3 sentences max)
```

Boxed-answer rewrite for Counting and Jigsaw. The two open-source subsets do not pass through the per-step rewrite (their reasoning text already exists), but their final segments rarely terminate in $\boxed{\text{answer}}$ form. We run a separate rewrite (Qwen3.5-27B, temperature 0.3, thinking disabled, concurrency 32) that takes each sample's final paragraph and the ground-truth answer, and rewrites only the ending so it concludes naturally with $\boxed{\text{answer}}$. The system prompt forbids changing earlier reasoning and constrains the inside of $\boxed{\text{answer}}$ to a number or short mathematical expression.

Output parsing and retries. Each rewrite call asks the LLM to emit `TEXT 1: <rewritten>`, `TEXT 2: <rewritten>`, ... in a fixed format; we parse this with a regular expression and mark ($parse_ok = \text{False}$) on any malformed output. Failed parses are retried up to three times with a slightly different system prompt and otherwise dropped. The rewrite output is stored as a JSONL keyed by sample id so that downstream stages can join on it.

Prompt: Stage-1 rewrite system prompt for LineTrack (excerpt)

```
You are rewriting the text labels in a visual reasoning
chain-of-thought for a line-tracing task.
The model being trained sees N same-color tangled lines connecting
letters on the left to numbers on the right.  It must trace the
queried line step by step through crossing points to find which number
it connects to.  Each CoT image shows the query line highlighted in
red from its starting letter up to the current point.
Desired style:
- TEXT 1:  observe the tangled lines and announce the plan to trace
letter X step by step.  Do NOT mention how many crossings there are.
- TEXT 2 ...  TEXT (N-1):  you have just seen the previous image.
Acknowledge that the red trace has passed through a crossing point,
then naturally lead into continuing the trace.  Do NOT state which
crossing number this is or how many remain.
- TEXT N (last):  acknowledge the trace is almost complete and lead
into seeing which endpoint it reaches.  Do NOT state the answer number
- the image reveals it.
Rules:
- Do NOT reveal the answer number in any text piece
- Do NOT mention the total number of crossings
- Keep each piece concise (2-3 sentences max)
```

C.4 Correctness Verification Details

The rewrite step preserves the simulator's intermediate images verbatim but replaces the surrounding text. We therefore need an end-to-end check that the rewritten trace, taken as a whole, is still solvable: a strong external model, given exactly the multi-turn conversation our SFT student will see, must still recover the simulator's ground-truth answer. Stage 2 of the pipeline performs this check.

Verifier model. We use Gemini 3.1 Pro (`gemini-3.1-pro`) at temperature 0.6 and top-p 0.95 (the settings of the BlockCount and LineTrack runs) and with $\mathrm{max_tokens}{=}32768$ (the setting of the Maze and CubeView runs), using 8 concurrent requests. We run the verifier without its extended-thinking mode, so it continues the conversation only briefly before committing to a boxed answer: the test is whether the trace is solvable when the verifier is forced to commit to an answer based only on the visual chain we constructed, not whether the verifier can recover the answer with its own private chain of thought.

Multi-turn replay. For each sample we replay the trace as a conversation: the system message instructs the verifier that "each subsequent user turn provides the image you requested; continue your reasoning from where you left off"; the first user turn carries the problem image Q_{I_1} and the question Q_T; each assistant turn carries the corresponding rewritten text segment T_k from Stage 1; each subsequent user turn carries the next simulator image R_k; after the last image the verifier is asked to continue and emit the final $\boxed{\text{answer}}$.

Pass criterion. We extract $\boxed{\text{answer}}$ from the verifier's continuation and compare it to the simulator's ground-truth answer using task-specific answer normalization: exact match for BlockCount/Counting/LineTrack, case-insensitive sequence match for Maze, option-letter equivalence for CubeView/Jigsaw. Samples whose verifier prediction does not match are dropped; samples with no $\boxed{\text{answer}}$ are also dropped. The procedure is given in Alg. 5.

C.5 Data Filtering Pipeline

The end-to-end pipeline (Fig. A.1) feeds raw simulator output into the final Sperry-43K dataset through five filters. Stages 1 and 2 are the rewrite and correctness check of Secs. C.3–C.4; the remaining three are deduplication, helpfulness filtering for the open-source subsets, and the train/bench stratified split.

Algorithm 5 Correctness-ensure check (Stage 2).

Require: sample $(Q, T_{0:K-1}, R_{1:K}, g)$ with verifier V
msgs $\leftarrow$ [system; (user, Q_{I_1}, Q_T)]
for $k \leftarrow 0$ **to** $K-1$ **do**
  append (assistant, T_k) to msgs
  append (user, R_{k+1}) to msgs
end for
$y \leftarrow V(\text{msgs})$
$\hat{a} \leftarrow$ the last balanced boxed answer in y, or $\perp$ if absent
if $\hat{a} = \perp$ **then return** `drop`
if $\text{MATCH}_{\text{subset}}(\hat{a}, g)$ **then return** `keep` **else return** `drop`

Raw simulator output $\rightarrow$ **(F1)** canonical-form dedup $\rightarrow$ **(F2)** Stage-1 LLM rewrite
$\rightarrow$ **(F3)** Stage-2 correctness check $\rightarrow$ **(F4)** helpfulness filter (Counting/Jigsaw only)
$\rightarrow$ **(F5)** stratified train/bench split $\rightarrow$ *Sperry-43K* + *SperryBench*

Figure A.1: End-to-end Sperry-43K filtering pipeline. Filters F2 and F3 are described in Secs. C.3 and C.4; F1, F4, F5 are described below.

F1: structural deduplication. For BlockCount and CubeView the simulator can in principle emit duplicate height maps or voxel structures up to translation. We therefore canonicalize each emission and reject duplicates within the pool. For BlockCount, the canonical form is the sorted tuple of (c, r, h) triples after subtracting the bounding-box minimum; for CubeView, it is the analogous 3D-translation-canonical voxel set. Maze and LineTrack draw from much larger structural spaces (random lattice mazes; Catmull–Rom curves with continuous control points), so exact duplicates are rare; we deduplicate them by sample seed only.

F4: helpfulness filtering for Counting and Jigsaw. The two subsets drawn from Zebra-CoT (Li et al., 2025a) do not have a simulator-derived correctness check; their intermediate images come from the source dataset and may be irrelevant or even misleading. We therefore filter them with a per-image helpfulness rubric. Each intermediate image is judged independently by Qwen3-VL-235B-A22B-Instruct-FP8 (prompt in Prompt A.4), which receives the original problem image, the question, the model's final answer, and the candidate intermediate image, and emits a $1-5$ Likert score plus a binary helpful flag with a one-sentence reason. We retain a sample only if every intermediate image scores ≥ 3; the flag is used as a tie-breaker for borderline cases. The resulting Counting pool keeps $3{,}490$ of the original $\sim$5,300 candidates and the Jigsaw pool keeps $9{,}932$ of $\sim$13,400.

F5: stratified train/bench split. For each of the six subsets we reserve $\approx 5\%$ of the surviving samples for the SperryBench visual reasoning benchmark; the split is stratified by sample-level difficulty proxies (path length for Maze, max height for BlockCount, total angular distance for CubeView, number of crossings for LineTrack, image count for Counting, piece count for Jigsaw) so that the benchmark distribution matches the training distribution. The split is computed once with a fixed seed and never revisited; SperryBench samples are excluded from every training stage (SFT, GSPPO) and from the Monet-baseline retraining (Sec. C.7). The resulting split sizes are listed in Tab. 1 of the main paper: $42{,}682$ training samples and $2{,}249$ evaluation samples, summing to the headline $44{,}931$.

Reproducibility. The simulator stages are deterministic given the per-subset random seeds. The LLM stages (Qwen3.5-27B rewriting for F2, Gemini 3.1 Pro verification for F3, Qwen3-VL-235B helpfulness scoring for F4) are sampled and API-dependent, so we release their outputs as caches rather than claiming bitwise reproducibility. We release the simulator code, the rewrite/verify scripts, and the Stage-1/Stage-2 JSONL caches together with Sperry-43K, so reruns can either (a) regenerate from scratch with new seeds, or (b) reproduce the released split exactly.

C.6 Data Samples

To give the reader a concrete sense of what Sperry learns from at SFT time, Figs. A.2–A.7 show five or six training traces from each of the six Sperry-43K subsets, one full page per subset. Each row is one sample: the leftmost cell is the problem image plus the question (truncated for space), the middle cells are the simulator-rendered intermediate images $R_{1:K}$ paired with the corresponding rewritten reasoning text segments $T_{1:K}$ (Sec. C.3), and the rightmost red chip is the simulator's ground-truth

Prompt: Per-image helpfulness rubric (Qwen3-VL-235B, F4)

```
You are an expert evaluator specialising in multimodal AI
training-data quality.
You will be given:
1.  The original image(s) provided to an AI model, along with the
question asked.
2.  The AI model's final answer to that question.
3.  One intermediate visual reasoning image the model generated while
solving the question.
Your sole task is to judge whether the intermediate image genuinely
helped the model arrive at the correct answer.
Required output (exactly three lines):
Helpful:  True/False
Score:  <integer 1-5>
Reason:  <one concise sentence>
Scoring rubric:
1 - Irrelevant or misleading; does not help at all.
2 - Marginally related; provides little practical value.
3 - Somewhat helpful; offers useful but non-essential visual context.
4 - Clearly helpful; highlights or reveals information important for
the answer.
5 - Essential; it would be very hard to reach the correct answer
without this image.
```

answer that the trace eventually emits inside $\boxed{\text{answer}}$. We picked the displayed samples by difficulty within each subset (trace length where it varies, answer-path length for Maze, number of crossings for LineTrack): the rows are among the hardest-looking traces in the training pool, so the figures are not cherry-picked for visual clarity, but instead represent the upper end of difficulty seen during training.

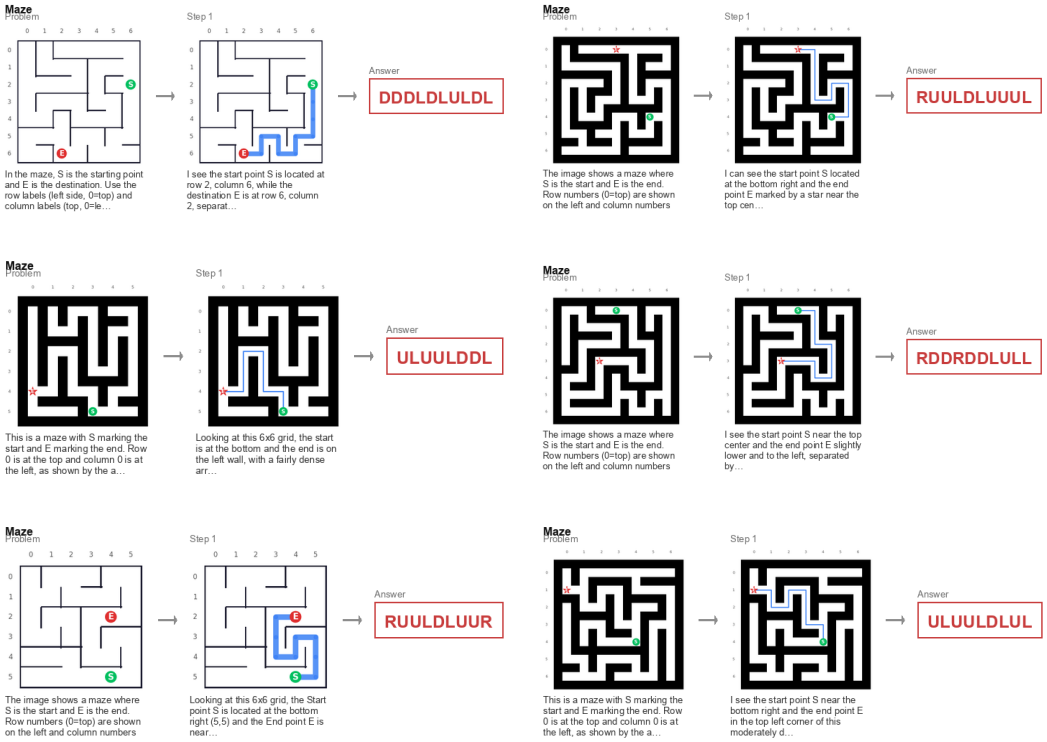

Figure A.2: Six training samples from the Maze subset, ranked by answer-path length (longest first). Trace length is always $K{=}1$: a single intermediate image with the solution path overlaid in blue. The model emits the UDLR move sequence after observing the path image.

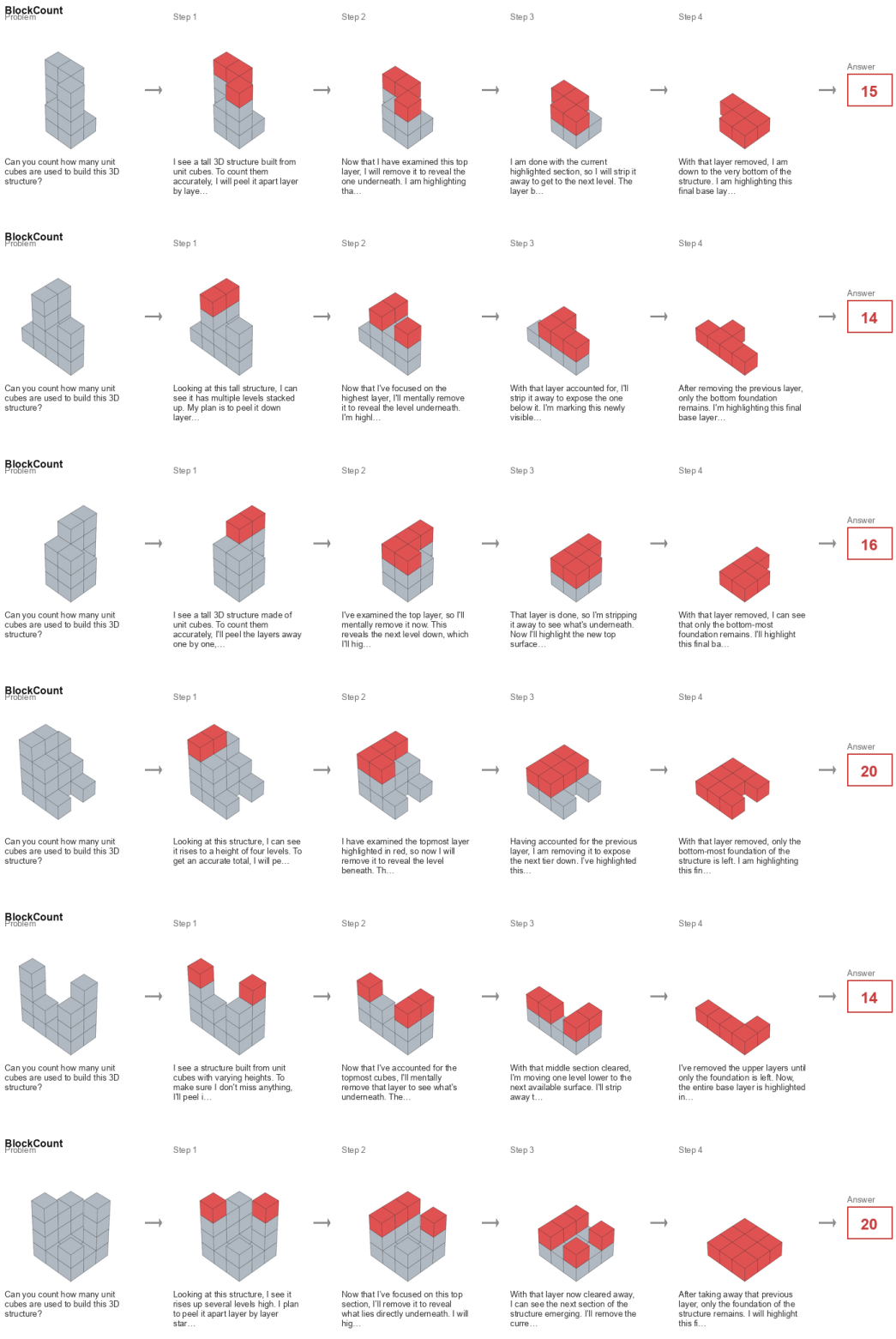

Figure A.3: Six training samples from the BlockCount subset, ranked by trace length (deepest structures first). The intermediate images peel the structure from the top: at each step the current top layer is highlighted in red, then removed in the next step. Trace length equals H where H is the maximum column height: each layer, including the base, is highlighted in turn.

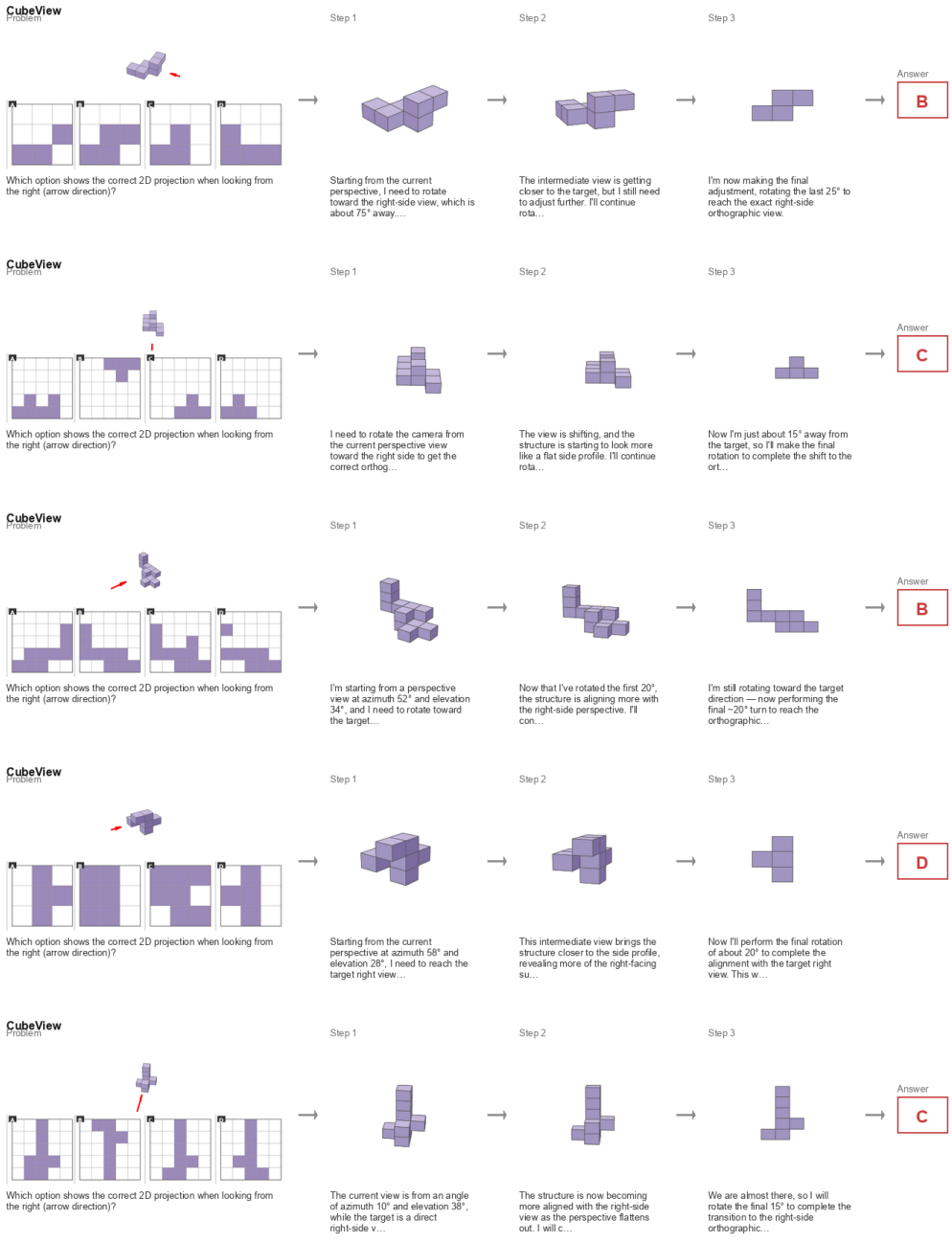

Figure A.4: Five training samples from the CubeView subset. The problem image places the perspective view with the red target-direction arrow above the four answer options (bottom). The three intermediate images interpolate the camera linearly toward the target view, in equal angular steps. The final answer is one of `A`/`B`/`C`/`D`.

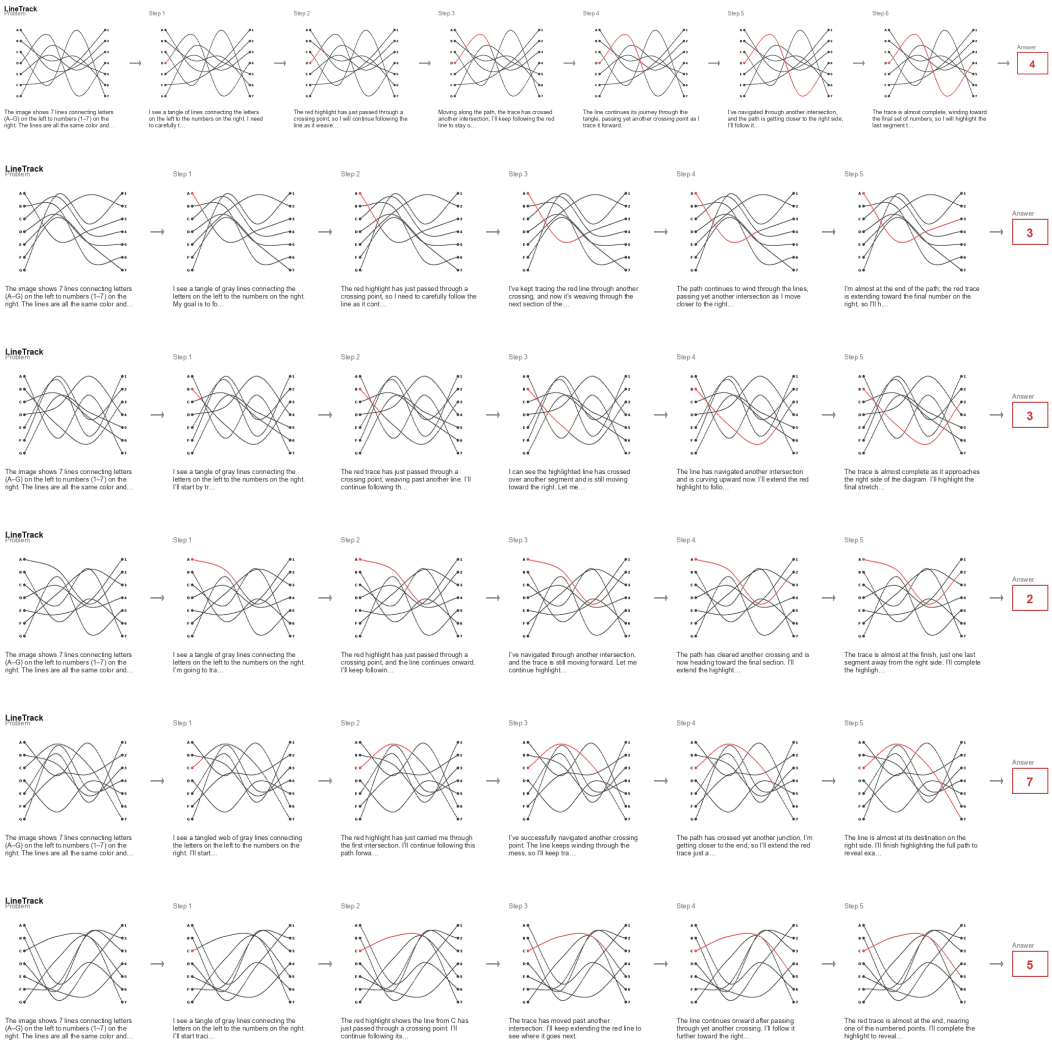

Figure A.5: Six training samples from the LineTrack subset, ranked by number of crossings on the query line (most crossings first). The intermediate images progressively trace the queried line in red, one crossing at a time, until the right-hand endpoint is reached. Trace length equals $C+1$ where C is the number of crossings.

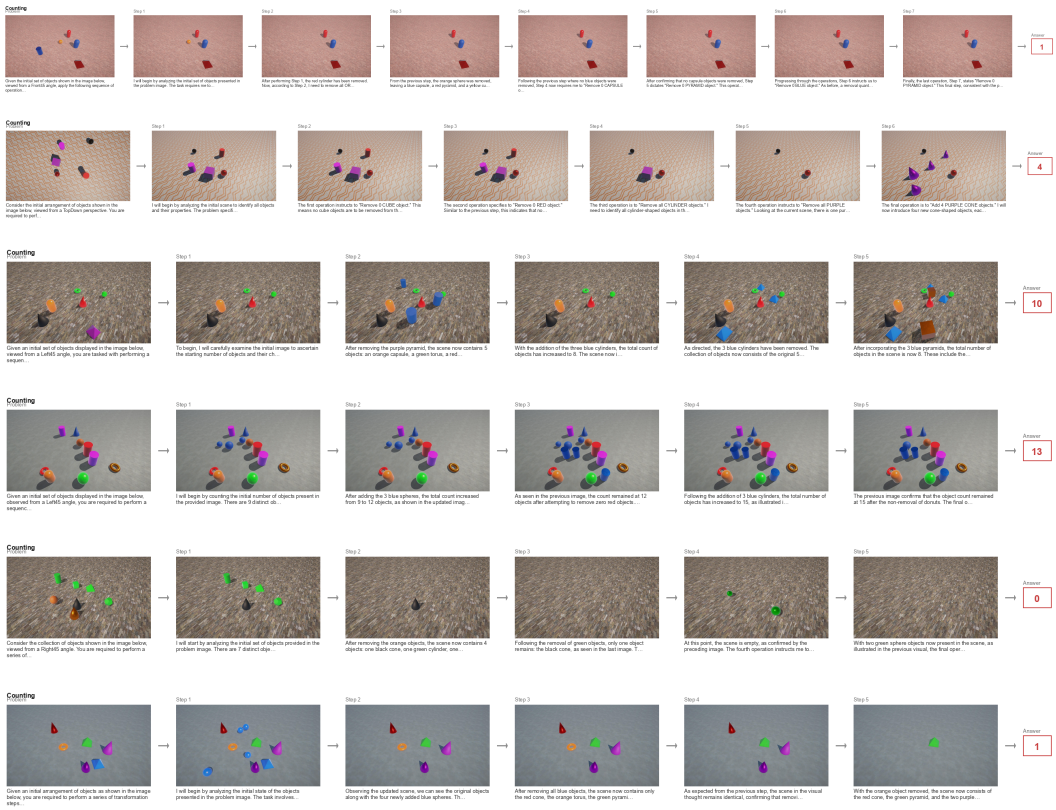

Figure A.6: Six training samples from the Counting subset, ranked by trace length (most intermediate frames first). The intermediate images are open-source "remove-an-object" frames, kept only after the helpfulness filter of Sec. C.5; the model updates its running count across them and emits the final integer.

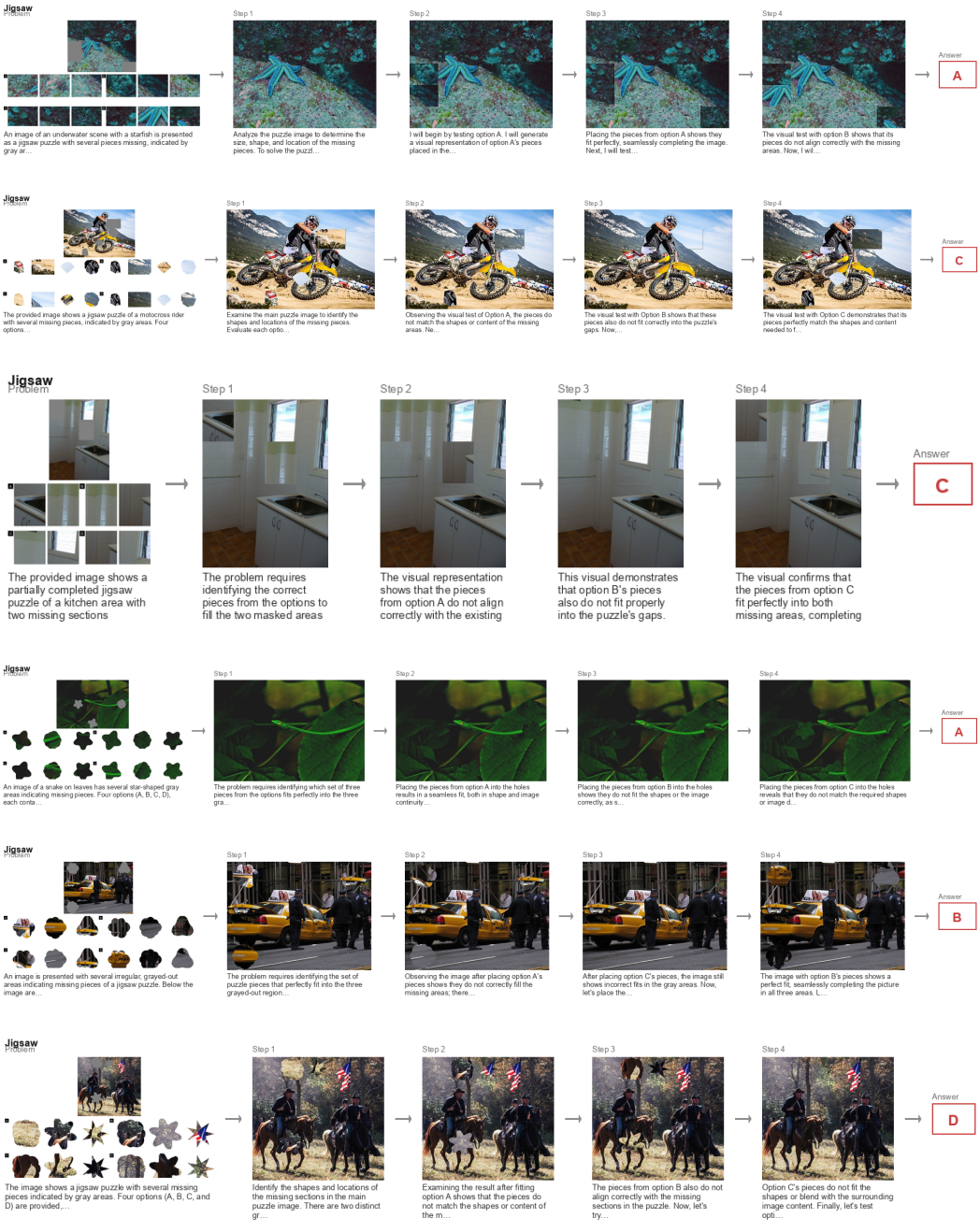

Figure A.7: Six training samples from the Jigsaw subset. The problem image stacks the puzzle source (top) and the four candidate assemblies; intermediate images progressively reconstruct the target tiling. The final answer is the option letter that matches the reconstruction.

C.7 MONET DATA CONVERSION

The Monet baseline reported in our experiments is a latent visual-reasoning model whose reasoning traces use a single <observation>...</observation> tag pair to mark short factual spans that were read off an auxiliary image. To retrain Monet on Sperry-43K we therefore convert each Sperry trace into the Monet supervision format. The conversion preserves the original interleaved structure: the system prompt, the user message, the number of intermediate images, and the final $\boxed{\text{answer}}$ answer are kept verbatim; image paths are prefixed with the subset name to match Monet's directory layout; the auxiliary-image marker <abs_vis_token> is rewritten in its paired form <abs_vis_token></abs_vis_token> to align with Monet's tokenizer. The only content change is the insertion of <observation> tags inside the assistant text, which we drive with a strong text-only LLM (Qwen3.5-397B-A17B served via Lepton) at temperature 0 with thinking disabled and an 8,192-token output budget. The system prompt is shown in Prompt A.5; the user message follows the template in Prompt A.6.

Prompt: Monet conversion system prompt (Qwen3.5-397B-A17B)

```
You tag Chain-of-Thought reasoning traces for latent visual reasoning
training.  Given an assistant response that interleaves TEXT and
auxiliary-image placeholders <abs_vis_token></abs_vis_token>, you
insert <observation>...</observation> tags around SHORT factual spans
that were OBSERVED FROM the auxiliary images.
HARD RULES:
1.  Wrap SHORT specific facts only:  a number, a count, a named
object/color/position, a coordinate, a label, an extracted
value.  Examples:  <observation>3</observation>, <observation>red
cube</observation>, <observation>(4,1)</observation>.
2.  NEVER wrap full sentences, plans, intents, transitions.
3.  POSITION RULE: <observation> tags are ONLY allowed in
the portion of the response that appears AFTER the first
<abs_vis_token></abs_vis_token>.
4.  Every <abs_vis_token></abs_vis_token> should have at least one
<observation> in some TEXT block that appears after it.
5.  Do NOT tag values that are given in the problem statement.
6.  Do NOT wrap the content inside \boxed{...}.
7.  Do NOT paraphrase or alter any word.  Stripping all <observation>
tags from your output must yield the exact original text
character-for-character.
8.  Preserve ALL <abs_vis_token></abs_vis_token> markers, verbatim and
in the original positions.
9.  Output ONLY the tagged assistant response.
```

Prompt: Monet conversion user message (template)

```
QUESTION (for context only, DO NOT tag anything already stated here):
{question}
ASSISTANT RESPONSE TO TAG (insert observation tags; preserve all other
content):
{assistant_text}
```

We then validate every returned trace and reject malformed outputs. The validator enforces: (i) the count of <abs_vis_token></abs_vis_token> markers is unchanged from the input; (ii) opening and closing <observation> counts match; (iii) any tags placed before the first auxiliary image are stripped (rule 3); and (iv) after removing all <observation> tags and applying a small Unicode normalization (smart quotes, en/em dashes), the output is identical to the input character-for-character. Failures trigger up to three retries with the same prompt; on persistent failure we keep the untagged trace, which Monet treats as a no-observation example. We run the conversion in parallel over the six subsets of Sperry-43K with a global concurrency cap of 32 requests; the conversion preserves the same number of training examples and the same step count per trace as

the original Sperry-43K, so the Monet baseline is trained on identical reasoning content modulo the observation-tag annotation. We train Monet with its official repository and default hyperparameters, changing only the training budget to match ours.