GROUNDEDVCOT: GROUNDING INTERLEAVED TEXT-IMAGE REASONING WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual intelligence requires reasoning about how a scene changes and using the resulting state to guide later decisions. Unified multimodal models can make this explicit by interleaving textual reasoning with generated images. Yet generated images also enter the context for later steps. If any generated image misrepresents an action or contradicts the text, the error can compound across the trajectory. We introduce GroundedVCoT, a reinforcement-learning (RL) framework for grounding interleaved text–image chain-of-thought. At each step, task-specific verifiers assess textual claims and generated visual states against the state implied by the model's own decisions. Step-wise grounding rewards and task success yield a shared advantage that updates both text and image policies. Verification follows the model's trajectory, allowing alternative valid solutions while preserving its self-generated visual history. We evaluate three distinct demands on visual reasoning: planning moves in Rush Hour, constructing a semantic figure in Tangram, and tracking rotations in Rolling Dice. Relative to supervised fine-tuning, GroundedVCoT raises Rush Hour solve rate from 48.0% to 71.3% (+23.3%), Tangram accuracy from 25.6% to 50.5% (+24.9%), and Rolling Dice accuracy from 25.6% to 60.0% (+34.4%), also consistently exceeding terminal reward RL, supporting the value of grounding intermediate steps rather than rewarding task completion alone.

1 INTRODUCTION

Many visual intelligence tasks require predicting how actions change a scene and using those predictions to guide later decisions. Unified multimodal models (UMMs) (Deng et al., 2025) can make such predictions explicitly by generating images within a reasoning trajectory. In *interleaved text-image chain-of-thought (CoT) reasoning*, a model produces textual reasoning, generates an intermediate visual state, and conditions later decisions on that state (Li et al., 2025b). The generated image thus becomes part of the evidence used for subsequent reasoning. If it depicts the wrong state, the error can propagate even when the preceding textual reasoning was correct. This creates a need for training signals that assess whether both what the model says and what it draws remain grounded in the evolving task state.

Current models struggle with this requirement. STARE (Li et al., 2025c) and SpatialViz-Bench (Wang et al., 2026) document failures in spatial transformation and geometric visualization, while BabyVision (Chen et al., 2026) finds errors on basic visual tasks that young children solve reliably. MIRA (Zhou et al., 2025) shows that intermediate visual cues can improve reasoning when they are provided correctly. Yet MentisOculi (Zeller et al., 2026) finds that multimodal models struggle to maintain correct visual states across multiple steps and sometimes fail to use even ground-truth visual aids. Intermediate images can therefore help only when they faithfully represent the evolving reasoning state and are used consistently by later steps.

Figure 1 illustrates the resulting failure mode. The next step treats the head as complete and attaches another piece to the misplaced triangle, moving the construction farther from the target silhouette. We call this propagation *compounding drift*. The misplaced triangle may satisfy low-level geometric constraints while contradicting the semantic claim in the text. Conversely, text and image may agree with each other while jointly violating the task rules. Detecting these failures requires step-wise checks of both modalities. A terminal reward can identify an unsuccessful construction, but cannot locate the claim or image that initiated the drift.

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.

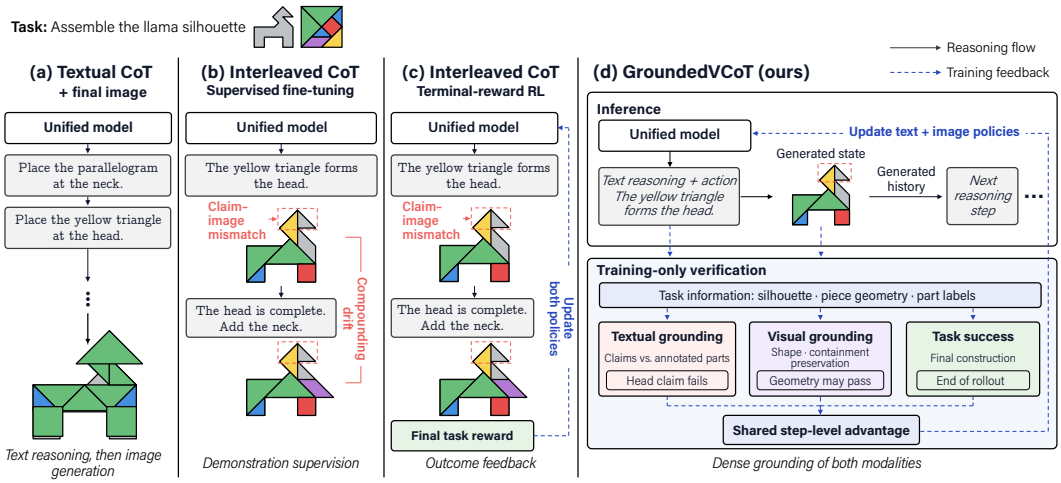


Figure 1: **Grounding interleaved text-image chain-of-thought.** Errors in generated states can propagate through later reasoning. GroundedVCoT applies step-wise textual and visual grounding rewards and assigns their shared advantage to both policies.

Existing approaches provide several forms of supervision for interleaved generation. Supervised fine-tuning learns from demonstration trajectories. UniGRPO (Liu et al., 2026b) jointly optimizes text and image policies using final task-level rewards. BRAID (Hu et al., 2026) introduces intermediate visual feedback for multi-turn reasoning, and VBVR-Pro (Xu et al., 2026) trains visual reasoning through video generation using deterministic task rewards. These approaches establish that unified text-image policies can be optimized jointly, but do not explicitly verify, at every reasoning step, whether the textual claims and generated visual state are both consistent with the task transition induced by the model's own trajectory.

We introduce GroundedVCoT, a reinforcement learning framework for step-wise grounding of interleaved visual reasoning. At each step, the model produces textual reasoning and an intermediate visual state. During training, task-specific verifiers evaluate checkable textual claims and properties of the generated state against the evolving task state. For tasks with deterministic dynamics, the verifier executes the model's predicted action and evaluates both modalities against the resulting transition. For tasks such as Tangram, geometric and semantic constraints provide analogous supervision. Crucially, verification follows the model's own trajectory rather than a fixed demonstration, so alternative valid reasoning paths can receive full grounding credit. The resulting dense grounding rewards are combined with task success into a shared step-level advantage that updates both the autoregressive text policy and the flow-matching image policy. Thus, errors detected in either modality can influence both components of the interleaved reasoning process.

We evaluate GroundedVCoT across complementary forms of visual reasoning, including visual planning with Rush Hour (Zeller et al., 2026), semantic visual construction with Tangram (Ji et al., 2022), and spatial rotation with Rolling Dice (Zhou et al., 2025) (Figure 2). Across these settings, we measure both intermediate-state grounding and final task performance. On Rush Hour, GroundedVCoT increases solve rate from 48.0% with supervised fine-tuning and 63.3% with terminal-reward RL to 71.3%, with the largest improvements on longer reasoning trajectories. Also, based on our ablations, adding textual and visual grounding consistently improves accuracy over the task-success based terminal reward RL.

In summary, our contributions are threefold: (1) we identify *compounding drift* as a central failure mode of interleaved text-image reasoning, where inconsistencies between textual reasoning and generated visual states propagate through subsequent steps, (2) we introduce GroundedVCoT, which provides dense, solution-agnostic textual and visual grounding along the model's own trajectory and uses shared cross-modal credit assignment to optimize both policies, (3) we evaluate grounded interleaved reasoning across visual planning, semantic construction, and spatial rotation, measuring both intermediate grounding and final task success.

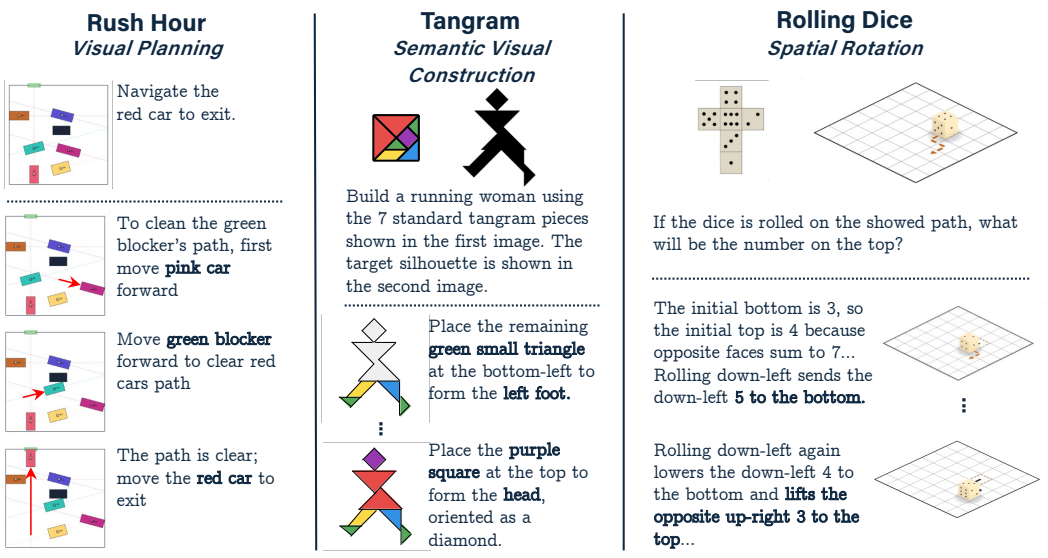


Figure 2: **Three distinct forms of interleaved visual reasoning.** Examples from Rush Hour (visual planning), Tangram (semantic visual construction), and Rolling Dice (spatial rotation).

2 Related Work

Benchmarks. Recent benchmarks expose complementary limitations in visual reasoning. STARE (Li et al., 2025c) evaluates multi-step visual simulation across geometric transformations, spatial composition, and real-world settings, while SpatialViz-Bench (Wang et al., 2026) targets spatial visualization abilities such as mental rotation and 2D-3D transformation. BabyVision (Chen et al., 2026) isolates basic visual abilities with limited dependence on language or domain knowledge. More specialized benchmarks study compositional construction through geometrically verifiable Tangram problems (Liu et al., 2026a), or couple multimodal understanding and generation across diverse reasoning domains, as in Uni-MMMU (Zou et al., 2026). MIRA (Zhou et al., 2025) studies tasks where annotated intermediate visualizations can aid reasoning, whereas MentisOculi (Zeller et al., 2026) examines whether unified models can generate and use such visualizations themselves. IMUG-Bench (Meng et al., 2026) and VBVR-Pro (Xu et al., 2026) extend this line to multi-turn and verifiable native visual reasoning. Our evaluation focuses on the reliability of the model's own intermediate states: whether textual decisions and generated images remain grounded as they are fed back into a multi-step reasoning trajectory.

Explicit and latent visual reasoning. Unified models such as Anole (Chern et al., 2024), BAGEL (Deng et al., 2025), and SenseNova-U1 (Diao et al., 2026) support interleaved text and image generation within a single sequence. Zebra-CoT (Li et al., 2025a) and Uni-CoT (Qin et al., 2026) train models with interleaved reasoning traces that include intermediate visual states. Other methods reason in continuous visual representations. Mirage (Yang et al., 2025) interleaves language with visual embeddings, Monet (Wang et al., 2025) reasons through continuous visual thought representations, and AD-Loop (Wu et al., 2026) alternates analysis with implicit visual drafting. Latent representations avoid rendering a full image at every step, but they cannot be compared as directly with rendered or geometric task states. We instead work with explicit generated images, which allows the model's visual state and textual claims to be checked against the task at each step.

Reinforcement learning for unified generation. Flow-GRPO (Liu et al., 2025) enables online reinforcement learning for flow-matching image models by introducing stochastic denoising transitions with tractable likelihoods. Nie et al. (2026) applied group-relative optimization to interleaved text–image outputs using rewards for textual relevance, visual alignment, and output structure. Uni-GRPO (Liu et al., 2026b) jointly optimizes autoregressive textual reasoning and flow-matching image generation under a shared reward. More recent work extends reinforcement learning to sequential visual reasoning. BRAID (Hu et al., 2026) considers multi-turn interleaved reasoning and provides intermediate visual feedback using a VLM, while VBVR-Pro (Xu et al., 2026) trains visual reasoning

through video generation with deterministic task rewards. GroundedVCoT differs in how intermediate supervision is defined and assigned. At each step, we verify both checkable textual claims and the generated visual state against the same evolving task state or transition induced by the model's own trajectory. The grounding rewards therefore measure whether the two modalities remain faithful to the task during reasoning, rather than only whether an intermediate image is plausible or whether the final trajectory succeeds. A shared step-level advantage then assigns this feedback to both the autoregressive text policy and the flow-matching image policy.

3 METHODOLOGY

GroundedVCoT trains a unified multimodal model to keep its textual reasoning and generated visual states grounded throughout an interleaved rollout. The model reasons from its own generated images, exactly as it will at inference time. During training, a task-specific verifier separately maintains the task state needed to check each step. It scores checkable claims in the text and properties of the generated image, but its verified state is never inserted into the model's context. We combine these step-wise grounding signals with task success and use the resulting shared advantage to update both the autoregressive text policy and the flow-matching image policy. Figure 3 gives an overview.

3.1 INTERLEAVED VISUAL REASONING

Let x denote a task instance containing an instruction and its visual inputs, such as an initial board or a target silhouette. The unified model generates an interleaved trajectory

$$\tau = (y_1, I_1, \ldots, y_L, I_L), \tag{1}$$

where L is the number of reasoning steps. Each textual output y_t contains a reasoning segment and a structured action a_t. The text and image experts generate

$$y_t \sim \pi^{\mathrm{txt}}_{\theta_{\mathrm{txt}}}(\cdot \mid h_t), \tag{2}$$

$$I_t \sim \pi^{\mathrm{img}}_{\theta_{\mathrm{img}}}(\cdot \mid h_t, y_t), \tag{3}$$

with

$$h_t = (x, y_1, I_1, \ldots, y_{t-1}, I_{t-1}). \tag{4}$$

The history h_t contains the model's own generated images. A verifier determines what the task state should be after an action, but this verified state does not replace I_t in the rollout. The model therefore remains exposed to errors from its own generations during training.

This distinction matters in multi-step reasoning. The text may specify an invalid action, the image may fail to depict the stated action, or the two may agree while violating the task constraints. Since I_t becomes part of h_{t+1}, any of these errors can alter later decisions and produce the compounding drift described in Section 1.

3.2 TASK-SPECIFIC STEP VERIFICATION

A task-specific verifier $\mathcal{V}$ evaluates the transition proposed at each step. Besides the model history, the verifier maintains a separate state $\bar{s}_t$ used only for verification. Given the previous verified state and the current model outputs,

$$\big(r^{\mathrm{txt}}_t, r^{\mathrm{vis}}_t, \bar{s}_t, v_t\big) = \mathcal{V}\big(x, \bar{s}_{t-1}, y_t, I_t\big). \tag{5}$$

Here, $r^{\mathrm{txt}}_t \in [0,1]$ scores checkable claims in y_t, and $r^{\mathrm{vis}}_t \in [0,1]$ scores the generated visual state. The indicator $v_t \in \{0,1\}$ records whether the structured action can be parsed and processed by the verifier. Claims outside the task representation are left unscored rather than evaluated by a language-model judge.

The verifier follows the model's own trajectory rather than a fixed demonstration. Consider two legal Rush Hour solutions that take different first moves. Comparing both rollouts with one demonstration would mark one trajectory wrong as soon as their actions diverge. In Eq. 5, the verifier instead evaluates the state transition induced by the model's own action. Alternative valid trajectories can therefore receive full grounding credit.

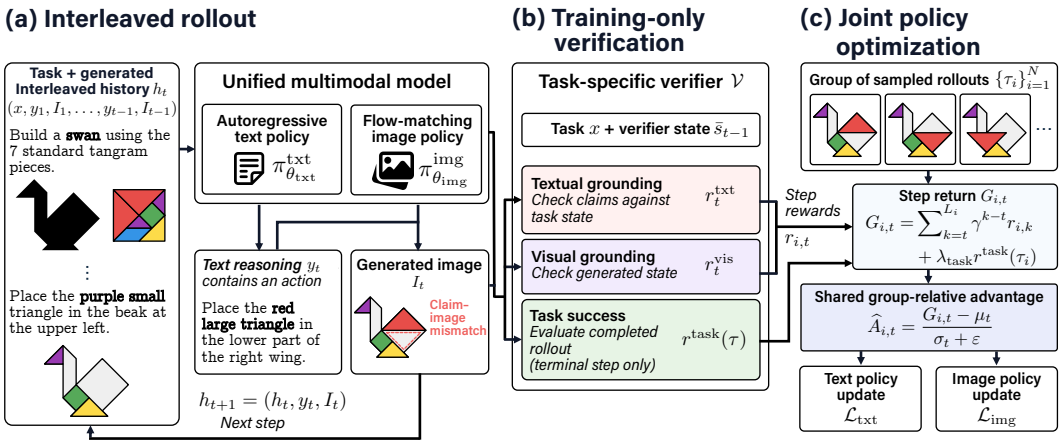


Figure 3: **Overview of GroundedVCoT.** The unified multimodal model generates textual reasoning, an action, and a visual state at each step. A task-specific verifier checks the text and image against the evolving task state while the model continues to reason from its own generated history. Step-wise grounding and task success form a shared group-relative advantage that updates both policies.

Grounding and action quality are separate. A legal but unhelpful action may be described and depicted correctly, so it can receive high textual and visual grounding scores. Whether that action contributes to solving the task is handled by the task-level reward in Section 3.3.

For tasks with deterministic dynamics, including Rush Hour and Rolling Dice, the verifier parses a_t and applies the task transition function

$$\bar{s}_t = \mathcal{T}(\bar{s}_{t-1}, a_t). \tag{6}$$

The visual verifier compares I_t with a rendering of $\bar{s}_t$. The textual verifier checks claims about the preceding state, the selected action, and its effect against $(\bar{s}_{t-1}, a_t, \bar{s}_t)$.

Tangram requires a different verifier because a free-form placement does not define a unique rendered next image. Its verifier recognizes the newly placed piece in I_t and evaluates its geometry relative to the target silhouette and the preceding construction. Visual checks cover piece identity and shape, containment, and preservation of previously placed pieces. Textual checks compare the claimed placement and semantic part, such as the head or left foot, with the recognized piece and its annotated target region. The verifier stores accepted piece geometry in $\bar{s}_t$ for the next step. This gives the same interface as Eq. 5 without requiring simulator replay.

3.3 Grounding and Task Rewards

We score each step along one channel per modality. The visual channel rewards the generated image, and the textual channel rewards both the structure of the generated text and the claims it makes:

$$r_t = \lambda_{\text{vis}}\, v_t\, r^{\text{vis}}_t + \lambda_{\text{txt}} \big(r^{\text{fmt}}_t + \lambda_c\, v_t\, r^{\text{txt}}_t \big). \tag{7}$$

Here λ_{vis} and λ_{txt} weight the two modalities, and λ_c weights claim grounding against format validity within the textual channel. Grounding measures whether the model faithfully describes and depicts its transition. It does not reward an action merely for moving the task closer to completion. A legal but unhelpful move can therefore receive high grounding scores and a low task score.

Verification also requires a structured, executable action, so the format validity reward r^{fmt}_t is given to outputs that follow the required structure and refer to valid task entities that can be processed by the verifier $\mathcal{V}$. The mask v_t records whether that check succeeds. If $v_t = 0$, the verifier cannot establish the proposed transition, so both grounding terms vanish and only format validity contributes.

A separate trajectory-level reward $r^{\text{task}}(\tau) \in [0, 1]$ measures task completion or verified progress. Simulator-based tasks evaluate the executed action sequence, while construction tasks evaluate the

Table 1: **Task-specific grounding and task-success rewards.**

Task	Visual grounding r_t^{vis}	Textual grounding r_t^{txt}	Task success r^{task}
Rush Hour	Per-car mask IoU with the simulated next state	Claims about blockers, legal motion, and the action's effect	Target car exits, with partial credit for verified progress
Tangram	Piece shape and containment, plus preservation of prior pieces	Piece identity, relative placement, and claimed semantic part	Final silhouette coverage and number of valid placed pieces
Rolling Dice	One minus normalized pixel error to the rendered orientation	Fraction of stated face assignments that match the verified state	Correct final answer and compliance with the prescribed rotations

final recognized geometry. For step t, we define

$$G_t = \sum_{k=t}^{L} \gamma^{k-t} r_k + \lambda_{\text{task}} r^{\text{task}}(\tau). \tag{8}$$

The discounted sum propagates later grounding failures to earlier steps whose outputs shaped the subsequent context. The task reward is included in every G_t, so each decision also receives feedback from the final outcome.

Table 1 summarizes the verifier and task rewards. Appendix B gives their scoring rules and thresholds

3.4 Shared Cross-Modal Credit Assignment

The text and image experts use different likelihood models, but their outputs form the same reasoning step. We therefore compute one advantage from the combined return and use it for both policies. For each task instance x, we sample N trajectories $\{\tau_i\}_{i=1}^{N}$ from the rollout policy. The number of reasoning steps L is determined by the instance, so all trajectories in a group share the same horizon and we normalize returns step by step within the group:

$$\widehat{A}_{i,t} = \frac{G_{i,t} - \mu_t}{\sigma_t + \varepsilon}, \qquad \mu_t = \frac{1}{N} \sum_{j=1}^{N} G_{j,t}, \tag{9}$$

where σ_t is the standard deviation of $\{G_{j,t}\}_{j=1}^{N}$. Normalizing at each step rather than over the whole trajectory keeps the comparison between rollouts at the same point in the reasoning process. No learned critic is required. Because both policies are updated with this single $\widehat{A}_{i,t}$, a poor visual grounding score affects the text policy that produced the action as well as the image policy that rendered its consequence, and textual grounding reaches the image policy in the same way. This shared assignment is what couples reasoning and generation during training.

Policy optimization. We update each modality with its corresponding policy-gradient objective. For the autoregressive text policy, let

$$\rho_{i,t}^{\text{txt}} = \frac{\pi_{\theta_{\text{txt}}}^{\text{txt}}(y_{i,t} \mid h_{i,t})}{\pi_{\text{old}}^{\text{txt}}(y_{i,t} \mid h_{i,t})} = \exp\left(\sum_{q=1}^{|y_{i,t}|} \log \frac{\pi_{\theta_{\text{txt}}}^{\text{txt}}(y_{i,t,q} \mid h_{i,t}, y_{i,t,<q})}{\pi_{\text{old}}^{\text{txt}}(y_{i,t,q} \mid h_{i,t}, y_{i,t,<q})} \right). \tag{10}$$

denote the likelihood ratio of the whole reasoning step $y_{i,t}$, obtained by summing the log-probabilities of its tokens. We use the clipped GRPO objective (Shao et al., 2024)

$$\mathcal{L}_{\text{txt}} = -\mathbb{E}_{i,t}\left[\min\left(\rho_{i,t}^{\text{txt}} \widehat{A}_{i,t}, \text{clip}\big(\rho_{i,t}^{\text{txt}}, 1-\epsilon, 1+\epsilon\big) \widehat{A}_{i,t} \right) \right]. \tag{11}$$

The reasoning step is therefore the unit of the text update: clipping acts on the segment likelihood, and $y_{i,t}$ receives the advantage of its step as a whole rather than one ratio per token.

For image generation, Flow-GRPO (Liu et al., 2025) introduces stochastic sampling transitions with tractable conditional likelihoods. Let $\rho_{i,t,m}^{\text{img}}$ denote the image-policy ratio at flow step m. We optimize

$$\mathcal{L}_{\text{img}} = -\mathbb{E}_{i,t,m}\left[\min\left(\rho_{i,t,m}^{\text{img}} \widehat{A}_{i,t}, \text{clip}(\rho_{i,t,m}^{\text{img}}, 1-\epsilon, 1+\epsilon) \widehat{A}_{i,t} \right) \right] + \beta\, \mathcal{R}_{\text{img}}. \tag{12}$$

Following UniGRPO (Liu et al., 2026b), $\mathcal{R}_{\text{img}}$ regularizes the image velocity field toward the frozen supervised policy. The joint objective is

$$\mathcal{L} = \lambda_{\text{txt}}\mathcal{L}_{\text{txt}} + \mathcal{L}_{\text{img}}. \tag{13}$$

The modality weight λ_{txt} of Equation (7) also scales the text loss, so a single parameter controls how much the textual channel counts, both in the return and in the update.

4 EXPERIMENTS

We evaluate whether step-wise grounding improves final task performance, whether textual and visual grounding provide complementary signals. The evaluation spans the three reasoning tasks in Figure 2: Rush Hour, Tangram, and Rolling Dice.

4.1 EXPERIMENTAL SETUP

Baseline. We use BAGEL (Deng et al., 2025) as the unified multimodal backbone.[1] For each structured task, we first supervise interleaved text–image trajectories and then optimize the text and image policies with the rewards in Section 3. The model conditions each step on its own previously generated images during both training rollouts and evaluation. We compare the resulting GroundedVCoT checkpoint with the SFT checkpoint and with reinforcement learning using task reward alone. Where applicable, we also compare against text-only, unified multimodal, and image or video generation baselines. Appendix C specifies training data, sampling settings, reward weights, compute, and the inference budget for every method.

Tasks and metrics. Rush Hour requires moving the target car to the exit through a sequence of legal moves. We report the fraction of puzzles solved at each difficulty level, L1–L5, and in aggregate. Tangram requires constructing a target silhouette from seven pieces. We report construction accuracy under the geometric evaluation described in Appendix B, together with the number of fully solved puzzles out of 99. Rolling Dice requires tracking the visible faces through a prescribed sequence of rotations. We report final-answer accuracy by difficulty level and in aggregate.

Comparison protocol. All comparisons between our SFT, task-reward RL, and GroundedVCoT checkpoints use the same backbone, evaluation set, prompt format, and inference budget. We report denominators for each difficulty level and task category. For external models, we distinguish scores taken from prior work from scores obtained under our evaluation protocol. Appendix D provides prompts, decoding settings, and answer extraction details.

4.2 GROUNDED REASONING ACROSS STRUCTURED TASKS

Table 2 evaluates GroundedVCoT across three tasks that place different demands on intermediate visual states: planning legal moves in Rush Hour, constructing a semantic figure in Tangram, and tracking orientation in Rolling Dice. The principal comparison is between BAGEL SFT and BAGEL-GroundedVCoT, which share the same backbone and evaluation protocol. We include other model families to provide context, but their training data and inference procedures differ.

Visual planning. GroundedVCoT raises Rush Hour solve rate from 48.0% to 71.3%. The gain is concentrated where maintaining an accurate board state matters most: performance rises from 40.0% to 83.3% at L3 and from 3.3% to 56.7% at L4, while both models solve all L1 puzzles. This difficulty profile is consistent with grounding helping the model sustain a multi-step plan, rather than merely improving its handling of easy initial states. GroundedVCoT also exceeds the fine-tuned latent-visual baseline Mirage overall (71.3% vs. 56.7%), though the distinct training and representation choices prevent attributing that gap solely to explicit images. L5 remains difficult: the 16.7% solve rate is a substantial improvement over SFT's 0%, but is well below the human average of 56.0%.

Semantic visual construction. On Tangram, construction accuracy rises from 25.6% with SFT to 50.5% with GroundedVCoT. The task requires both geometrically valid placements and semantic consistency with the target figure, which motivates joint visual and textual grounding. Joint textual and

[1]We also apply the same training framework to SenseNova-U1 to test whether its gains extend beyond the BAGEL backbone, and this comparison is reported in Appendix G.

Table 2: **Performance on the three structured task settings.** Rush Hour and Rolling Dice report solve or answer accuracy by difficulty level. Tangram reports construction accuracy and fully solved puzzles. Group headings distinguish model families and reasoning or generation approaches, with our BAGEL training comparison shown separately.

	Rush Hour						Tangram		Rolling Dice					
Model / method	Avg.	L1	L2	L3	L4	L5	Acc.	Solved	Avg.	L1	L2	L3	L4	L5
Human average	0.811	1.000	0.987	0.847	0.660	0.560	–	–				–		
Vision–language models														
Qwen3-VL-235B Thinking	0.233	0.200	0.800	0.133	0.033	0.000	–	–	0.131	0.156	0.188	0.094	0.156	0.063
Gemini 2.5 Flash	0.253	0.500	0.567	0.167	0.000	0.033	–	–	0.150	0.150	0.125	0.175	0.175	0.125
Gemini 3 Pro	0.493	1.000	0.800	0.267	0.300	0.100	–	–				-		
Gemini 3.1 Pro				–			–	–	0.325	0.325	0.350	0.300	0.325	0.325
GPT-5.1	0.413	1.000	0.767	0.267	0.033	0.000	–	–	0.156	0.250	0.219	0.125	0.125	0.063
Latent visual reasoning														
Mirage (Yang et al., 2025), fine-tuned	0.567	1.000	0.967	0.500	0.267	0.100	–	–	0.531	0.625	0.719	0.656	0.344	0.313
Image generation														
Gemini 2.5-I (Flash Image)	0.180	0.400	0.400	0.100	0.000	0.000	0.0740	0/99	0.135	0.150	0.150	0.150	0.150	0.075
Gemini 3-I (Pro Image)	0.333	0.500	0.700	0.333	0.067	0.067	0.1220	1/99	0.225	0.575	0.100	0.125	0.175	0.150
Gemini 3.1 Flash-Lite Image				–			0.0131	0/99	0.145	0.050	0.125	0.225	0.175	0.150
Gemini 3.1 Flash Image				–			0.0276	0/99	0.275	0.400	0.450	0.175	0.125	0.225
Video generation														
Veo 3.1 (video generation)	0.287	1.000	0.400	0.033	0.000	0.000	0.0490	0/99				–		
Wan 2.6 (video generation)	0.227	0.967	0.100	0.067	0.000	0.000	–	–				–		
Wan 2.2, one-shot				–			0.0200	0/99				–		
Wan 2.2, finetuned				–			0.0725	0/99				–		
Programmatic visual construction (SVG)														
Qwen2.5-7B, SVG-trained				–			0.0279	0/99				–		
Qwen2.5-VL-7B, SVG-trained				–			0.0587	0/99				–		
Qwen2.5-VL-32B, SVG-trained				–			0.1197	0/99				–		
Unified multimodal models														
Emu3.5	0.022	0.000	0.037	0.071	0.000	0.000	–	–	0.162	0.062	0.125	0.218	0.156	0.250
SenseNova-U1 SFT	0.513	1.000	0.900	0.400	0.267	0.000	–	–				–		
Uni-COT SFT	0.500	1.000	0.867	0.367	0.233	0.033	–	–				–		
BAGEL	0.093	0.400	0.067	0.000	0.000	0.000	0.0027	0/99				–		
BAGEL SFT	0.480	1.000	0.967	0.400	0.033	0.000	0.2560	1/99	0.256	0.688	0.156	0.156	0.094	0.188
BAGEL-GroundedVCoT	**0.713**	**1.000**	**1.000**	**0.833**	**0.567**	**0.167**	**0.5051**	**5/99**	**0.600**	**0.906**	**0.844**	**0.625**	**0.375**	**0.250**

visual checks address these complementary requirements. Still, only 5 of 99 puzzles are fully solved, compared with 1 of 99 for SFT. This gap between construction accuracy and complete solutions shows that better intermediate grounding does not yet reliably produce a finished seven-piece figure. The low scores of image-, video-, and SVG-generation baselines further indicate that generating an image of a plausible figure is different from satisfying the task's piece-level construction constraints.

Spatial rotation. Rolling Dice tests whether the model can preserve orientation across successive transformations. GroundedVCoT improves accuracy from 25.6% to 60.0% over BAGEL SFT and exceeds the fine-tuned Mirage result of 53.1%. Gains appear at every difficulty level, including L3 (62.5% versus 15.6%) and L4 (37.5% versus 9.4%). Accuracy nevertheless declines as rotation sequences lengthen, reaching 25.0% at L5. Together with the Rush Hour results, this suggests that step-wise grounding reduces error accumulation but does not eliminate it on the longest trajectories.

5 What Drives the Gains?

Grounding rewards. Table 3 isolates the rewards on Rush Hour and Tangram. On Rush Hour. terminal reward RL raises L4 solve rate from 0.033 to 0.400, showing that the terminal signal provides useful training. Adding either grounding term alone changes the difficulty profile without consistently improving performance over task-reward RL across levels. Combining textual and visual grounding reaches 0.833 at L3 and 0.567 at L4, compared with 0.667 and 0.400 for task-reward RL. The combined reward therefore improves performance beyond the terminal signal on the harder levels shown here. Tangram shows a similar progression in construction accuracy, 0.4561 with task-reward RL, 0.4736 with visual grounding, and 0.5051 with the full method.

Shared credit assignment. We compare the shared advantage in Equation 9 with separate advantages for the text and image policies while keeping the reward terms fixed. On Rush Hour, shared

Table 3: **Ablation of grounding rewards.** All RL variants include the task-level reward, and the grounding rows add the indicated intermediate signals.

	Rush Hour						Tangram	
Method	Avg.	L1	L2	L3	L4	L5	Acc.	Solved
BAGEL SFT	0.480	1.000	0.967	0.400	0.033	0.000	0.2561	1/99
+ Terminal Reward RL	0.633	1.000	1.000	0.667	0.400	0.100	0.4561	1/99
+ Visual Grounding	0.627	1.000	0.933	0.600	0.433	0.167	0.4736	3/99
+ Textual Grounding	0.633	1.000	0.933	0.700	0.367	0.167	–	–
+ Visual & Textual Grounding	**0.713**	**1.000**	**1.000**	**0.833**	**0.567**	**0.167**	**0.5051**	**5/99**

credit raises L3 solve rate from 0.667 to 0.833 and L4 from 0.400 to 0.567, and both variants obtain 0.167 at L5 (Table 4). These results support sending grounding feedback from each modality to both policies. The analysis does not isolate whether the gain arises primarily through changed action selection, improved rendering, or their interaction, which we examine with the interventions below.

Table 4: **Rush Hour ablation of cross-modal credit assignment.** Both variants use visual and textual grounding rewards.

Credit assignment	Overall	L3	L4	L5
Per-head advantages	0.647	0.667	0.400	0.167
Shared advantage (GroundedVCoT)	**0.713**	**0.833**	**0.567**	0.167

Use of generated visual history. On Rush Hour, at inference time, we intervene on the image history while holding the textual trace fixed, then regenerate only the final answer step. Solve rate is highest with the model's own images (71.3%) and falls to 66.7–68.0% when they are removed or replaced (Table 5). This modest advantage suggests that visual history contributes to the final decision beyond the retained text. Yet replacing generated images with oracle images does not improve performance. Because the replacements may conflict with claims in the fixed textual trace, their lower solve rate cannot be attributed solely to the quality of the images.

Table 5: **Evaluation-time interventions on the visual history.**

Visual history	Solve	L3	L4	L5	Δ Solve
Own generated images (GroundedVCoT)	0.713	0.833	0.567	0.167	—
No image history	0.673	0.800	0.433	0.133	-0.0400
Oracle images	0.667	0.733	0.467	0.133	-0.0467
Random images	0.680	0.767	0.500	0.133	-0.0333

6 Conclusion

We studied a failure mode of interleaved text-image reasoning in which errors in generated visual states enter the reasoning history and propagate to later decisions. We refer to this as *compounding drift*. To address it, we introduced GroundedVCoT, which verifies textual claims and generated visual states throughout the model's own reasoning trajectory. This solution-agnostic supervision is combined into a shared step-level advantage that updates both the text and image policies. Across visual planning, semantic construction, and spatial rotation, step-wise grounding improves task performance over supervised fine-tuning and task-reward reinforcement learning. Our ablations show that textual and visual grounding provide complementary signals and that shared cross-modal credit improves multi-step reasoning. Intermediate images help when they remain consistent with the reasoning process that generates and uses them. The current framework depends on task-specific verifiers. Extending step-wise grounding to less structured tasks, where state transitions cannot be simulated or checked through explicit constraints, is an important direction for future work.

AI USE STATEMENT

Generative AI tools were used in this work in two roles. First, generative multimodal models are part of the research pipeline itself. They produce the interleaved text-image reasoning trajectories studied in our experiments and are also evaluated as external baselines. The grounding rewards and benchmark scores reported in the paper are computed using the task-specific parsers, simulators, geometric checks, and image comparisons described in the paper rather than language-model judges. Second, generative AI tools were used in a limited supporting role for coding assistance, literature discovery, and improving the clarity and presentation of the manuscript. They were not used as autonomous decision-makers for the research design, experimental protocol, or interpretation of the results. AI-assisted code and text were reviewed by the authors, and any suggested factual claims or citations were checked against their original sources. The authors take full responsibility for the final manuscript, implementation, experimental results, citations, and claims.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model backbone, interleaved training procedure, reinforcement learning objectives, reward construction, and evaluation protocols. The Appendix documents the task and dataset construction, train and test splits, task-specific verifiers, reward definitions and thresholds, optimization settings, compute configuration, and evaluation details. We also describe how the structured tasks are generated and scored so that the reported experiments can be reproduced independently of the training code. We plan to release the training and evaluation code, task generators and verifiers, prompts, and other evaluation resources upon completion of the review process.

REFERENCES

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, Ming Wu, Xiaobo Hu, Kaiyuan Chen, Yixin Ren, Yang Liu, Yuan Gong, and Kuan Li. Babyvision: Visual reasoning beyond language, 2026. URL https://arxiv.org/abs/2601.06521.

Ethan Chern, Jiadi Su, Yan Ma, and Pengfei Liu. Anole: An open, autoregressive, native large multimodal models for interleaved image-text generation, 2024. URL https://arxiv.org/abs/2407.06135.

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, Silei Wu, Weichen Fan, Wenjie Ye, Wenwen Tong, Xiangyu Fan, Yan Li, Yubo Wang, Zhijie Cao, Zhiqian Lin, Zhitao Yang, Zhongang Cai, Yuwei Niu, Yue Zhu, Bo Liu, Chengguang Lv, Haojia Yu, Haozhe Xie, Hongli Wang, Jianan Fan, Jiaqi Li, Jiefan Lu, Jingcheng Ni, Junxiang Xu, Kaihuan Liang, Lianqiang Shi, Linjun Dai, Linyan Wang, Oscar Qian, Peng Gao, Pengfei Liu, Qingping Sun, Rui Shen, Ruisi Wang, Shengnan Ma, Shuang Yang, Siyi Xie, Siying Li, Tianbo Zhong, Xiangli Kong, Xuanke Shi, Yang Gao, Yongqiang Yao, Yves Wang, Zhengqi Bai, Zhengyu Lin, Zixin Yin, Wenxiu Sun, Ruihao Gong, Quan Wang, Lewei Lu, Lei Yang, Ziwei Liu, and Dahua Lin. Sensenova-u1: Unifying multimodal understanding and generation with neo-unify architecture, 2026. URL https://arxiv.org/abs/2605.12500.

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 multi-modal reasoning as a unified decision process, 2026. URL https://arxiv.org/abs/2607.03748.

Anya Ji, Noriyuki Kojima, Noah Rush, Alane Suhr, Wai Keen Vong, Robert D. Hawkins, and Yoav Artzi. Abstract visual reasoning with tangram shapes, 2022. URL https://arxiv.org/abs/2211.16492.

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.

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. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=6vk6Xg24ZC.

Linjie Li, Mahtab Bigverdi, Jiawei Gu, Zixian Ma, Yinuo Yang, Ziang Li, Yejin Choi, and Ranjay Krishna. Unfolding spatial cognition: Evaluating multimodal models on visual simulations, 2025c. URL https://arxiv.org/abs/2506.04633.

Daixian Liu, Jiayi Kuang, Yinghui Li, Yangning Li, Di Yin, Haoyu Cao, Xing Sun, Ying Shen, Hai-Tao Zheng, Liang Lin, and Philip S. Yu. Tangrampuzzle: Evaluating multimodal large language models with compositional spatial reasoning, 2026a. URL https://arxiv.org/abs/2601.16520.

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, 2026b. URL https://arxiv.org/abs/2603.23500.

Lingyi Meng, Zecong Tang, Haoran Li, Tengju Ru, Zhejun Cui, Weitong Lian, Qi Kang, Hangshuo Cao, Yichen Zhu, Yechi Liu, Kaixuan Wang, Yu-Jie Yuan, Chunwei Wang, Yu Zhang, and Bo Dai. Imug-bench: Benchmarking unified multimodal models on interleaved understanding and generation, 2026. URL https://arxiv.org/abs/2606.09169.

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.

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, 2026. URL https://arxiv.org/abs/2508.05606.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language, 2025. URL https://arxiv.org/abs/2511.21395.

Siting Wang, Minnan Pei, Luoyang Sun, Cheng Deng, Yuchen Li, Kun Shao, Zheng Tian, Haifeng Zhang, and Jun Wang. Spatialviz-bench: A cognitively-grounded benchmark for diagnosing spatial visualization in mllms, 2026. URL https://arxiv.org/abs/2507.07610.

Shengqiong Wu, Bobo Li, Xinkai Wang, Xiangtai Li, Lei Cui, Furu Wei, Shuicheng Yan, Hao Fei, and Tat seng Chua. Synergizing understanding and generation with interleaved analyzing-drafting thinking, 2026. URL https://arxiv.org/abs/2602.21435.

Junxiang Xu, Ruisi Wang, Fanyi Pu, Maijunxian Wang, Ran Ji, Tongxi Zhou, Chenyang Gu, Jing Zuo, Hongcan Xiao, Yimeng Geng, Wanqi Yin, Chen Wei, Oscar Qian, Zhengan Yan, Ziqi Huang, Haiwen Diao, Liang Pan, Bo Li, Xiangyu Fan, Dezhi Luo, Fengyuan Yu, Zehong Zhao, Qingying Gao, Tinghui Zhu, Yilan Zhang, Jingqi Tong, Pinyuan Feng, Zhengze Jiang, Letian Wang, Ziyu Guo, Renrui Zhang, Jieneng Chen, Sonia Joseph, Constantin Venhoff, Saman Motamed, Mengyue Yang, Chandra Sripada, Alan Yuille, Philip Torr, Lvmin Zhang, Vikash Kumar, Daniel Khashabi, Nikolaus Kriegeskorte, Raphaël Millière, Vincent C. Müller, Anyi Rao, Quan Wang, Ziwei Liu, Dahua Lin, Lei Yang, Hokin Deng, and Zhongang Cai. Vbvr-pro: A scalable and

verifiable suite for native visual reasoning. *arXiv preprint arXiv:2608.26105*, 2026. URL https://arxiv.org/abs/2608.26105.

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.

Yiyang Zhou, Haoqin Tu, Zijun Wang, Zeyu Wang, Niklas Muennighoff, Fan Nie, Yejin Choi, James Zou, Chaorui Deng, Shen Yan, Haoqi Fan, Cihang Xie, Huaxiu Yao, and Qinghao Ye. When visualizing is the first step to reasoning: Mira, a benchmark for visual chain-of-thought, 2025. URL https://arxiv.org/abs/2511.02779.

Kai Zou, Ziqi Huang, Yuhao Dong, Shulin Tian, Dian Zheng, Hongbo Liu, Jingwen He, Bin Liu, Yu Qiao, and Ziwei Liu. Uni-MMMU: A massive multi-discipline multimodal unified benchmark. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 908–924, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.40. URL https://aclanthology.org/2026.acl-long.40/.

APPENDIX CONTENTS

A Task and dataset details
- A.1 Rush Hour
- A.2 Tangram
- A.3 Dice rotation

B Reward details
- B.1 Rush Hour
- B.2 Tangram
- B.3 Dice rotation

C Training configuration
D Evaluation details
E Baseline details
F Prompt details
- F.1 Rush Hour
- F.2 Tangram
- F.3 Rolling Dice

G Additional unified backbone results
H Qualitative results

A TASK AND DATASET DETAILS

Table A.1 summarizes the datasets. For every task, the reference solution sets only the rollout horizon L and rollouts are rewarded on the model's own actions rather than on imitation of the demonstration.

Table A.1: Dataset statistics. Steps per puzzle is the number of interleaved text–image steps. Validation puzzles are used only for checkpoint selection and are disjoint from the test puzzles.

Task	Train	Val	Test	Steps per puzzle
Rush Hour	1,000	100	150	1–5
Tangram	895	19	99	7
Dice rotation	1,000	40	160	1–5

A.1 RUSH HOUR

Boards are generated geometrically rather than on a unit grid, and each car may move along its own rotated local axis. An action is a car label followed by "forward" or "backward". The simulator moves that car to the furthest legal position along its axis, subject to the board boundary and collisions, and leaves the state unchanged if the move is unknown or blocked. Each puzzle has a metadata record with the board dimensions and exit, the geometry, color, label, and movement axis of every car, and the reference moves. This metadata drives the simulator and the renderer used for rewards but is not included in the prompt.

The training split contains exactly 200 puzzles at each shortest-solution length $L = 1, \ldots, 5$, and the held-out split contains 50 at each length. The held-out puzzles come from the MentisOculi generator (Zeller et al., 2026). The two splits share no generation seeds and no board with identical geometry, and each split keeps its own metadata. We split these 50 puzzles per length into test and validation puzzles as described in Section D, and report Rush Hour results on the same released pool of puzzles used for the MentisOculi baselines in Table 2.

Figure A.1 shows representative Rush Hour trajectories across the five difficulty levels, including the input board, step-by-step reasoning, selected moves, and generated intermediate states.

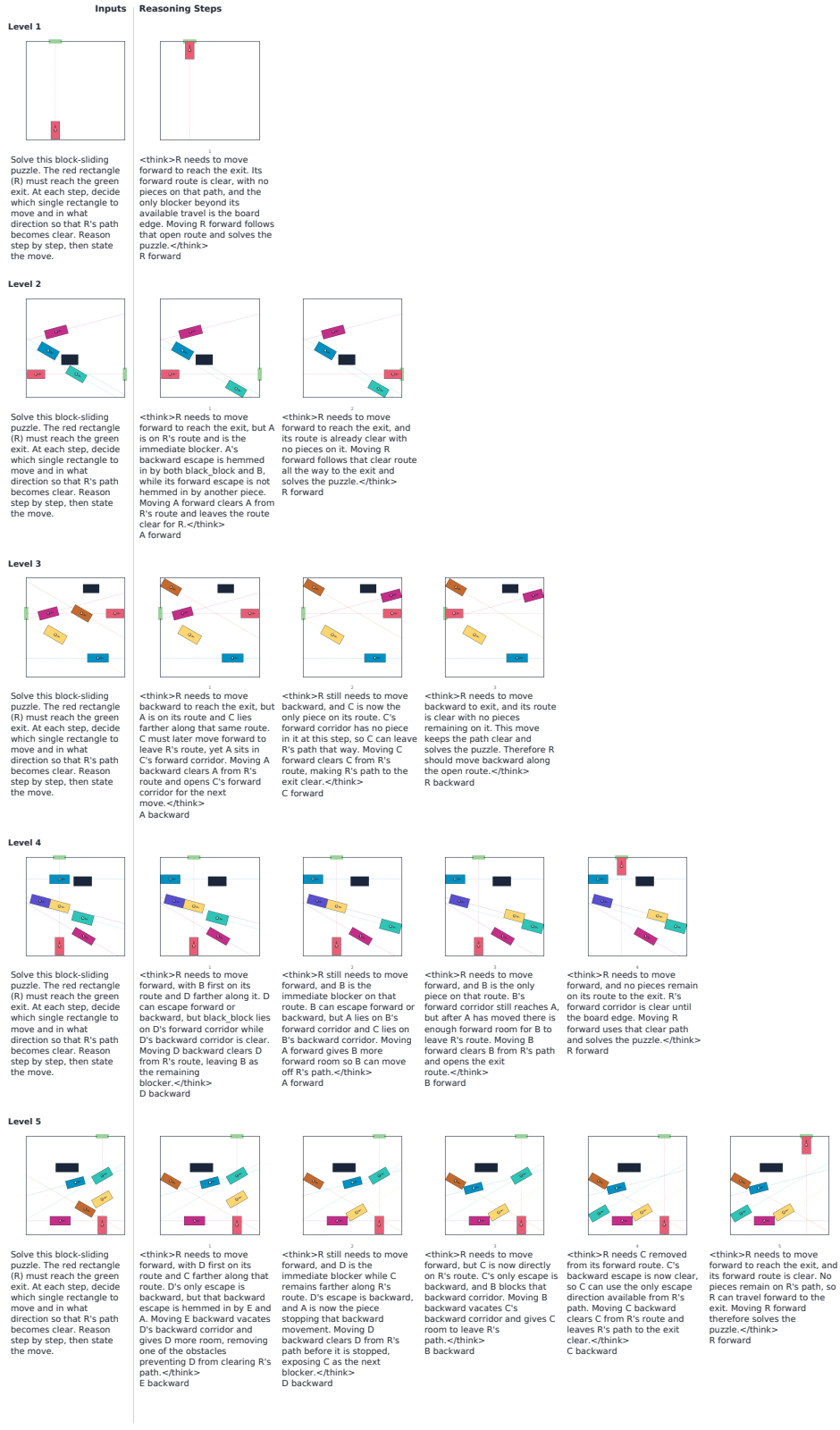


Figure A.1: **Rush Hour example trajectories.** Representative examples across the five difficulty levels. Each row shows the input puzzle followed by the interleaved reasoning, selected moves, and intermediate visual states leading toward the exit.

A.2 TANGRAM

Our data is derived from KiloGram (Ji et al., 2022), a collection of 1,013 human-annotated tangram puzzles. Each puzzle arranges the seven standard pieces (two large triangles, one medium triangle, two small triangles, one square, and one parallelogram) into a recognizable figure. KiloGram provides SVG polygons for each piece, whole-figure labels (e.g. “man kicking something”), and per-piece part labels (e.g. “left leg”, “head”). We use these geometric definitions and labels, but generate the step-by-step assembly sequences, color assignments, rendered images, and chain-of-thought reasoning traces ourselves. We split the puzzles into 895 for training, 99 for testing, and 19 for validation. Checkpoints are selected on the validation puzzles, so the test puzzles are never used to choose a checkpoint.

At every step the model receives the silhouette, the colored piece reference, and its previous drawings, and places one piece. Each piece is drawn in its own color, so pieces of the same size can be told apart. To verify semantic statements, we register the KiloGram part labels to our canvas: the original and rendered polygons must match with IoU at least 0.98, and the rendered silhouette must match the original with IoU at least 0.90 and 99.5% boundary agreement within two pixels.

Figure A.2 shows representative Tangram construction trajectories, from the input pieces and target silhouette to the completed seven-piece construction.



Figure A.2: **Tangram example trajectories.** Representative seven-step construction examples. Each row shows the colored piece reference and target silhouette, followed by the interleaved reasoning and intermediate visual states as the seven Tangram pieces are placed.

A.3 DICE ROTATION

The dataset contains four subsets of 250 training and 50 held-out puzzles each: cubes with pips, cubes with symbols, octahedra with digits, and octahedra with symbols. Each subset is balanced over 1–5 rolls, with 50 training and 10 held-out puzzles per subset and length. The question shows an initial 512×512 rendered scene, the roll route, and a net of the die. Cube questions ask for the final top face and octahedron questions for the final bottom face.

Each step contains the reasoning, the roll direction, and an explicit state of the die after the roll. For a cube the state gives the top and bottom faces and the four faces toward the camera-aware screen directions. For an octahedron it gives the bottom face, the face opposite it, and the visible and hidden side faces. The final step also contains the answer. We replay every roll with the original generator to verify the intermediate states and answers, and store them as reward-only metadata.

Figure A.3 shows representative Rolling Dice trajectories across the four task variants, including the initial state, reference net, reasoning for each roll, and generated intermediate states.

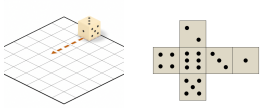
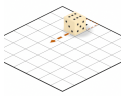
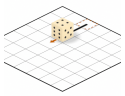
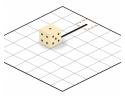
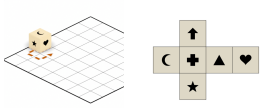
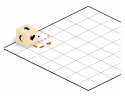
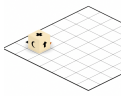
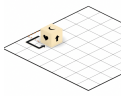
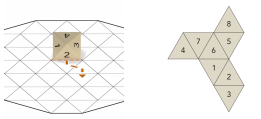
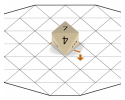
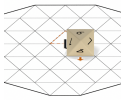
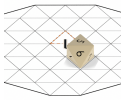
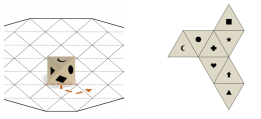
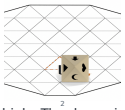
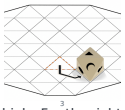
Inputs	Reasoning Steps		
Cube · Pips			
			
If the die is rolled along the shown path, what number will be on the top face at the end? The first image shows the initial state and roll path; the second image is the reference net. Reason through each roll.	<think>The initial top is 3, opposite the bottom face 4. Rolling down-left sends the down-left face 1 to the bottom and brings the opposite up-right face 6 to the top. The old top 3 moves to the down-left side, while the old bottom 4 moves to the up-right side; the up-left side remains 2 and the down-right side remains 5.</think> Roll down-left.	<think>Another down-left roll tips the current down-left face 3 onto the bottom, lifting the up-right face 4 onto the top. Meanwhile, the old top 6 rotates to the down-left side and the old bottom 1 swings to the up-right side. The perpendicular sides stay fixed, with 2 on the up-left side and 5 on the down-right side.</think> Roll down-left.	<think>For the final down-left roll, the up-right face 1 rises to become the top as the down-left face 6 rotates underneath to become the bottom. The previous top 4 takes the down-left position, and the previous bottom 3 takes the up-right position. With the up-left side still 2 and the down-right side still 5, the final pose has 1 on top and 6 beneath.</think> Roll down-left. The number on the top face at the end is 1. <answer>1</answer>
Cube · Symbols			
			
If the die is rolled along the shown path, which symbol will be on the top face at the end? The first image shows the initial state and roll path; the second image is the reference net. Reason through each roll.	<think>With triangle on the bottom, its opposite, moon, starts on top. Rolling down-left sends star to the bottom and brings arrow from the up-right side to the top, while moon rotates to the down-left side and triangle swings to the up-right side. Heart remains on the down-right side, and cross remains on the up-left side.</think> Roll down-left.	<think>The down-right roll tips heart to the bottom and lifts cross from the opposite up-left side to the top. Arrow moves from the top to the down-right side, while star swings from the bottom to the up-left side. Moon stays on the down-left side, and triangle stays on the up-right side.</think> Roll down-right.	<think>To roll up-right, triangle tips from the up-right side to the bottom, bringing moon from the opposite down-left side onto the top. Cross rotates from the top to the up-right side, and heart swings from the bottom to the down-left side. The perpendicular sides remain unchanged, with arrow on the down-right side and star on the up-left side.</think> Roll up-right. The symbol on the top face at the end is moon. <answer>moon</answer>
Octahedron · Digits			
			
If the die is rolled along the shown path, what number will be face down at the end? The first image shows the initial state and roll path; the second image is the reference net. Reason through each roll.	<think>Rolling down tips the die across the bottom edge shared with visible face 2, bringing that face onto the board. The new bottom is 2, and the face opposite it is 7 because opposite faces sum to 9.</think> Roll down.	<think>On the down-right roll, side face 3 swings down onto the board as face 2 lifts away. This makes 3 the new bottom and 6 the face opposite it, since $9 - 3 = 6$.</think> Roll down-right.	<think>Face 4 is now the visible side face across the edge used for the final down roll, so it tips onto the board. The bottom becomes 4, with face 5 opposite it by the sum-to-9 rule.</think> Roll down. The number face down at the end is 4. <answer>4</answer>
Octahedron · Symbols			
			
If the die is rolled along the shown path, which symbol will be face down at the end? The first image shows the initial state and roll path; the second image is the reference net. Reason through each roll.	<think>Rolling down tips the die over the edge shared by star and the visible square face, bringing square onto the board. Square becomes the bottom, and its paired opposite, heart, becomes the face opposite the bottom.</think> Roll down.	<think>The down-right roll carries the adjacent circle face onto the board as the die tips away from square. With circle now the bottom, the opposite-face rule places arrow opposite the bottom.</think> Roll down-right.	<think>For the right roll, the die tips over the edge shared by circle and the hidden cross face. Cross lands on the board as the new bottom, so its paired opposite, triangle, is now opposite the bottom.</think> Roll right. The symbol face down at the end is cross. <answer>cross</answer>

Figure A.3: **Rolling Dice example trajectories.** Representative examples from the four task variants. Each row shows the initial rendered state and reference net, followed by the interleaved reasoning, prescribed rolls, and intermediate visual states used to determine the final face.

B Reward Details

This section gives the exact per-task definitions of the rewards in Section 3.3. All rewards are computed by deterministic parsers, simulators, geometry checks, and image comparisons. The optimizer does not differentiate through them. In Rush Hour and Tangram, the textual grounding reward is the precision of the claims the parser can check,

$$r_t^{\text{txt}} = \frac{n_t^{\text{true}}}{\max\big(n_t^{\text{checkable}},\, 2\big)}, \tag{14}$$

where statements the parser cannot check abstain, and the floor of two in the denominator means a step needs at least two correct claims for full credit. In dice rotation, where each step states the full state of the die, it is the fraction of state fields stated correctly. Each subsection below defines the visual grounding reward r_t^{vis}, the validity reward r_t^{fmt}, the claims, and the task reward r^{task}.

B.1 Rush Hour

The simulator starts from the puzzle's initial state and executes each parsed action. Unknown or blocked moves leave the state unchanged. Let $\hat{s}_t$ be the simulated state after step t and $\mathcal{C}$ the set of cars. The IoU term compares per-car color masks M_c of the generated board and of a rendering of $\hat{s}_t$:

$$\text{IoU}_t = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{|M_c(I_t) \cap M_c(\text{render}(\hat{s}_t))|}{|M_c(I_t) \cup M_c(\text{render}(\hat{s}_t))|}. \tag{15}$$

Car-only IoU ignores color casts in the background, so we subtract $p_t = \lambda_{\text{bg}} \max(0, b_t - \tau_{\text{bg}})$, where b_t is the median distance from white over the reference background pixels, $\lambda_{\text{bg}} = 2$, and $\tau_{\text{bg}} = 0.025$. The image reward is halved when the action cannot be parsed and set to zero when the action does not change the state.

The format reward gives 0.25 for a `<think>` block, 0.25 for a parsable move, and 0.5 for naming a car that exists in the puzzle. Claims about the state (blockers, available movement, clear routes) are checked against the pre-action state. Claims about the effect of the move (e.g. that it clears a car or solves the puzzle) are checked against the actual transition. Repeated true claims are counted once. False claims lower the precision in Equation (14) but never make it negative, and a step whose action changes nothing receives no claim reward. With n_0 and n_L the numbers of cars blocking the exit corridor at the start and at the end,

$$r^{\text{task}} = \begin{cases} 1 & \text{if the red car reaches the exit,} \\ \alpha \max(0, n_0 - n_L)/n_0 & \text{otherwise, if } n_0 > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

with $\alpha = 0.5$. Success is determined by the simulated action sequence, not by whether the final image looks solved. We report the solve rate separately from this shaped training reward.

B.2 Tangram

A piece is recognized when its color region forms a single dominant connected component (at least 97% of its mass), fills at least 94% of its convex hull, matches the polygon geometry of the named piece up to rotation and reflection, and has an area between 0.80 and 1.25 times that of the reference piece. The image reward of a step is

$$r_t^{\text{vis}} = q_{\text{shape}} \cdot c_{\text{inside}} \cdot r_{\text{preserve}} \quad \text{if the addition is consistent, and 0 otherwise,} \tag{17}$$

where q_{shape} is the recognition quality of the new piece, $c_{\text{inside}} = \exp(-20 \cdot \text{overflow})$ penalizes newly added paint outside the silhouette, and r_{preserve} requires that at least 95% of previously recognized pieces are retained, within a two-pixel tolerance. An addition that introduces substantial unclaimed colors is inconsistent. The inventory of remaining pieces is recognized from the generated image rather than taken from the model's earlier claims, so a piece that is claimed but not drawn becomes available again.

The format reward gives 0.2 for the required reasoning and action format, 0.4 for an accepted action, and 0.4 for a correct inventory of remaining pieces. The final step must also contain an answer tag.

Claims about relative placement, location in the silhouette, orientation, and remaining pieces are checked against the previous generated image or against the new addition. Positive claim credit requires an accepted transition with $c_{\text{inside}} \geq 0.85$. A semantic claim such as "the square forms the head" is true if at least 85% of the piece's paint lies in the annotated region for that part. Unknown parts and ambiguous references abstain. The terminal reward is computed from the final image alone:

$$r^{\text{task}} = \text{coverage} \times \text{containment} \times \frac{\#\text{recognized pieces}}{7} \times \text{surplus} \times \text{overlap}, \tag{18}$$

where coverage is normalized by the coverage of the reference solution, and the last two factors penalize surplus paint and overlap between piece colors. A puzzle counts as fully solved when all seven pieces are recognized, coverage is at least 0.97, containment is at least 0.95, and surplus paint and overlap stay within the rendering noise.

B.3 Rolling Dice

Each step must contain a reasoning block, a roll direction, and a structured state object (Section A.3). The format reward gives 0.25 for a non-empty reasoning block with a well-formed state and direction, and 0.75 if the direction matches the prescribed roll. The claim reward is the fraction of required post-roll state fields that are correct, with a penalty for extra fields. For an octahedron, each side list counts as one field. It is zero if the direction is wrong or missing.

Let J_t be the target rendering and J_{t-1} the reference rendering of the previous step. A change mask selects pixels whose maximum channel difference between J_{t-1} and J_t exceeds 0.06, dilated with a 9×9 kernel. With $E(I, J_t) = 0.75\ \text{MSE}_{\text{change}}(I, J_t) + 0.25\ \text{MSE}_{\text{full}}(I, J_t)$,

$$r_t^{\text{vis}} = \text{clip}_{[0,1]}\left(1 - \frac{E(I_t, J_t)}{\min\big(E(\text{white}, J_t),\ E(J_{t-1}, J_t)\big)}\right), \tag{19}$$

so a blank canvas and a copy of the previous frame both score zero and an exact match scores one. The image reward is zero when the direction is wrong and halved when the state cannot be parsed. The task reward r^{task} is one only if the final answer is correct *and* every roll direction is correct; we also report answer-only accuracy.

C Training Configuration

Table A.2 lists the hyperparameters for each task.

Table A.2: **RL hyperparameters.** λ_{vis}, λ_{txt}, and λ_c are the step-reward weights in Equation (7), and λ_{task} is the task-reward weight in Equation (8).

Setting	Rush Hour	Tangram	Dice rotation
SFT initialization	Yes	Yes	Yes
Rollouts per question N	8	8	8
Image resolution (generation / conditioning)	512 / 512	256 / 256	512 / 512
Epochs	10	10	10
Learning rate (image / text)	5e-5 / 3e-5	5e-5 / 3e-5	5e-5 / 3e-5
Text sampling temperature	1.5	1.5	1.5
Maximum text tokens per step	256	320	384
Flow schedule points / SDE window W	15 / 3	15 / 3	15 / 3
SDE noise level	1.0	1.0	1.0
PPO clip ϵ / advantage clip	0.2 / 5	0.2 / 5	0.2 / 5
Reference coefficient β	1e-4	1e-4	1e-4
λ_{vis} / λ_{txt} / λ_c / λ_{task}	1 / 1 / 1 / 4	1 / 0.3 / 1 / 0.5	1 / 1 / 1 / 4
Discount γ	0.7	0.7	0.7

Model and optimizer. LoRA adapters have rank 64 and α = 128, with no dropout, and are attached to the query, key, value, and output projections and to the gate, up, and down MLP projections of both experts. Frozen weights are kept in BF16 and adapter weights in FP32. We use AdamW with no weight decay and separate learning rates for the text and image adapters, clip gradients to norm 1.0, and skip updates with non-finite gradients.

Optimization. Advantages are clipped to $[-5, 5]$, and a group whose returns have standard deviation below 0.02 receives zero advantage, because its rollouts are indistinguishable. The text likelihood ratio is computed for a whole step (the sum of its token log-probabilities) rather than per token, and the text loss is scaled by λ_{txt} as in Equation (13). No KL penalty is applied to text. We accumulate gradients over all steps of a question, normalized by its length, and take one optimizer step per group.

Flow sampling. The schedule has 15 points and therefore 14 integration transitions, three of which form the stochastic window. The start of the window is sampled from the first half of the schedule, where transitions have the most variance. Classifier-free guidance is disabled (scale 1) during RL. At evaluation we decode text greedily and add no SDE noise, using seeded initial latents.

Distributed training. We train with Distributed Data Parallel (DDP) on 8 NVIDIA H200 GPUs. Each GRPO group consists of N rollouts sampled from the same input prompt, ensuring that the group-relative normalization in Equation (9) compares trajectories for the same task instance. Groups are distributed across GPUs during training, while gradients are synchronized across workers through DDP.

D Evaluation Details

Checkpoint selection. Each RL reward condition is trained for 10 epochs, and we report the checkpoint with the best validation score. Validation and test puzzles are disjoint for all three tasks, so no test puzzle influences the choice of checkpoint. We apply this protocol identically to each reward condition, so the comparisons in Tables 2 and 3 differ only in their training signal.

Evaluation subsets. For Rush Hour we reuse the held-out puzzles released with MentisOculi. The MentisOculi paper reports results on 30 puzzles per solution length, but its repository contains 50 per length, and it does not state which 30 were evaluated. We therefore take the first 30 puzzles at each length $L = 1, \ldots, 5$ as our test set (150 puzzles) and the remaining 20 at each length as our validation set (100 puzzles). Every Rush Hour number in Tables 2 and 3 comes from this test set. Our test puzzles thus come from the same released pool as the MentisOculi baselines, but since their selection is unknown we cannot claim the two sets are puzzle-for-puzzle identical. Tangram is evaluated on all 99 test puzzles, with checkpoints chosen on the 19 validation puzzles (Section A.2). Dice rotation uses 160 of its 200 held-out puzzles for testing and the other 40 for validation.

Differences between training and evaluation. On Rush Hour, the training reward checks the state after the last step, whereas the evaluator counts a puzzle as solved if the red car reaches the exit at any step of the rollout. A trajectory that solves the puzzle and then moves a car back into the corridor is therefore rewarded as unsolved during training but counted as solved at evaluation. The evaluation horizon equals the reference length; we do not give the model extra steps. On Tangram, failed generations remain in the denominator with a score of zero. On dice rotation, training accuracy requires every roll direction to be correct, and we additionally report answer-only accuracy.

Limitations of the verifiers. The Rush Hour IoU is computed on color masks and can miss distortions of car shapes or labels, the dice image reward measures pixel fidelity rather than recognizing the drawn symbols, and the Tangram verifier works with finite raster tolerances. The claim parser only checks the statements it recognizes, so claim precision says nothing about the rest of the reasoning text. We therefore pair scalar metrics with qualitative inspection of sampled trajectories.

E Models and Inference details

E.1 Rush Hour

UniCoT-SFT We fine-tuned UniCoT-7B-MoT-v0.2 on 1,000 Rush Hour puzzles using LoRA with rank 64 and $\alpha = 128$. Each dataset cycle included one Macro-CoT planning example and one rotating Micro-CoT next-board generation example per puzzle. Text cross-entropy and image flow-matching losses were equally weighted. Training ran for 1,250 optimizer updates with batch size 16 and trains for 20 epochs. We used a learning rate of 2×10^{-5}, 30 warmup updates, cosine decay to 10^{-7}, and

gradient clipping at 1.0. Training used 512×512 generation images and four A100 GPUs across two nodes with FSDP full sharding.

We evaluated the final UniCoT adapter on 150 held-out Rush Hour puzzles, selecting the first 30 examples from each of five difficulty levels in dataset order. Macro-CoT generated a complete move plan using temperature 0.3 and a maximum length of 1,024 tokens. Moves were extracted from `<subtask>` tags and capped at the reference solution's action count. Micro-CoT then executed the predicted subtasks sequentially, conditioning each image on the previously generated board. Images were generated at 512×512 resolution using 50 flow-matching steps, text/image guidance scales of $4.0/2.0$, and a timestep shift of 3.0. We measured solve rate by executing the predicted textual moves in the Rush Hour simulator; generated images served as visual diagnostics and did not determine success.

E.2 TANGRAM

Gemini. For Gemini models, only image-output models can be evaluated since evaluation is done on generated image: Gemini 2.5-I (`gemini-2.5-flash-image`), Gemini 3.1-I (`gemini-3.1-flash-image`), Gemini 3.1-I Lite (`gemini-3.1-flash-lite-image`), and Gemini 3-I (`gemini-3-pro-image`). Each completed all 99 test figures. We sample at temperature 0.3 with `response_modalities=[TEXT, IMAGE]`; all other decoding parameters are left at the API default, and no system instruction, safety-setting override or tool is configured. Each figure is a single `generate_content` call carrying the prompt text and the two 512×512 question images, the target silhouette first and the piece inventory second, which is the reverse of the order they are stored in. The prompt asks for the completed construction and allows intermediate states, so the number of images is the model's own choice: most responses return one image, and a handful return a full seven-step construction. Every returned image is downsampled to 256×256 with bilinear resampling, and the last one is taken as the submitted solution.

Base BAGEL. We evaluate the frozen `ByteDance-Seed/BAGEL-7B-MoT` checkpoint zero-shot, with no Tangram fine-tuning or adapter. Each rollout contains exactly seven text–image steps, one for each piece; after every text step BAGEL generates an updated construction image, which is included in the history for the following step.

We draw one rollout per puzzle with seed 20260721. Text decoding is greedy, with a maximum of 1536 tokens per step and temperature 0.3. Images are generated at 512×512 resolution with 50 flow steps, text/image CFG scales $3.0/1.5$, and CFG interval $[0.4, 1.0]$. The final generated image is taken as the submitted construction. We report the Tangram construction accuracy from the geometric verifier together with the number of fully solved puzzles; a missing final image is scored as zero.

Qwen2.5-VL. We fine-tune Qwen2.5-VL-7B-Instruct (with a text-only Qwen2.5-7B-Instruct ablation and a larger Qwen2.5-VL-32B run on the Leonardo cluster) on the SVG tangram task: given the seven standard tangram pieces and a target silhouette, the model reconstructs the figure step by step, producing chain-of-thought reasoning before rendering each board state as SVG. Training uses the full set of 895 puzzles, re-rendered from PNG into SVG. We apply QLoRA (4-bit NF4 base, bf16 LoRA adapters, rank 64, $\alpha = 128$, dropout 0.05) to the language-model attention and MLP projections with the vision tower frozen, using paged AdamW 8-bit, a cosine schedule peaking at 5×10^{-5} with 30 warmup steps, effective batch size 16, max sequence length 8192, and 8 epochs, supervising only assistant-turn tokens. The implementation is a custom `transformers/peft` script modeled on the BAGEL-Canvas/ MathCanvas recipe. Evaluation compares generated SVG outputs against ground truth via silhouette IoU, per-step color IoU, piece shape/area matching, an overlap penalty, and format validity.

Wan2.2. We fine-tune Wan2.2-TI2V-5B on the same task with LoRA to learn a purely visual construction trajectory: given a two-panel conditioning image (piece tray and target silhouette) and a text instruction, the model generates a short video assembling the puzzle across seven stages, using each piece exactly once, under rectified-flow velocity prediction rather than an interleaved reasoning objective. Training data is synthesized from the same 895-puzzle dataset, converting each puzzle's seven solution stills into a 121-frame, 1280×704, 24fps clip. We use LoRA rank 64, $\alpha = 128$, dropout 0.05 on the attention and feed-forward projections, a cosine schedule peaking

at 5×10^{-5} with 30 warmup steps, 20 epochs, effective batch size 16 (4 data-parallel A40 GPUs, gradient accumulation 4), text-conditioning dropout 0.1, and a training-time noise-schedule shift of 1.0 (vs. 5.0 at inference), with the VAE and text encoder frozen and precomputed into a cache. The implementation is a custom script built directly against the official Wan2.2 codebase, following the same recipe as the Qwen training. Evaluation samples seven frames per video and scores them with the same silhouette IoU, color IoU, piece matching, and overlap metrics used for the Qwen and Veo comparisons, keeping results comparable across model families.

Veo3.1. We evaluated the pretrained Veo 3.1 model (veo-3.1-generate-preview) without additional training or fine-tuning. Each input consisted of an image showing the seven tangram pieces alongside a target silhouette and a text prompt requesting sequential assembly. The prompt required moving one piece at a time, keeping previously placed pieces fixed, and preserving piece geometry and color. It also required a static camera and background, prohibited additional visual elements, and specified a final assembly without gaps or overlaps.

The generation configuration specified one 8-second video per input at 720p resolution with a 16:9 aspect ratio.

The evaluation compared generated video frames with ground-truth SVG geometry and reference solution renderings. Color-based masks were normalized to 512 × 512 pixels. The evaluator used a color-distance threshold of 90.0, a minimum intersection-over-union of 0.50, a maximum normalized centroid distance of 0.18, and an extra-color ratio threshold of 0.10. It checked for missing or extra color groups, invalid scene changes, and final geometric mismatch. Where direct SVG rasterization did not sufficiently align with the reference rendering, the final reference image provided the comparison masks.

E.3 DICE

GPT-5.1. We query GPT-5.1 (gpt-5.1-2025-11-13) via the OpenAI Responses API, with reasoning effort set to `low`. Temperature and top-p are not applicable, as these sampling parameters are not exposed for reasoning models on this endpoint. Rolling Dice does not use tool access; no code interpreter is enabled.

Gemini. We query Gemini 2.5 Flash (`gemini-2.5-flash`), Gemini 2.5 Pro (`gemini-2.5-pro`), Gemini 3 Flash (`gemini-3-flash-preview`), Gemini 3.1 Pro (`gemini-3.1-pro-preview`), Gemini 3.8 Flash (`gemini-3.8-flash`), Gemini 2.5-I (`gemini-2.5-flash-image`), Gemini 3.1-I (`gemini-3.1-flash-image`), Gemini 3.1-I Lite (`gemini-3.1-flash-lite-image`), and Gemini 3-I (`gemini-3-pro-image`). The task is answered in text, so each puzzle is a single call carrying the prompt, the board image and the net image, and the answer is parsed from the final `ANSWER:` line. The `-I` models are given the same prompt and scored the same way; images they return are stored for inspection but do not enter the score. We sample at temperature 0 and leave each model's own reasoning budget untouched.

Mirage. For the Rolling Dice experiments, we follow the Mirage fine-tuning setup used in MentisOculi (Zeller et al., 2026). We use a latent size of 4 and train each stage for 3 epochs on 1,000 training examples distributed uniformly across the five difficulty levels. Rolling Dice is evaluated using final-answer accuracy: given a sequence of prescribed rotations, the model predicts the final die state as text. Intermediate images are not scored.

Base BAGEL. We evaluate the frozen `ByteDance-Seed/BAGEL-7B-MoT` checkpoint zero-shot (no dice fine-tuning or adapter) on 160-puzzle test split. The rollout contains exactly N text–image steps, where N is the number of rolls on the shown route.

We draw one rollout per puzzle (seed 20260721). Text decoding is greedy (`do_sample=False`) with at most 1536 tokens per roll; image generation uses 512×512 resolution, 50 flow steps, text/image CFG scales $3.0/1.5$, and CFG interval $[0.4, 1.0]$.

Emu3.5. We evaluate `BAAI/Emu3.5` with `BAAI/Emu3.5-VisionTokenizer` zero-shot (no dice fine-tuning) using a single GPU. Each of the 160 test puzzles is given as two images (initial board with

the orange route, and the reference net) plus the dataset question and the BAGEL dice instruction (exactly N rolls, tagged `Roll/<state>/<answer>`, one board image after each roll). The official `howto` template adds "Please generate a response with interleaved text and images."

We draw **one** interleaved sample per puzzle (seed 42). Unlike MentisOculi (up to 15 Emu retries), we do not resample omissions and report the omission rate. Sampling follows the official defaults: text k=200, p=0.8, T=0.7; image k=10240, p=1.0, T=1.0; `do_sample=True`, one beam, 512×512 images. Decoding is free-form; we only halt after N image-end tokens and cap `max_new_tokens` $= \min(4096,\ 512 + 1400N)$.

Headline metric: final-answer accuracy (top face for cubes; bottom face for octahedra), using lenient answer extraction (`<answer>`, `\boxed{}`, or `{"answer": . . . }`). Puzzles with no image count as omitted and as incorrect.

Qwen3-VL-235B-A22B-Thinking. We evaluated Qwen3-VL-235B-A22B-Thinking through Novita AI's serverless, OpenAI-compatible chat completions API using the model ID `qwen/qwen3-vl-235b-a22b-thinking`. Each Dice puzzle was submitted as a single user message containing the dataset question followed by two embedded PNG images: the initial die and roll path, then the reference net. We provided no system prompt, few-shot examples, ground-truth answer, or solution image.

We requested one response per puzzle with temperature 0.8, top-p 0.95, top-k 20, and a maximum of $32{,}768$ completion tokens. No random seed or separate thinking-token budget was specified, and `enable_thinking` was not sent because the selected model was the Thinking variant. The runner stored `reasoning_content` separately from the final `content`; the scorer extracted answers only from the final content. All 160 responses ended with the finish reason `stop`, and the mean completion length was approximately $15{,}395$ tokens.

F Prompt Details

F.1 Rush Hour

Base BAGEL. This prompt is shown to the model at each step of each case together with the initial or generated board-state image, asking the model to produce a move in the required action format. Then the produced images are used for the next step of the puzzle.

Prompt 1: Rush Hour, base BAGEL

```
You are solving a Rush Hour block-sliding puzzle shown in the image.

Goal:
Move the red vehicle labeled R to the pale-green exit.

Rules:
- Every movable colored rectangle has an uppercase letter label and an arrow.
- Black rectangles without letters are fixed obstacles.
- A vehicle can move only along its own dashed movement axis and can never rotate.
- "forward" means the direction indicated by that vehicle's arrow and "backward" means the opposite direction.

- A selected vehicle slides as far as possible in the chosen direction until blocked by another vehicle, an
obstacle, or the board boundary.
- Exactly one vehicle moves in each action.

Inspect the CURRENT board image and choose exactly one legal action that helps R reach the exit.
Use the following response format:
<think>brief reasoning about the current board</think>
X forward

or:
<think>brief reasoning about the current board</think>
X backward

Then generate an image representing the current state of the board after the action you choose is applied.
```

F.2 TANGRAM

Gemini. Shown to the image-generating Gemini models in a single call per puzzle. The model may return intermediate images; the last generated image is evaluated as the submitted construction. Two images are attached in the order named by the prompt: the target silhouette and the inventory of the 7 standard pieces. This is the reverse of the dataset's own field order, where the piece inventory comes first. `[figure]` is the target figure name, substituted identically in all four places. Only models that can return images can be evaluated here; text-only models are excluded by the task rather than by the prompt. The last generated image is taken as the submitted solution and scored against the target silhouette, so the numbers are directly comparable to the BAGEL-Canvas runs. A rollout that returns no image is recorded as a failure and scored zero.

Prompt 2: Tangram, Gemini

```
Attached are two images.
The first image shows the target silhouette to be constructed.
The second image shows the 7 standard tangram pieces available for construction.

Question:
Build a [figure] using the 7 standard tangram pieces.

Rules:
Use all 7 standard tangram pieces exactly once.
Pieces may be translated and rotated. Do not change their size or shape.
Pieces must fit together to form the target figure, with no overlaps.
Preserve the original color of each tangram piece so that all pieces remain distinguishable.
Once a piece is placed, keep it fixed in its position and orientation.

Task:
Construct the [figure] step by step using the 7 tangram pieces.
At each step, place one previously unused piece into the target figure.
You may reason through the construction process and generate intermediate construction states if helpful.
The final result should show the completed [figure] assembled from all 7 pieces.

Output:
Generate the final image showing the completed tangram construction of the [figure].
If intermediate images are generated during reasoning, ensure that the last generated image is the completed
solution.
The final image should clearly match the target silhouette as closely as possible.

[figure] is the target figure name from the dataset, the same word in all four places.
```

Wan2.2 Shown to Wan2.2 together with a two-panel conditioning image containing the colored piece inventory on the left and the target silhouette on the right. The placeholder `<label>` is replaced with the target figure name. The prompt requests a seven-stage construction video, with one piece placed at each stage and all previously placed pieces kept fixed. Seven frames are sampled from the generated video and evaluated using the Tangram geometric metrics.

Prompt 3: Tangram, Wan2.2

```
The reference image contains a two-panel tangram puzzle. The left panel shows the seven available colored
tangram pieces, and the right panel shows the black target silhouette.

Task:
Solve the puzzle by inferring the correct final position and orientation of every piece, then generate a
short step-by-step video constructing a <label>.

Construction rules:
1. Use each of the seven pieces exactly once.
2. Move exactly one previously unused piece at a time from the left panel into the target silhouette in the
right panel.
3. A piece may only be translated and rotated. Do not reflect, resize, deform, split, merge, recolor,
duplicate, or replace any piece.
4. Preserve the exact geometry and original color of every piece.
5. Once a piece reaches its final position and orientation, keep it completely fixed for the remainder of the
 video.
6. Pieces not currently being moved must remain stationary.
7. The placed pieces must not overlap one another or extend outside the target silhouette.

Video requirements:
```

```
- Show exactly seven clearly separated construction stages, evenly spaced through the video.
- Use a fixed top-down view.
- Keep the white background, two-panel layout, camera position, framing, and scale unchanged.
- Do not add hands, text, labels, arrows, shadows, textures, gradients, or extra objects.

Final state:
By the final frame, all seven colored pieces must be in the right panel and must tile the black target
silhouette completely, with no gaps or overlaps. The completed arrangement's outer boundary must match the
target silhouette as closely as possible. Keep the completed solution motionless and clearly visible through
the end of the video.
```

Veo 3.1. The model receives a two-panel conditioning image showing the seven tangram pieces on the left and the target silhouette on the right. The placeholder `{whole_label}` is replaced with the target figure name, and `{num_steps}` with the number of reference construction steps.

Prompt 4: Tangram, Veo 3.1

```
Look at this image:
The left panel shows the 7 standard tangram pieces. The right panel shows the target silhouette to build: "{
whole_label}".

Rules:
1. Use exactly the 7 visible tangram pieces: two large triangles, one medium triangle, two small triangles,
one square, and one parallelogram.
2. Place exactly one piece per action. Once a piece is placed, it must stay fixed for the rest of the video.
3. Pieces may translate, rotate, or flip only as needed to match the target. They must never scale, stretch,
bend, morph, split, merge, or change color.
4. Do not add or remove anything: no new objects, labels, arrows, lights, shadows, reflections, textures,
markings, text, hands, tools, or UI elements.
5. No camera motion: no zoom, no pan, no rotate, no tilt, no dolly, no perspective shift.
6. The white background and target silhouette remain perfectly static.
7. Preserve the crisp flat vector style and dark outlines from the input image.
8. The final frame must show all 7 pieces assembled to fill the target silhouette with no gaps or overlaps.

Task:
Plan and demonstrate the full tangram construction. Move one tangram piece at a time from the starting set
into the target silhouette until the target is complete.

Output:
A video demonstrating the full solution, one piece placement at a time. The video should contain {num_steps}
clear placement actions.

<image combined.png>
```

Base BAGEL. The model is given the two Tangram input images: the piece inventory first and the target silhouette second. It is then asked to solve the puzzle in 7 steps. At each step, the model describes one piece placement and generates an updated image of the Tangram. That generated image is used as the input for the next step.

Prompt 5: Tangram, base BAGEL

```
Build a [figure] using the 7 standard tangram pieces shown in the first image.
The target silhouette is shown in the second image.

Solve the puzzle in exactly 7 steps.
At each step:
- place one unused tangram piece;
- briefly describe that placement;
- generate one updated image showing the current construction.

Use each of the 7 pieces exactly once.
Keep the piece colors, sizes, and shapes unchanged.
The final generated image should be the completed tangram and should match the target silhouette as closely
as possible.
```

F.3 ROLLING DICE

Emu 3.5. Shown to Emu 3.5 in the free-form interleaved (image-generation) setting. Two images are attached, in order: the rendered initial board with the orange path, and the reference net. The first sentence is puzzle-specific; the four stems are "what number will be on the top face at the end?" (cube, numbered/pips), "which symbol will be on the top face at the end?" (cube, symbols), "what number will be face down at the end?" (octahedron, numbered), and "which symbol will be face down at the end?" (octahedron, symbols). N is the number of rolls on the shown path. The JSON example below is the cube schema; octahedron puzzles instead receive `{"bottom": 1, "opposite_the_bottom": 8, "visible_sides": [2, 3, 4], "hidden_sides": [5, 6, 7]}` (or the same keys with lowercase symbol names). Emu3.5's chat template additionally prefixes "You are a helpful assistant for howto task. Please generate a response with interleaved text and images."

Prompt 6: Rolling Dice, Emu 3.5

```
If the die is rolled along the shown path, what number will be on the top face at the end?
The first image shows the initial state and roll path; the second image is the reference net. Reason through
each roll.
Follow the shown route for exactly $N$ rolls, one roll per reasoning/image step. After each roll write your
reasoning inside think tags, then a separate Roll line giving the on-screen direction, then one JSON object
inside state tags describing the POST-roll die. Use the keys and value types in this hypothetical format
example. Its values are NOT this puzzle's state; replace every value with your own result:
<state>{"top": 1, "bottom": 6, "up-left": 2, "down-right": 5, "up-right": 3, "down-left": 4}</state>
For example, a down-left action line is: Roll down-left. Choose the action from this puzzle's route, not from
 the example. Cube side keys name on-screen directions and each has ONE value, not a list. For octahedra
visible_sides and hidden_sides each contain three values. Use integers for numbered/pip dice and lowercase
symbol names for symbol dice. Draw the resulting board after each step. On the LAST roll only, put the
requested final number or symbol between answer tags before the image. Do not give alternative states or copy
 placeholder words.
Please provide exactly $N$ reasoning steps. After each roll generate one image of the updated board. End with
 the final answer in <answer>VALUE</answer> or {"answer": "VALUE"}.
```

Gemini. Shown to the Gemini models as a single call per puzzle: the prompt text, the rendered initial board with the orange path, and the reference net, in that order. The question line is taken verbatim from the dataset; the four stems are "what number will be on the top face at the end?" (cube, pips), "which symbol will be on the top face at the end?" (cube, symbols), "what number will be face down at the end?" (octahedron, digits), and "which symbol will be face down at the end?" (octahedron, symbols). Square brackets mark the parts that vary with the subset: two-way choices are cube versus octahedron, four-way choices follow the subset order above, and the two three-way choices are the marking (pips, symbols, digits) and the opposite-face rule, which differs between the cube net, the digit octahedra (faces summing to 9), and the symbol octahedra. Listing the answer space on the last line fixes the vocabulary so that a symbol named "crescent" rather than "moon" is not scored as a miss; it does not reveal which face carries which marking, which is readable only from the net. No system prompt is used, and the solution fields of the dataset — the per-roll states, the step images, and the answer — are never sent, since an octahedron's final top face would give away its bottom through the opposite-face pairing. The answer is parsed from the final `ANSWER:` line, falling back to the last token of the answer space that appears anywhere in the response, and scored by exact match.

Prompt 7: Rolling Dice, Gemini Models

```
Attached are two images.
The first image shows the board with the die in its starting position and the path to roll it along.
The second image shows the net of the same die, laid flat with all [6 | 8] faces visible.

Question:
If the die is rolled along the shown path, [what number will be on the top face | which symbol will be on the
 top face | what number will be face down | which symbol will be face down] at the end?

Rules:
The die is [a cube with 6 square | an octahedron with 8 triangular] faces, marked with [pips (dot patterns
standing for the numbers 1 to 6) | symbols | digits].
Roll it one cell at a time along the path shown, starting from the position in the first image.
[ Each roll tips the die over one edge onto the next cell, so the leading face goes to the bottom, the old
bottom face comes up on the trailing side, and the old top face moves to the leading side. The two faces
```

```
perpendicular to the roll stay where they are. | The die rests on one triangular face, and each roll tips it
over an edge of that face onto the neighbouring triangle, which becomes the new face that is down. ]
[ On a cube net, opposite faces are the pairs that never share an edge once the net is folded. | Each face
has exactly one opposite face, and opposite faces always sum to 9. | Each face has exactly one opposite face,
 and the face that is down and the face directly opposite it are always such a pair. ]
Read the pairing of opposite faces off the net.
The markings stay fixed to their faces. They may appear rotated or upside down after a roll, which does not
change which face they are on.

Task:
Work out the starting orientation of the die step by step from the first image and the net.
At each step, roll the die once along the path and state which marking is on [the top face and which is on
the bottom | the face that is down].
You may reason through the rolls and generate intermediate images, if helpful.
The final result should be the marking on [the top face | the face that is down] after the last roll in the
path.

Output:
End your response with a single final line in exactly this form:
ANSWER: <[number | symbol]>
Give only the single [number | symbol name], one of [1, 2, 3, 4, 5, 6 | arrow, cross, heart, moon, star,
triangle | 1, 2, 3, 4, 5, 6, 7, 8 | arrow, circle, cross, heart, moon, square, star, triangle] on that line,
with no extra words.
```

GPT5.1 Shown to GPT-5.1 together with the dataset question and both input images: the initial board with the roll path and the reference die net. The model is asked to reason internally and return only a JSON object containing the final answer, without intermediate images or an explicit reasoning trace. The requested answer is the final top face for cubes and the final bottom face for octahedra, expressed as a digit or a lowercase symbol name. We use low reasoning effort with no tool access.

Prompt 8: Rolling Dice, GPT5.1

```
Solve the dice rotation puzzle using the question and both images. Reason through the rolls internally.
Return only one JSON object with exactly this format: {"answer":"value"}. For a number, use a single digit as
 a string. For a symbol, use one lowercase English word. Do not include explanations, Markdown, or any other
text outside the JSON object.
```

G Additional Unified Backbone Results

To assess whether GroundedVCoT depends on the choice of unified model, we repeat the Rush Hour experiment with SenseNova-U1. For each backbone, we compare its supervised checkpoint with the checkpoint obtained by applying GroundedVCoT. The task data, verifier, reward definitions, evaluation set, and inference budget are held fixed. The pretrained backbone and its resulting supervised checkpoint differ. As shown in Table A.3, GroundedVCoT improves overall solve rate from 48.0% to 71.3% with BAGEL and from 51.3% to 67.3% with SenseNova-U1. Both backbones improve most at intermediate and higher difficulty levels, although SenseNova-U1 remains below BAGEL-GroundedVCoT at L3–L5. These results show that the training procedure can improve a second unified backbone under the same Rush Hour protocol. They do not establish backbone-independent gains across tasks.

Table A.3: **GroundedVCoT transfers across unified backbones.** Rush Hour solve rate overall and by difficulty level, with 30 puzzles per level. The tasks, verifiers, reward definitions, evaluation set, and inference budget are identical for both backbones, only the pretrained unified model changes.

Backbone / training	Avg.	L1	L2	L3	L4	L5
BAGEL						
SFT	0.480	1.000	0.967	0.400	0.033	0.000
+ GroundedVCoT	**0.713**	**1.000**	**1.000**	**0.833**	**0.567**	**0.167**
SenseNova-U1						
SFT	0.513	1.000	0.900	0.400	0.267	0.000
+ GroundedVCoT	**0.673**	**1.000**	**1.000**	**0.767**	**0.467**	**0.133**

H QUALITATIVE RESULTS

In Figures A.4-A.9, we show successful and unsuccessful GroundedVCoT rollouts for each task. Each example includes the input, the model's textual reasoning and actions, its generated intermediate images, and the final outcome. These examples complement the aggregate results by showing whether a failure begins in the selected action, the generated visual state, or the agreement between text and image. They are illustrative rather than a measure of the frequency of each error type.



Figure A.4: **High-scoring Rush Hour rollouts.**

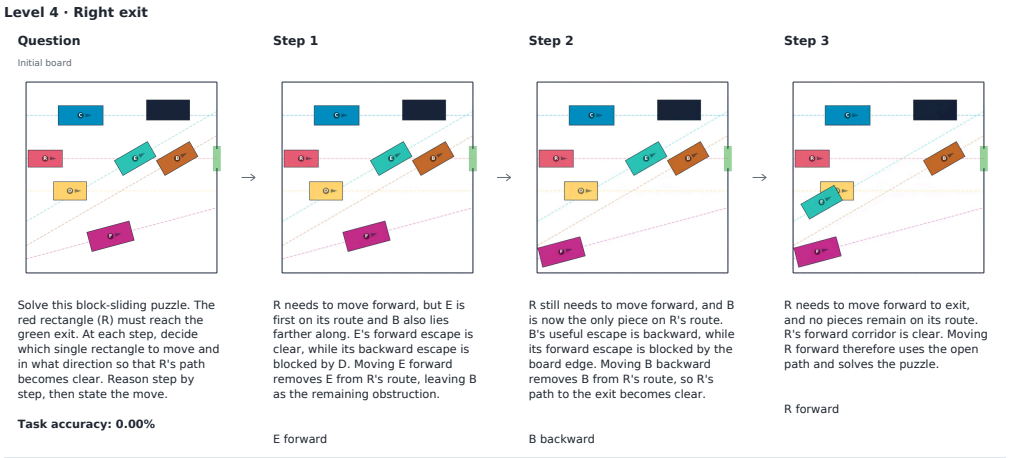


Figure A.5: **Unsuccessful Rush Hour rollouts.**

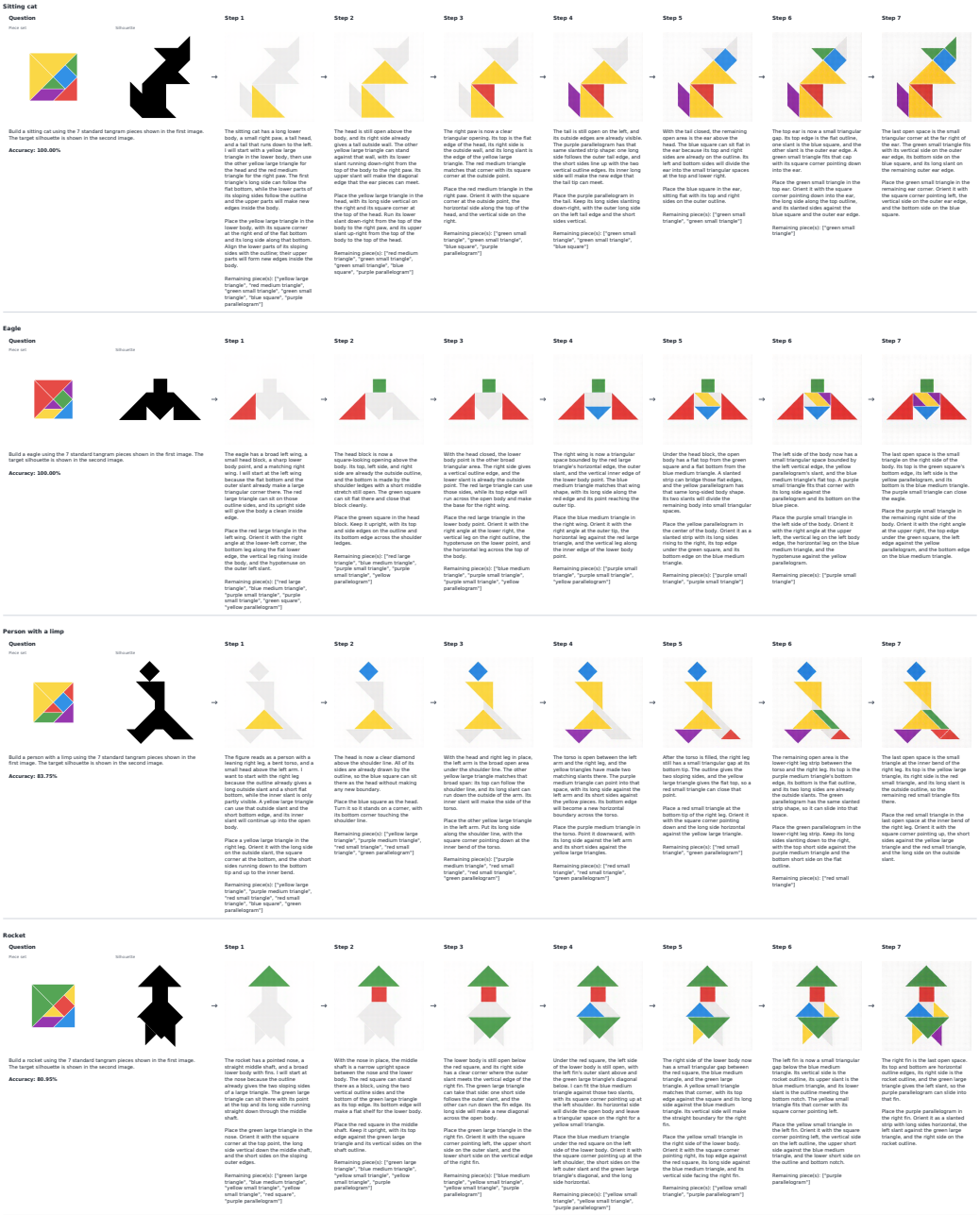


Figure A.6: **High-scoring Tangram rollouts.**

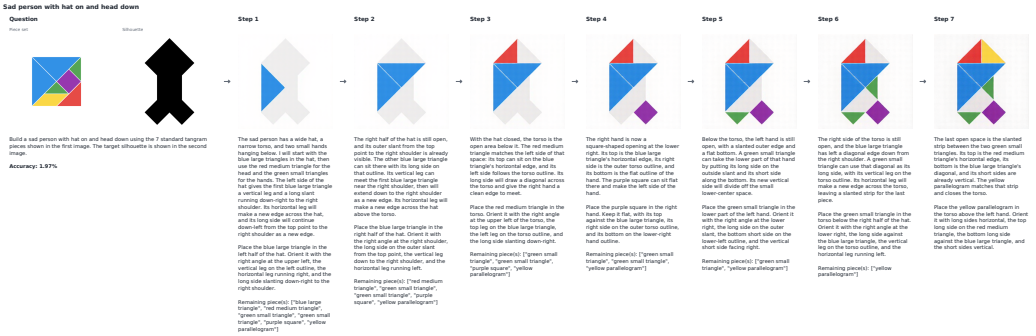
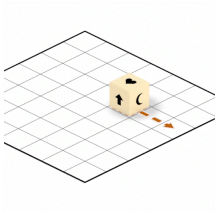


Figure A.7: **Unsuccessful Tangram rollouts.**

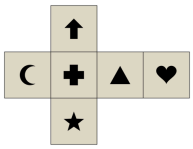
Cubic die · symbols

Question

Initial state and path



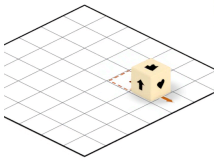
Reference net



If the die is rolled along the shown path, which symbol will be on the top face at the end?
The first image shows the initial state and roll path; the second image is the reference net. Reason through each roll.

Task accuracy: 100.00%

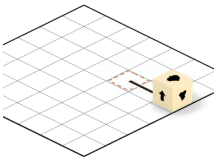
Step 1



Rolling down-right tips moon from the down-right side onto the bottom, lifting triangle from the opposite up-left side to the top. Heart moves from the top to the down-right side, while cross swings from the bottom to the up-left side. Arrow remains on the down-left side, and star remains on the up-right side.

Roll down-right.

Step 2



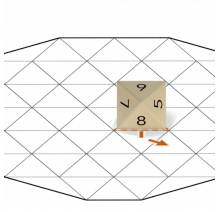
The down-right roll sends heart from the down-right side to the bottom, raising cross from the up-left side to the top. Triangle rotates from the top to the down-right side, and moon moves from the bottom to the up-left side. The perpendicular sides stay fixed, leaving arrow on the down-left side and star on the up-right side.

Roll down-right.
The symbol on the top face at the end is cross.
cross

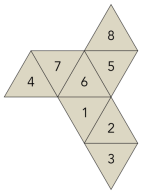
Octahedral die · digits

Question

Initial state and path



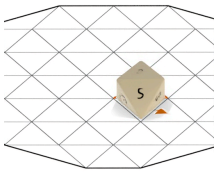
Reference net



If the die is rolled along the shown path, what number will be face down at the end?
The first image shows the initial state and roll path; the second image is the reference net. Reason through each roll.

Task accuracy: 100.00%

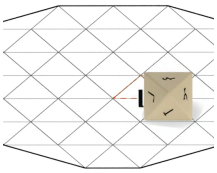
Step 1



Rolling down tips the die over the edge shared by bottom face 1 and visible face 8, bringing face 8 onto the board. The new bottom is 8, and the face now opposite it is 1, since opposite faces sum to 9.

Roll down.
The number face down at the end is 8.
8

Step 2



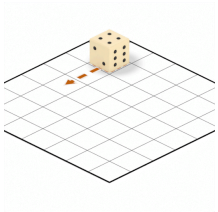
The down-right roll lowers visible face 5 across the rolling edge and onto the board. Face 5 becomes the new bottom, with face 4 now opposite it because 5 + 4 = 9.

Roll down-right.
The number face down at the end is 5.
5

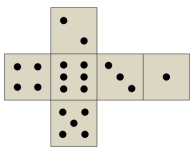
Cubic die · pips

Question

Initial state and path



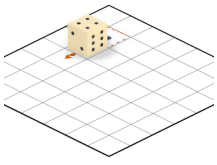
Reference net



If the die is rolled along the shown path, what number will be on the top face at the end?
The first image shows the initial state and roll path; the second image is the reference net. Reason through each roll.

Task accuracy: 100.00%

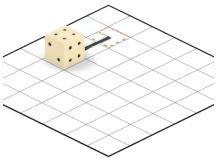
Step 1



The starting top is 4, opposite the bottom face, 3. Rolling down-left tips the down-left face, 2, onto the bottom and brings the opposite up-right face, 5, to the top. The old top, 4, moves to the down-left side, while the old bottom, 3, moves to the up-right side. The down-right and up-left sides remain at 6 and 1, respectively.

Roll down-left.
The number on the top face at the end will be 5.
5

Step 2



The down-left roll sends the current down-left face, 4, to the bottom, lifting the up-right face, 3, to the top. The previous top, 5, rotates to the down-left side, and the previous bottom, 2, swings to the up-right side. The up-left and down-right sides stay at 1 and 6.

Roll down-left.
The number on the top face at the end will be 3.
3

Figure A.8: **High-scoring Rolling Dice rollouts.**

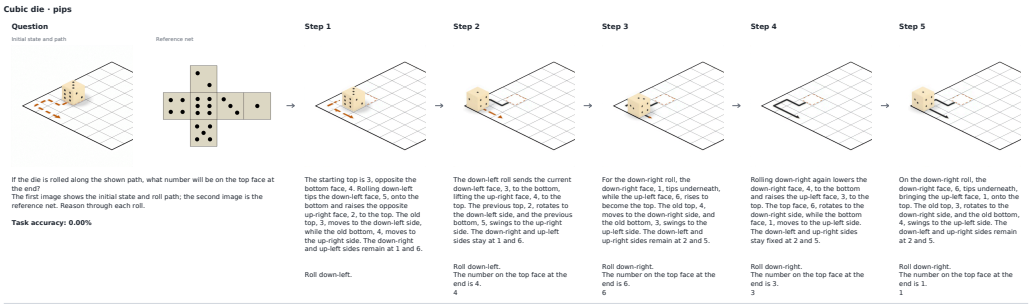


Figure A.9: **Unsuccessful Rolling Dice rollouts.**