LEARNING TO ZOOM AT THE RIGHT DIFFICULTY: DYNAMIC CURRICULA FOR VISUAL TOOL-USE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Crop-and-zoom tools enable vision-language models to acquire fine-grained visual evidence and answer questions that require visual detail beyond their native input resolution. Prior work shows that reinforcement learning on tasks solvable without visual tools can induce overfitting rather than improve tool use capability, motivating training on hard tool-essential data. However, training on challenging datasets can be highly inefficient under Group Relative Policy Optimization (GRPO), as filtering out numerous uninformative samples with all-zero rewards requires substantial additional rollout generation. This overhead is especially high when visual tool use involves multiple rounds of generation, tool execution, and visual feedback. We propose *Calibrated Dynamic Difficulty Shaping (CalDS)*, an adaptive sampling framework that makes efficient use of hard examples by matching training-batch difficulty to the model's evolving capabilities. CalDS organizes examples into dynamically updated buckets based on tool-augmented accuracy. It uses a bounded reward-feedback controller with difficulty shaping to adjust sampling probabilities based on bucket difficulty and reward history, emphasizing challenging but currently learnable examples. To account for stale difficulty estimates as the policy evolves, CalDS reuses recent rollouts to calibrate bucket difficulty estimation. On VisualProbe, a challenging high-resolution VQA benchmark, CalDS improves both answer accuracy and crop localization accuracy at a much lower training cost. These results highlight adaptive difficulty control as a means of learning effective visual tool use efficiently from hard training data.

1 INTRODUCTION

Visual tool use enables vision-language models (VLMs) to invoke visual operations, such as cropping and zooming, during reasoning. These operations allow models to inspect selected image regions in greater detail and access visual evidence that may be difficult to perceive from an initial global view. Recent work has increasingly explored reinforcement learning to train models to interleave visual operations with reasoning, using high-resolution images and questions that require detailed visual inspection. These approaches aim to teach models not only to invoke tools but also to identify informative regions and integrate the resulting observations into their reasoning. Representative methods (Wu & Xie, 2024; Su et al., 2026; Zheng et al., 2026; Lai et al., 2026) demonstrate the promise of this paradigm for improving visual understanding.

However, improved task accuracy does not necessarily imply better visual tool use. Recent studies (Ma et al., 2026b; Shao et al., 2026; Guo et al., 2026) find that performance gains can arise from fitting the data distribution and adapting to the tool-use schema rather than from enhanced tool-use capabilities. Complementary research (Liu et al., 2025; Wang et al., 2025a; Hou et al., 2026; Zhao et al., 2026b) reports that models can remain insensitive to the content of cropped regions. Together, these findings suggest that existing visual tool-use models do not necessarily rely on the visual evidence returned by tools to solve a task.

We define *Visual Tool Necessity (VTN)* as the extent to which a task requires a given model to use visual tools: a task has low VTN if the model can easily solve it without tools. To examine this property in existing training data, we evaluate without-tool accuracy on the training set used by

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.

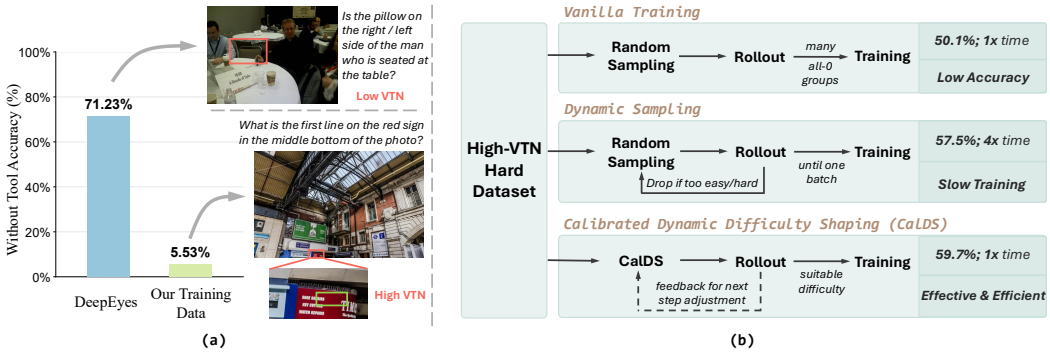


Figure 1: (a) The training dataset we use has substantially lower without-tool accuracy than the data used by DeepEyes (Zheng et al., 2026), with examples requiring finer-grained visual inspection. (b) On a hard dataset, vanilla GRPO produces many all-incorrect rollout groups, providing no learning signal. Dynamic sampling addresses this through post-rollout filtering, but repeatedly generating replacements makes training inefficient. Our CalDS maintains dynamic buckets and uses rollout feedback to adapt the sampling distribution, achieving high accuracy with efficient training.

DeepEyes (Zheng et al., 2026). As shown in Figure 1 (a), the accuracy without tools is already high, indicating low VTN. Such examples provide limited incentive to acquire and use additional visual evidence. Ideally, high-VTN training data should be important for developing visual tool-use capabilities. We therefore construct a high-VTN dataset, which has substantially lower without-tool accuracy, for visual tool-use training. This raises our central question: ***How can we effectively and efficiently train models to use visual tools on challenging, high-VTN data?***

Directly applying Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to this dataset is ineffective. As shown in Figure 1 (b), many rollout groups contain only incorrect answers and provide no relative reward signal for reinforcing successful behavior. Dynamic sampling, which repeatedly generates rollouts and retains groups containing both correct and incorrect responses, is commonly used to address this challenge (Yu et al., 2026; Feng et al., 2026). Although this maintains informative batches, rollout costs can be substantial when few groups qualify on challenging high-VTN data. This cost is particularly consequential for visual tool use, where trajectories may involve multiple rounds of generation, tool execution, and visual feedback.

We propose *Calibrated Dynamic Difficulty Shaping (CalDS)*, an adaptive sampling framework that adjusts training-batch difficulty to match the model's evolving capabilities. CalDS organizes training examples into multiple difficulty buckets based on their tool-augmented accuracy and continuously updates the difficulty estimates during training. This allows CalDS to control the difficulty of upcoming batches by adjusting the sampling probabilities over buckets. (1) To determine the sampling distribution, CalDS combines current bucket difficulty estimates with historical reward feedback through a bounded reward-feedback controller, emphasizing challenging examples that are currently learnable. (2) A further challenge is that difficulty estimates become stale as the policy evolves: estimates accumulated from earlier rollouts no longer reflect the model's current capabilities. CalDS addresses this mismatch by reusing readily available recent rollouts to calibrate bucket difficulty estimates. Together, these components enable CalDS to align batch allocation with the model's changing capabilities, obtaining informative rollouts effectively and efficiently to turn successful tool-assisted trajectories into improved visual tool-use behavior.

2 Methodology

We introduce *Calibrated Dynamic Difficulty Shaping (CalDS)* for effective and efficient GRPO training with crop-and-zoom tools. CalDS uses the outcomes of tool-augmented rollouts to update both the policy and the difficulty composition of subsequent training batches, as illustrated in Figure 2.

2.1 Preliminary

Crop-and-zoom interaction. Given an image question pair (I_i, Q_i), the policy π_θ observes a resolution-limited global image and generates an interleaved sequence of reasoning, tool calls, and

a final answer, following the paradigm of Zheng et al. (2026); Lai et al. (2026); Ma et al. (2026b). A crop-and-zoom call specifies a bounding box $[x_1, y_1, x_2, y_2]$, and the tool extracts the specified region from the original full-resolution image and returns it as a new visual observation, providing local details that may be lost in the resolution-limited global view. The policy conditions subsequent generation on the returned image crops, allowing it to inspect region details before answering. We train models on a challenging high-VTN dataset selected from examples that are difficult for the base model to answer. The data construction procedure is described in the implementation details.

Outcome reward and policy optimization. At iteration t, we select B distinct prompts $\mathcal{D}_t$ and generate G trajectories per prompt with π_{θ_t}. A judge model assigns each trajectory a binary reward $r_{i,g,t} \in \{0, 1\}$ by comparing its final answer with the reference. Since the training data lack annotations of the image regions supporting the correct answers, we only use answer correctness reward. We train with GRPO (Shao et al., 2024), obtaining a group-relative advantage $\hat{A}_{i,g,t}$ by comparing each reward with the other outcomes for the same prompt. Let ℓ index token positions in the g-th trajectory for prompt i. For response token $a_{i,g,\ell}$ with its context $h_{i,g,\ell}$, define $\rho_{i,g,\ell}(\theta) = \pi_\theta(a_{i,g,\ell} \mid h_{i,g,\ell})/\pi_{\theta_t}(a_{i,g,\ell} \mid h_{i,g,\ell})$. We maximize the clipped objective

$$\mathcal{J}(\theta) = \frac{1}{\sum_{i,g,\ell} m_{i,g,\ell}} \sum_{i,g,\ell} m_{i,g,\ell} \min\Big[\rho_{i,g,\ell}(\theta)\hat{A}_{i,g,t},\ \mathrm{clip}(\rho_{i,g,\ell}(\theta), 1-\epsilon_{\mathrm{low}}, 1+\epsilon_{\mathrm{high}})\,\hat{A}_{i,g,t}\Big], \quad (1)$$

where $m_{i,g,\ell}$ masks valid policy-generated response tokens, excluding tool observations. The clipping parameters ϵ_{low} and ϵ_{high} specify the allowable deviations of the probability ratio below and above 1. This objective encourages the policy to assign higher probability to trajectories with above-average answer rewards and lower probability to below-average ones. However, when the dataset consists predominantly of hard examples, many rollout groups receive all-zero rewards and provide no relative reward signal, making policy optimization inefficient.

2.2 Dynamic Buckets and Reward-Guided Sampling

The group relative reward property makes GRPO training inefficient on hard high-VTN datasets. Dynamic sampling addresses this by filtering out groups with identical rewards after generation and rollout replacements, but this can require many additional tool-augmented rollouts. We instead seek to adjust the difficulty composition of each batch before generation, so that a fixed rollout budget is more likely to yield useful relative reward signals.

What difficulty should we target? Informative GRPO groups require both successes and failures. For G independent Bernoulli outcomes with a success probability p, the probability of such a mixed group is $1-p^G-(1-p)^G$, maximized at $p = 0.5$. This motivates choosing a target training success rate $\tau = 0.5$. Because real per-example success probabilities are unknown, we use observed batch reward as a coarse feedback signal to adjust the difficulty mixture. However, a batch mean of 0.5 can also arise from a mixture of all-one and all-zero groups, so our sampler must seek not only to match this target but also to favor prompts that yield both correct and incorrect trajectories.

Building a difficulty estimation that evolves with the policy. To achieve such a mixture, we first need to distinguish examples by how difficult they are for the policy model. A fixed ranking from the base model would become outdated during training, while reevaluating the entire dataset at every iteration would introduce numerous additional cost. We therefore estimate difficulty from recent training outcomes and organize examples into buckets that can change membership over time.

For each example i, we maintain an ordered history $H_{i,t}$ of its latest X binary rollout outcomes collected before batch t. We initialize $H_{i,0}$ with tool-enabled base-model rollouts for the training dataset. The historical success rate and bucket assignment are

$$\hat{p}_{i,t} = \frac{1}{|H_{i,t}|} \sum_{r \in H_{i,t}} r, \qquad b_{i,t} = \min(K-1, \lfloor K\hat{p}_{i,t} \rfloor), \quad (2)$$

where K is the number of buckets, ordered from hardest to easiest. Write $\mathcal{B}_{b,t} = \{i : b_{i,t} = b\}$, $n_{b,t} = |\mathcal{B}_{b,t}|$, and $\mathcal{A}_t = \{b : n_{b,t} > 0\}$. After generating G trajectories for a selected prompt i, we form $H_{i,t+1}$ by appending their rewards to $H_{i,t}$ and retaining the most recent X outcomes. We then recompute its bucket using Equation (2); unselected examples retain their histories and assignments. This lets the difficulty representation evolve using the same rollouts of policy optimization.

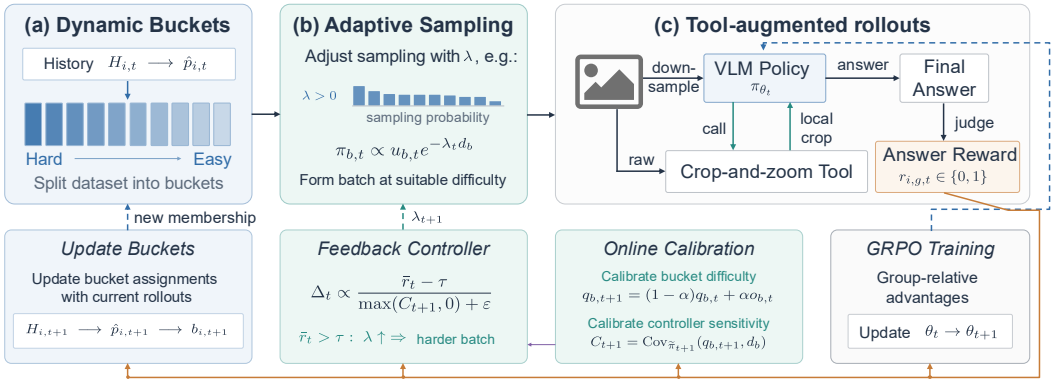


Figure 2: **Overview of CalDS.** Tool-augmented accuracies determine dynamic buckets. A feedback-controlled sampling strategy forms a batch at an adaptive difficulty. The policy interleaves reasoning with crop-and-zoom calls, receiving binary rewards. These rollouts drive GRPO, refresh sample histories, and perform calibration, and provide feedback to adjust next batch's difficulty.

Turning difficulty estimates into a controllable batch mixture. The bucket strategy allow us to control batch difficulty by adjusting how many prompts we draw from each bucket. An intuitive starting point is a uniform reference distribution $u_{b,t} = 1/|\mathcal{A}_t|$ over nonempty buckets. Ideally, sampling uniformly across buckets would yield an expected batch reward near 0.5. In practice, the policy evolves continuously while bucket assignments reflect historical outcomes, so the mean reward will drift. We therefore use reward feedback from preceding batches to shift probability toward harder or easier buckets, keeping the mean near our target τ.

We represent bucket b by a fixed score $d_b \in [0, 1]$, increasing from hard to easy. It is a monotone transformation of the bucket's success-rate interval center, specified in Section 2.3. Then we use a scalar λ_t that controls the sampling preference: positive values favor harder buckets, negative values favor easier ones, and zero recovers uniform sampling. Define the bucket sampling distribution:

$$\pi_t(\lambda_t) = \arg\min_{w \in \Delta(\mathcal{A}_t)} \left\{ D_{\mathrm{KL}}(w \| u_t) + \lambda_t \sum_{b \in \mathcal{A}_t} w_b d_b \right\}, \tag{3}$$

where w_b is a candidate probability for bucket b, and $\Delta(\mathcal{A}_t)$ is the probability simplex over nonempty buckets. The KL divergence $D_{\mathrm{KL}}(w\|u_t) = \sum_{b \in \mathcal{A}_t} w_b \log(w_b/u_{b,t})$ penalizes departure from the uniform reference u_t. The weighted mean score $\sum_b w_b d_b$ is smaller for harder mixtures, so λ_t sets the direction and strength of the preference. Using a Lagrange multiplier for $\sum_b w_b = 1$, the first-order conditions give $w_b \propto u_{b,t} \exp(-\lambda_t d_b)$. Normalizing these weights yields:

$$\pi_{b,t}(\lambda_t) = \frac{u_{b,t} \exp(-\lambda_t d_b)}{\sum_{c \in \mathcal{A}_t} u_{c,t} \exp(-\lambda_t d_c)}, \qquad b \in \mathcal{A}_t. \tag{4}$$

Here, c indexes nonempty buckets; we initialize $\lambda_0 = 0$ for uniform sampling. To see how λ_t changes the relative sampling probabilities, consider two nonempty buckets b and c. Taking the ratio in Equation (4) cancels the shared denominator and equal reference weights; taking its logarithm then gives $\log[\pi_{b,t}(\lambda_t)/\pi_{c,t}(\lambda_t)] = -\lambda_t(d_b - d_c)$. If b is harder than c, then $d_b < d_c$, so increasing λ_t raises the probability of sampling b relative to c. Detailed analysis is provided in Appendix D.

Adapting the mixture from observed rewards. Having defined how λ changes the sampling distribution, we can use the reward from batch t to determine the value of λ_{t+1} for batch $t+1$. The observed mean reward is $\bar{r}_t = \frac{1}{BG} \sum_{i \in \mathcal{D}_t} \sum_{g=1}^{G} r_{i,g,t}$. When $\bar{r}_t > \tau$, we increase λ to introduce harder examples, and vice versa. The size of this correction should also account for how strongly a change in λ affects expected reward. After updating bucket membership and accuracy forecasts, we estimate this sensitivity as C_{t+1}, derived in Section 2.3, and then compute λ_{t+1}

$$\Delta_t = \mathrm{clip}_{[-\Delta_{\max}, \Delta_{\max}]} \left(\frac{\eta_\lambda (\bar{r}_t - \tau)}{\max(C_{t+1}, 0) + \varepsilon} \right), \quad \lambda_{t+1} = \mathrm{clip}_{[-\lambda_{\max}, \lambda_{\max}]}(\lambda_t + \Delta_t). \tag{5}$$

Sensitivity normalization (the denominator) reduces the correction when reward is more sensitive to λ, and vice versa. The update rate η_λ scales this correction, while $\varepsilon > 0$ prevents division by zero.

To avoid overly large updates, clipping limits the correction magnitude to $\Delta_{\max}$ and the resulting control magnitude to $\lambda_{\max}$. Equation (4) then gives $\pi_{b,t+1}$ using the updated active set and λ_{t+1}. Each training iteration in the feedback loop requires exactly BG trajectories, without the repeated rollout filtering used in dynamic sampling (Yu et al., 2026). As shown in Table 3, this provides high performance with substantially lower rollout cost than dynamic sampling.

2.3 Online Calibration and Difficulty Shaping

Calibrating bucket accuracy. Historical success rates provide a basis for grouping examples, but may lag behind the current policy because examples are revisited at different times. We therefore maintain a separate forecast $q_{b,t}$ of rollout accuracy for examples selected from bucket b. It is initialized to the mean historical success rate of that bucket's members. For every sampled bucket, we estimate its observed accuracy and update the forecast by an exponential moving average:

$$o_{b,t} = \frac{1}{Ga_{b,t}} \sum_{i \in S_{b,t}} \sum_{g=1}^{G} r_{i,g,t}, \qquad q_{b,t+1} = (1-\alpha) q_{b,t} + \alpha o_{b,t}. \tag{6}$$

Here, $\alpha \in (0, 1]$ controls how quickly forecasts incorporate new observations. The set $S_{b,t}$ refers to the source bucket recorded before rollout. Thus, an example migrating from bucket b to b' contributes its current outcomes to q_b. This attribution matches the forecast to the population from which the outcomes were requested and avoids using those outcomes to redefine their own calibration group. The sample histories continue to determine membership through Equation (2).

Center-preserving difficulty shaping. The bucket sampling probabilities derived in Equation (4) simply tilt the overall distribution toward harder or easier examples. When training proceeds as expected, examples tend to migrate toward easier buckets. Large differences in sampling probability along this path can turn less frequently sampled buckets into bottlenecks, where examples will accumulate. To allow smoother migration, we assign relative close sampling scores to intermediate buckets, particularly those near a success rate of 0.5, keeping their sampling probabilities comparable as the overall difficulty mixture changes. Let $c_b = (b + 0.5)/K$ be the center of bucket b's success-rate interval. We define its fixed sampling score d_b using a shaping parameter γ:

$$d_b = \frac{1 + (2c_b - 1)^{\gamma}}{2}, \qquad c_b = \frac{b + 0.5}{K}. \tag{7}$$

Choosing γ as an odd integer greater than one preserves the hard-to-easy ordering and symmetry around 0.5, while compressing score differences near the center. With similar scores, intermediate buckets maintain comparable sampling probabilities even as increasing λ_t shifts sampling toward harder buckets, incentivizing smoother migration of examples toward easier buckets.

Calibrated controller sensitivity. The reward error $\bar{r}_t - \tau$ tells the controller whether to make the next batch harder or easier. To choose how much to adjust λ, we also need to estimate how much the adjustment will change expected reward. The fixed scores d_b determine how sampling probabilities change with λ, and the calibrated forecasts $q_{b,t}$ estimate the success rate of the examples drawn from each bucket. Together, they give the predicted batch reward $m_t(\lambda) = \sum_{b \in \mathcal{A}_t} \pi_{b,t}(\lambda) q_{b,t}$. Its derivative measures how predicted reward responds to a small change in λ:

$$\frac{\mathrm{d}m_t(\lambda)}{\mathrm{d}\lambda} = \sum_{b \in \mathcal{A}_t} \pi_{b,t}(\lambda) q_{b,t} \big(\bar{d}_t(\lambda) - d_b\big) = -\operatorname{Cov}_{b \sim \pi_t(\lambda)}(q_{b,t}, d_b). \tag{8}$$

Here, Cov denotes the covariance between predicted accuracy $q_{b,t}$ and sampling score d_b. A positive covariance means that buckets with higher scores tend to have higher predicted accuracies. Thus, a larger positive covariance means a larger derivative magnitude: the same small change in λ produces a larger change in predicted reward. A smaller adjustment to λ is therefore sufficient to correct a given reward error. This explains why Equation (5) divides the reward error by the estimated sensitivity. More derivation detail and analysis is provided in Appendix E.

After batch t, we first update the bucket forecasts and membership using the latest rollout outcomes. Keeping λ_t fixed, we then recompute the sampling distribution and estimate the sensitivity:

$$\widetilde{\pi}_{b,t+1} = \pi_{b,t+1}(\lambda_t), \qquad C_{t+1} = \operatorname{Cov}_{b \sim \widetilde{\pi}_{t+1}}(q_{b,t+1}, d_b). \tag{9}$$

We use C_{t+1} in Equation (5) to compute Δ_t, then update λ_{t+1} and the sampling probabilities for the next batch. The denominator $\max(C_{t+1}, 0) + \varepsilon$ remains positive and avoids division by zero.

Table 1: Main results on V*, HR-Bench 4K/8K and VisualProbe. Acc. denotes answer accuracy; $\mathrm{Acc}_{\mathrm{loc}}$ and $\mathrm{Acc}_{\mathrm{joint}}$ denote crop localization and joint answer-and-localization accuracy (refer to Section 3.1). Bold and underline mark the best and second-best non-proprietary results.

Model	V*	HR-4K	HR-8K	VisualProbe Easy			VisualProbe Medium			VisualProbe Hard		
	Acc.	Acc.	Acc.	Acc.	$\mathrm{Acc}_{\mathrm{loc}}$	$\mathrm{Acc}_{\mathrm{joint}}$	Acc.	$\mathrm{Acc}_{\mathrm{loc}}$	$\mathrm{Acc}_{\mathrm{joint}}$	Acc.	$\mathrm{Acc}_{\mathrm{loc}}$	$\mathrm{Acc}_{\mathrm{joint}}$
Proprietary Models												
GPT-5.6 Sol	90.1	93.4	91.0	73.0	86.3	65.6	63.8	90.8	62.9	43.4	67.4	35.9
Gemini 3.8 Flash	97.4	96.6	97.6	84.4	94.7	84.7	79.9	97.2	82.9	69.8	80.4	69.6
Open-source Base Models												
Qwen2.5-VL-7B	78.0	69.3	64.9	39.0	–	–	16.4	–	–	22.6	–	–
Qwen3-VL-8B	82.7	74.4	71.0	41.8	–	–	23.5	–	–	24.5	–	–
Visual Tool Use Models												
Pixel-Reasoner	86.3	74.0	66.9	67.4	13.7	12.2	29.5	13.6	6.4	29.3	6.5	2.2
Thyme	82.2	77.0	72.0	63.8	–	–	37.7	–	–	40.6	–	–
DeepEyes	83.3	73.2	69.5	66.7	38.2	32.1	32.5	38.7	15.5	35.9	27.2	12.0
DeepEyesV2	81.8	77.9	73.8	58.9	11.5	10.7	35.8	17.5	8.0	34.9	14.1	4.4
CodeV	84.3	75.6	67.5	67.4	33.6	24.4	36.2	35.9	14.3	45.3	25.0	13.0
Mini-o3	85.3	71.8	67.5	63.1	<u>90.8</u>	61.1	49.0	<u>91.2</u>	48.5	42.5	<u>73.9</u>	39.1
SenseNova-MARS	<u>92.2</u>	**83.1**	<u>78.4</u>	<u>69.5</u>	86.3	<u>64.1</u>	<u>56.7</u>	86.1	<u>56.0</u>	**53.8**	66.3	<u>42.4</u>
CalDS (Ours)	**93.7**	<u>82.0</u>	**79.8**	**71.8**	**91.1**	**67.2**	**59.7**	**91.5**	**59.2**	<u>53.5</u>	**79.3**	**46.7**

3 Experiment

3.1 Implementation Details

Training. We train Qwen3-VL-8B-Instruct (Bai et al., 2025a) with GRPO for 200 iterations, using a binary LLM judge answer correctness reward. We cap the global image at 1.5M pixels during training, while the crop tool accesses the original image. For CalDS, we use $K = 10$ difficulty buckets and a history window of $X = 20$. Additional training details are provided in Appendix A.1.

Data. We collect and filter training data from Mini-o3 (Lai et al., 2026) and Thyme (Zhang et al., 2025) to focus on challenging high-VTN visual questions. Details are in Appendix A.2. The images in resulting dataset has an average 16.45M pixels, and Qwen3-VL achieves only 5.53% accuracy.

Evaluation. We evaluate Avg@3 performance on V*Bench (Wu & Xie, 2024), HR-Bench (Wang et al., 2025b), and VisualProbe (Lai et al., 2026) and compare with proprietary, open-source base, and visual tool-use models (details in Appendix A.4). To assess whether the model locates relevant visual evidence, we manually annotate answer regions in VisualProbe. These annotations support evaluation of crop localization and joint accuracy; annotation and metric details are in Appendix A.3.

3.2 Main Results

Table 1 summarizes the results on V*Bench, HR-Bench, and VisualProbe. Since Qwen3-VL already achieves high without-tool accuracy on V*Bench and HR-Bench, our following analysis focuses on the more challenging VisualProbe. We reproduce all comparison methods on VisualProbe and evaluate crop localization and joint answer-and-loc accuracy. Reproduction details are in Appendix A.4.

CalDS achieves state-of-the-art answer and joint accuracy among the compared models on the most challenging VisualProbe benchmark. CalDS uses less training compute and data than strong competitors Mini-o3 and SenseNova-MARS by directly applying RL to the base model on filtered challenging examples, whereas both competitors use multi-node SFT followed by RL on larger datasets. Moreover, CalDS achieves the highest crop localization and joint answer-and-localization accuracy among the evaluated non-proprietary models, reaching 79.3% and 46.7%, respectively, on the Hard subset, indicating its ability to correctly locate relevant visual evidence. This suggests that CalDS encourages visual evidence acquisition beyond overfitting tool-use patterns.

3.3 Ablation Studies and Analysis

Ablation studies. Table 2 reports answer and joint answer-and-localization accuracy for CalDS ablations on VisualProbe. Training on hard examples alone (raw 2) yields only modest gains in answer accuracy. As discussed in Section 2.2, many hard/easy prompts yield all-zero/all-one reward

Table 2: Ablation studies on VisualProbe (%). Hard-only: training on hard examples only; Dyn. Buck.: dynamic buckets; Reward Samp.: reward-guided sampling; Diff. Calib.: online difficulty calibration; CP Shape.: center-preserving difficulty shaping. Analysis is in Section 3.3.

Components					Easy		Medium		Hard	
Hard-only	Dyn. Buck.	Reward Samp.	Diff. Calib.	CP Shape.	Acc.	$\mathrm{Acc}_{\mathrm{joint}}$	Acc.	$\mathrm{Acc}_{\mathrm{joint}}$	Acc.	$\mathrm{Acc}_{\mathrm{joint}}$
					62.4	54.7	48.5	47.1	40.5	29.2
✓					65.1	58.8	50.0	47.8	43.1	36.0
✓	✓				65.4	58.4	54.1	51.4	45.2	38.8
✓	✓	✓			68.0	63.2	56.1	55.4	49.1	44.7
✓	✓	✓	✓		70.7	66.2	57.6	56.8	49.3	45.1
✓	✓	✓		✓	68.6	63.1	56.8	56.1	53.4	46.9
✓	✓	✓	✓	✓	71.8	67.2	59.7	59.2	53.5	46.7

Table 3: Comparison with dynamic sampling on VisualProbe. Rollout acceptance rate denotes the percentage of generated examples retained after rollout filtering of dynamic sampling. CalDS selects examples before rollout, accelerating the training process while yielding comparable performance.

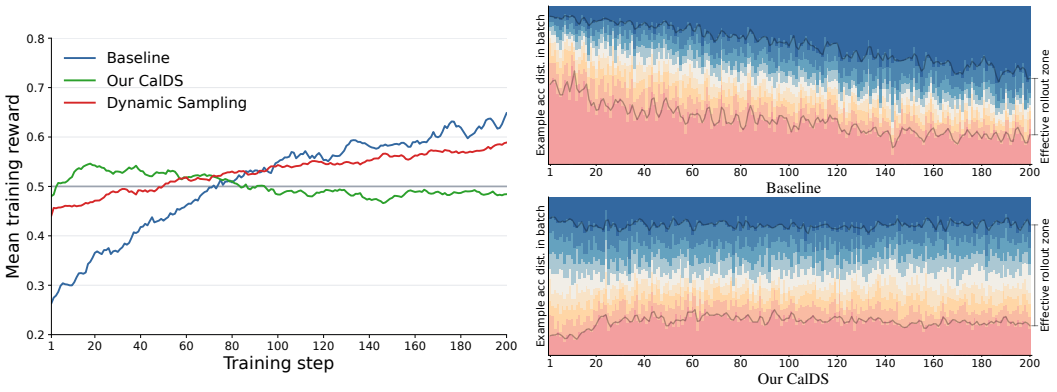
Method	Easy		Medium		Hard		Training Efficiency	
	Acc.	$\mathrm{Acc}_{\mathrm{joint}}$	Acc.	$\mathrm{Acc}_{\mathrm{joint}}$	Acc.	$\mathrm{Acc}_{\mathrm{joint}}$	Rollout Acceptance Rate	Batch Rollout Time
Dynamic sampling	71.6	64.4	58.3	57.9	46.8	46.2	32.8%	35.7 min
CalDS (Ours)	**71.8**	**67.2**	**59.7**	**59.2**	**53.5**	**46.7**	–	**9.1** min

groups, leaving GRPO with no learning signal. Dynamic buckets and reward-guided sampling provide larger gains by making the difficulty composition of training batches controllable. This reduces the risk of batches being dominated by unsuccessful rollout groups. Online difficulty calibration and center-preserving difficulty shaping further improve performance: calibration tracks changes in policy success rates, while shaping keeps sampling probabilities comparable across intermediate buckets as examples migrate toward easier ones. Besides, our baseline is already competitive with most comparison methods other than Mini-o3 and SenseNova-MARS, which we attribute in part to differences in training data. Even without hard-example filtering, the data contain challenging questions that can encourage visual evidence acquisition through tools. The ablations therefore measure the additional benefits of adaptive difficulty control on top of this baseline.

Comparison with dynamic sampling. As discussed in Section 1, dynamic sampling filters out examples after rollout to retain informative training groups. We compare CalDS with a dynamic sampling baseline that retains only examples with intermediate accuracy. Table 3 reports VisualProbe performance and training efficiency. On our hard training dataset, dynamic sampling retains only 32.8% of generated examples, increasing rollout time. Without our difficulty-aware selection before rollout, sampling can draw many examples that are too difficult for the current policy. Such examples can require more tool-use turns before their rollouts terminate, only to be discarded by the subsequent filter. Moreover, these rollouts must still be evaluated by the LLM judge, adding evaluation cost. CalDS adjusts the training distribution to the current policy's capabilities using difficulty estimates before rollout, retaining all generated rollouts. It achieves approximately a $3.9\times$ rollout speedup with comparable performance.

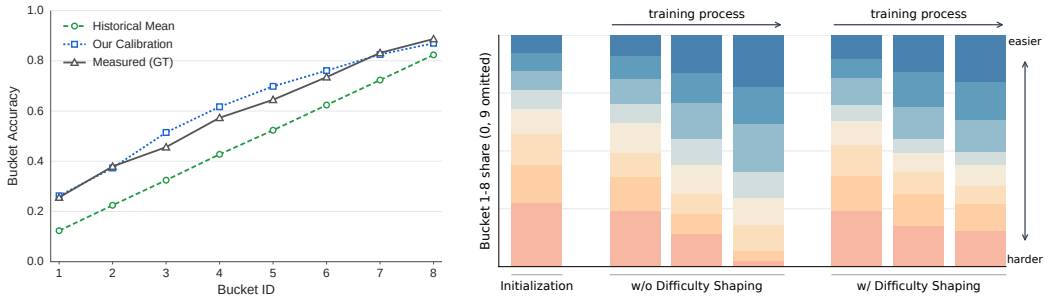
Effect of reward-guided sampling. CalDS uses reward feedback from the preceding batch to adjust sampling probabilities and control the next batch's difficulty (Section 2.2). Figure 3(a) plots the smoothed batch-average training reward. CalDS keeps the mean reward close to the target of 0.5 throughout training. In contrast, the baseline trained on our hard dataset starts with low rewards that later become too high. Costly dynamic sampling keeps rewards in an intermediate range, but its reward still drifts upward. These trajectories support the effectiveness of CalDS in adapting batch difficulty to the evolving policy without repeated rollout generation.

Controlling the mean alone, however, does not ensure informative GRPO groups: as discussed in Section 2.2, a batch reward of 0.5 can also arise from a mixture of all-zero and all-one groups. Figure 3(b) therefore shows the within-batch distribution of per-example accuracy at each step. With eight rollouts per example, accuracy takes one of nine values. Only examples with intermediate accuracy (between the smoothed gray lines) provide learning signals. The baseline contains too many hard examples early in training and too many easy examples later, reducing training efficiency. CalDS retains more informative groups with intermediate accuracy. This is consistent with more



(a) Training reward curve (b) Within-batch accuracy distribution

Figure 3: (a) Batch-average training reward for baseline, CalDS, and dynamic sampling. Our CalDS stabilizes the reward close to 0.5. (b) Per-step example accuracy distribution within batches for the baseline (top) and CalDS (bottom). Only examples between the gray lines provide GRPO learning signals. CalDS has more balanced mixture of accuracy levels, providing better training efficiency.



(a) Online difficulty calibration (b) Center-preserving difficulty shaping

Figure 4: (a) Estimated and measured accuracy of buckets. Our online calibration more closely matches the measured ground truth accuracies, while historical means consistently underestimate them. (b) Relative sizes of intermediate buckets during training. Center-preserving difficulty shaping mitigates the bottleneck problem and maintains a more balanced size distribution.

effective use of the rollout budget for GRPO updates. We also observe that a sizable fraction of all-zero groups remains. Sampling difficult examples is necessary to refresh their difficulty estimates and detect when they become learnable; excluding them would leave their estimates stale.

Effect of online difficulty calibration. Historical success rates can lag behind the evolving policy, making buckets appear harder than they really are (Section 2.3). To assess whether calibration reduces this mismatch, we evaluate all examples at training step 100 using the corresponding model checkpoint. The measured per-bucket accuracies provide ground truth for comparing historical mean success rates with calibrated accuracy forecasts. Figure 4(a) shows that historical means consistently underestimate the measured accuracies, whereas calibrated forecasts track them much more closely. Here, we show only the informative intermediate buckets. By reusing recent rollouts, online calibration estimates current bucket difficulty more accurately without additional cost.

Effect of center-preserving difficulty shaping. As examples migrate toward easier buckets, the bucket sampling distribution derived via Lagrange multipliers can create bottlenecks (Section 2.3): large probability differences deplete frequently sampled buckets, making examples accumulated in less frequent ones. Center-preserving difficulty shaping compresses score differences among intermediate buckets. To assess its effect, we track bucket sizes during the training. For clarity, Figure 4(b) shows the relative sizes of informative intermediate buckets; red denotes harder buckets and blue denotes easier ones. Without shaping, the imbalanced sampling distribution makes shares of some hard buckets shrink sharply as training progresses. This uneven evolution is consistent with the migration bottlenecks identified above. With shaping, bucket shares remain more balanced throughout the training, avoiding severe depletion of individual intermediate buckets. These dynamics support smoother migration across difficulty levels and more balanced learning.

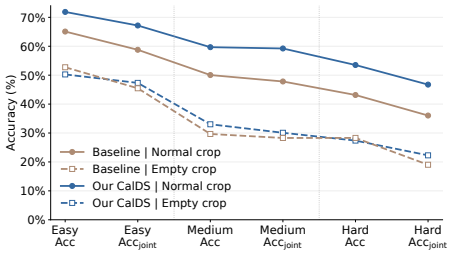


Figure 5: Experiment on masking tool returned crop image. CalDS's advantage comes from better reasoning with tools.

Table 4: Hyperparameter sensitivity on VisualProbe. Each variant changes one parameter from the defaults configuration: $\tau = 0.5$, $\alpha = 0.05$, $X = 20$, and $\gamma = 5$.

Setting		Easy		Medium		Hard	
		Acc.	Acc$_{joint}$	Acc.	Acc$_{joint}$	Acc.	Acc$_{joint}$
CalDS (default)		71.8	67.2	**59.7**	**59.2**	**53.5**	**46.7**
Reward target τ	0.4	70.0	62.8	57.1	55.5	45.9	35.9
	0.6	**76.4**	**68.4**	56.6	55.9	49.7	42.0
Calibration α	0.02	70.7	67.9	59.2	58.6	52.2	**46.7**
	0.1	69.0	63.1	56.1	54.7	41.2	37.7
History window X	8	69.3	61.8	58.2	57.6	50.3	42.8
	40	70.7	59.8	56.2	53.3	45.9	41.7
Shaping exponent γ	3	72.6	66.7	56.2	56.4	41.5	38.4
	7	69.2	65.8	58.9	57.2	49.9	45.5

Utilization of visual feedback. To examine whether CalDS benefits from crop images, we conduct an inference-time intervention inspired by TextCall (Shao et al., 2026): each returned crop image is replaced with `[Image output skipped]`, while the original image and tool-call text remain available. Figure 5 compares the baseline and CalDS on VisualProbe. CalDS substantially outperforms the baseline when crop images are available, but this advantage disappears when they are removed, yielding similar no-crop accuracies. Together with the improved crop localization in Table 1, this finding suggests that the gains of CalDS stem from more accurate cropping that supplies relevant visual evidence, rather than overfitting to the tool-call schema.

Hyperparameter sensitivity. Table 4 reports the effects of varying four hyperparameters individually. A higher reward target improves Easy performance at the expense of Medium and Hard, while a lower target underperforms the default because of slower learning. Slower calibration updates retain comparable performance, whereas aggressive updates harm overall performance, suggesting a benefit from gradual forecast adaptation. Both shorter and longer history windows underperform the default, consistent with a tradeoff between responsiveness and reliable difficulty estimates. Weaker shaping reduces Hard subset accuracy due to imbalanced training; stronger shaping offers comparable results. These observations favor the default settings for balanced performance.

4 Related Work

Large vision-language reasoning models. OpenAI o3 (OpenAI, 2025) demonstrates strong visual reasoning capabilities, while the open-source Qwen series (Bai et al., 2025b;a; Team, 2026) provides a foundation for multimodal reasoning. V* (Wu & Xie, 2024) first uses guided visual search to locate fine-grained evidence. Visual tool-use methods interleave reasoning with image operations (Su et al., 2026; Zheng et al., 2026), providing the crop-and-zoom setting in which we study efficient training.

Visual tool use via reinforcement learning. Pioneer Pixel Reasoner (Su et al., 2026) combine cold-start SFT with RL. DeepEyes (Zheng et al., 2026) instead trains directly with RL with tool-use reward. Training on questions difficult without tools encourages visual inspection, but all-0 groups provide no GRPO learning signal. DAPO (Yu et al., 2026) filters and resamples such groups, incurring additional rollout costs. CalDS adapts batch difficulty before generation to learn efficiently.

Tool utility and faithful visual tool use. MED (Ma et al., 2026b) attributes visual tool-use training gains mainly to intrinsic learning and reduced tool-induced harm, while TextCall (Shao et al., 2026) shows that gains can persist without returned images. Prior work uses explicit rewards or supervision for intermediate visual behavior to encourage task-relevant evidence acquisition (Liu et al., 2025; Hou et al., 2026; Wang et al., 2025a). CalDS complements these methods by improving training efficiency on questions difficult without tools. **Detailed survey is provided in Appendix C.**

5 Conclusion

We presented CalDS for efficient visual tool-use training on high-VTN datasets. By integrating dynamic difficulty tracking and reward-guided sampling with calibrations, CalDS matches batch difficulty to the evolving policy within a fixed rollout budget. On VisualProbe, it improves answer accuracy and crop localization, while substantially reducing rollout time compared with dynamic sampling. Crop-image ablations show that its gains depend on returned visual evidence. These results support adaptive difficulty control for efficient and effective visual tool use training.

AI USE STATEMENT

In this work, we used generative AI tools to polish the manuscript for clarity and readability and to assist in writing LaTeX code for figures and tables. We also used AI deep research functionality to support a broad search for relevant literature for the related work section. We reviewed all AI-assisted work, including the edited text and generated LaTeX code. For the literature search, we manually verified the papers identified through AI-generated lists before citing them in the manuscript. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We document the method and experimental procedures in the main paper and appendices to support reproduction of our results. Section 2 specifies the crop-and-zoom interaction, GRPO objective, dynamic bucket updates, reward-guided sampling, online calibration, and difficulty shaping. Algorithm 1 in Appendix F provides the training procedure, including initialization and the order of policy, bucket, forecast, and controller updates. Appendices D and E provide the assumptions and derivations underlying the sampling distribution and calibrated controller sensitivity. Section 3.1 identifies the base model, training data sources, image-resolution and without-tool accuracy filters, default CalDS hyperparameters, evaluation benchmarks, and repeated-evaluation protocol. Appendix A.1 further specifies the rollout engine, optimization and generation settings, checkpoint and validation schedule, answer-verification procedure, image and coordinate conventions, and interaction and token limits. Appendix H provides the LLM judge prompt used to assess answer correctness. For evaluation of visual evidence acquisition, Appendix A.3 describes the AI-assisted pre-annotation and manual review workflow, exclusion criteria and resulting evaluation subsets, crop eligibility and overlap criteria, and the definitions and aggregation of localization and joint answer-and-localization accuracy. Appendix A.4 distinguishes reproduced baseline results from those taken from prior reports and describes the adaptations used to align crop coordinates and evaluation protocols across methods. Section 3.3 documents the component ablations and dynamic-sampling comparison, with the tested hyperparameter values reported in Table 4. Training-efficiency results are reported in Table 3, and Appendix B.1 separately accounts for the initialization cost and describes the reusable rollout outcomes and bucket assignments. We commit to openly releasing all code, data, and models associated with this work, including training and evaluation code, experiment configurations, data-processing and annotation tools, processed datasets and region annotations, bucket-initialization results, and trained model checkpoints.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Yong Xien Chng, Tao Hu, Wenwen Tong, Xueheng Li, Jiandong Chen, Haojia Yu, Jiefan Lu, Hewei Guo, Hanming Deng, Chengjun Xie, et al. Sensenova-mars: Empowering multimodal agentic reasoning and search via reinforcement learning. *arXiv preprint arXiv:2512.24330*, 2025.

Hao Ding, Zhichuan Yang, Weijie Ge, Ziqin Gao, Chaoyi Lu, and Lei Zhao. Tikart: Stabilizing aperture-guided fine-grained visual reasoning with reinforcement learning. *arXiv preprint arXiv:2602.14482*, 2026.

Qihua Dong, Gozde Sahin, Pei Wang, Zhaowei Cai, Robik Shrestha, Hao Yang, and Davide Modolo. Visual reasoning through tool-supervised reinforcement learning. *arXiv preprint arXiv:2604.19945*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Dengxian Gong, Yuanzheng Wu, Haobo Yuan, Zhengdong Hu, Tao Zhang, Yikang Zhou, Shihao Chen, Quanzhu Niu, Kai Wang, Jason Li, et al. Pixeleyes: Decoupling perception and reasoning for pinpoint visual evidence seeking. *arXiv preprint arXiv:2607.00115*, 2026.

Google DeepMind. Gemini 3.8 Flash: Model card. `https://deepmind.google/models/model-cards/gemini-3-8-flash/`, September 2026.

Haokun Gui, Senqiao Yang, Mingkang Zhu, Meng Chu, Sitong Wu, Changsheng Lu, Zihao Wang, Zhuotao Tian, and Jiaya Jia. Visionleaf: Entropy-guided leaf-first reasoning for efficient and accurate think-with-image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5187–5198, 2026.

Garvin Guo, Donglei Yu, Yu Chen, Xiang Wang, Shuai Li, Xinpei Zhao, Huaxing Liu, Qinghao Wang, and Minpeng Liao. Do multimodal agents really benefit from tool use? a systematic study of capability gains. *arXiv preprint arXiv:2606.02357*, 2026.

Falko Helm and Iryna Gurevych. Learning to zoom efficiently with a contrastive curriculum. *arXiv preprint arXiv:2609.03206*, 2026.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. Deepeyesv2: Toward agentic multimodal model. *arXiv preprint arXiv:2511.05271*, 2025.

Xinhai Hou, Shaoyuan Xu, Manan Biyani, Moyan Li, Jia Liu, Todd C Hollon, and Bryan Wang. Codev: Code with images for faithful visual reasoning via tool-aware policy optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21500–21510, 2026.

Ruina Hu, Chen Wang, Lai Wei, Jionghao Bai, Bin Yu, Weiran Huang, Kai Wang, and Yue Wang. Attend to evidence: Evidence-anchored spatial attention supervision for multimodal rlvr. *arXiv preprint arXiv:2605.30912*, 2026.

Wenxuan Huang, Bohan Jia, Shaosheng Cao, Zheyu Ye, Zhe Xu, Yao Hu, Shaohui Lin, et al. Vision-r1: Incentivizing reasoning capability in multimodal large language models. In *International Conference on Learning Representations*, volume 2026, pp. 63794–63812, 2026.

Xinyu Huang, Yuhao Dong, Weiwei Tian, Bo Li, Rui Feng, and Ziwei Liu. High-resolution visual reasoning via multi-turn grounding-based reinforcement learning. *arXiv preprint arXiv:2507.05920*, 2025.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R Fung. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026.

Dong-Hee Kim, Reuben Tan, and Donghyun Kim. Diversity over frequency: Rethinking tool use in visual chain-of-thought agents. *arXiv preprint arXiv:2606.00096*, 2026.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. In *International Conference on Learning Representations*, volume 2026, pp. 76722–76746, 2026.

Xuchen Li, Xuzhao Li, Renjie Pi, Shiyu Hu, Jian Zhao, and Jiahui Gao. Beyond accuracy: Evaluating grounded visual evidence in thinking with images. *arXiv preprint arXiv:2601.11633*, 2026.

Anjie Liu, Yan Song, Zhixun Chen, Ziqin Gong, Zhongwei Yu, and Jun Wang. Toolgate: Token-efficient pre-call control for tool-augmented vision-language agents. *arXiv preprint arXiv:2606.03054*, 2026.

Zujing Liu, Junwen Pan, Qi She, Yuan Gao, and Guisong Xia. On the faithfulness of visual thinking: Measurement and enhancement. *arXiv preprint arXiv:2510.23482*, 2025.

Qinghe Ma, Zhen Zhao, Yiming Wu, Jian Zhang, Lei Bai, and Yinghuan Shi. Are tools always beneficial? learning to invoke tools adaptively for dual-mode multimodal llm reasoning. *arXiv preprint arXiv:2605.19852*, 2026a.

Yan Ma, Weiyu Zhang, Tianle Li, Linge Du, Xuyang Shen, and Pengfei Liu. What does vision tool-use reinforcement learning really learn? disentangling tool-induced and intrinsic effects for crop-and-zoom. *arXiv preprint arXiv:2602.01334*, 2026b.

Kartik Narayan, Yang Xu, Tian Cao, Kavya Nerella, Vishal M Patel, Navid Shiee, Peter Grasch, Chao Jia, Yinfei Yang, and Zhe Gan. Deepmmsearch-r1: Empowering multimodal llms in multimodal web search. *arXiv preprint arXiv:2510.12801*, 2025.

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

OpenAI. GPT-5.6 system card. `https://deploymentsafety.openai.com/gpt-5-6`, July 2026.

Jiahao Shao, Yuanbo Yang, Yiyi Liao, Yujun Shen, Ceyuan Yang, and Yinghao Xu. Thinking with tools, not with pixels: Tool calls as text scaffolds for visual reasoning. *arXiv preprint arXiv:2608.09682*, 2026.

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

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. *Advances in Neural Information Processing Systems*, 38:8222–8251, 2026.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, et al. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers. *arXiv preprint arXiv:2506.23918*, 2025.

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

Changpeng Wang, Haozhe Wang, Xi Chen, Junhan Liu, Taofeng Xue, Chong Peng, Donglian Qi, Fangzhen Lin, and Yunfeng Yan. From illusion to intention: Visual rationale learning for vision-language reasoning. *arXiv preprint arXiv:2511.23031*, 2025a.

Qixun Wang, Yang Shi, Letian Cheng, Zhuoran Zhang, Yan He, Yuqi Tang, Qi Zhang, Xinlei Yu, Ruizhe Chen, Tianrun Xu, et al. Beacon: Knowing when and how to perform agentic visual reasoning. *arXiv preprint arXiv:2607.28595*, 2026a.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7907–7915, 2025b.

Zhiheng Wang, Bo Peng, Lai Wei, and Chaochao Lu. The illusion of visual tool-use: A causal audit of thinking with images. *arXiv preprint arXiv:2608.06270*, 2026b.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13084–13094. IEEE, 2024.

Yizheng Wu, Jiashen Hua, Bing Deng, and Jieping Ye. Vc-tooler: Learning compositional and adaptive visual tool use. *arXiv preprint arXiv:2608.02217*, 2026.

Changhao Xiang, Shilin Zhang, Zheng Ma, Kanzhi Cheng, Ruize Ma, Yi Feng, Jianbing Zhang, Zhi Wang, Zhen Wu, Xinyu Dai, et al. Openvistool: An open recipe for synthesizing instructive visual tool-use trajectories. *arXiv preprint arXiv:2608.08557*, 2026.

Shilin Yan, Jintao Tong, Hongwei Xue, Xiaojun Tang, Yangyang Wang, Kunyu Shi, Guannan Zhang, Ruixuan Li, and Yixiong Zou. Act wisely: Cultivating meta-cognitive tool use in agentic multimodal models. *arXiv preprint arXiv:2604.08545*, 2026.

Wenhao Yang, Yu Xia, Jinlong Huang, Shiyin Lu, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, Yuchen Zhou, Xiaobo Xia, et al. Walk the talk: Bridging the reasoning-action gap for thinking with images via multimodal agentic policy optimization. *arXiv preprint arXiv:2604.06777*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Ruiyang Zhang, Qianguo Sun, Chao Song, Yiyan Qi, and Zhedong Zheng. Vsearcher: Long-horizon multimodal search agent via reinforcement learning. *arXiv preprint arXiv:2603.02795*, 2026.

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

Shitian Zhao, Shaoheng Lin, Ming Li, Haoquan Zhang, Wenshuo Peng, Kaipeng Zhang, and Chen Wei. Pyvision-rl: Forging open agentic vision models via rl. *arXiv preprint arXiv:2602.20739*, 2026a.

Xuanpu Zhao, Zhentao Tan, Dianmo Sheng, Tianxiang Chen, Yao Liu, Yue Wu, Tao Gong, Qi Chu, and Nenghai Yu. Learning to focus and precise cropping: A reinforcement learning framework with information gaps and grounding loss for mllms. *arXiv preprint arXiv:2603.27494*, 2026b.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, and Chao Shen. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 126775–126798, 2026.

A Experimental Details

A.1 Additional Training Details

We train Qwen3-VL-8B-Instruct (Bai et al., 2025a) with GRPO for 200 iterations on 8 GPUs, keeping the vision encoder frozen. We use asynchronous SGLang for rollout generation. Each iteration samples $B = 128$ distinct prompts and generates $G = 8$ trajectories per prompt at temperature 1.0, giving a fixed budget of $BG = 1024$ trajectories. We use one optimization epoch per rollout batch, a constant learning rate of 2×10^{-6}, and clipping widths $\epsilon_{\text{low}} = 0.20$ and $\epsilon_{\text{high}} = 0.28$, without KL or entropy regularization. We save checkpoints and run validation every 20 iterations. For CalDS, sampling starts from $\lambda_0 = 0$, with a uniform reference and target success rate $\tau = 0.5$. For CalDS, the reward-feedback controller uses $\eta_\lambda = 0.05$, $\varepsilon = 10^{-3}$, $\Delta_{\max} = 0.2$, and $\lambda_{\max} = 10$. We set the calibration coefficient to $\alpha = 0.05$ and the difficulty-shaping exponent to $\gamma = 5$.

We use only a binary reward for final-answer correctness. We first check the generated answer against the reference answer using a rule-based verifier. If the verifier accepts the answer, we assign a reward of 1. Otherwise, we use Qwen3.5-Flash as an LLM judge to assess its correctness against the reference answer, assigning a reward of 1 if the judge accepts it and 0 otherwise. No auxiliary format or tool-use rewards are applied.

To encourage inspection of local details, we cap the global image at $1{,}572{,}864$ pixels (approximately 1.5M) during training, while the crop tool accesses the original image. Crop coordinates are normalized to $[0, 1000]$. Each trajectory allows at most five assistant turns and five user/tool-observation turns, with prompt and response token limits of $16{,}000$ and $8{,}192$, respectively.

A.2 Additional Data Details

Starting from the training collection used by MED (Ma et al., 2026b), we retain the data from Mini-o3 (Lai et al., 2026) and Thyme (Zhang et al., 2025) to focus on challenging high-VTN visual questions. Mini-o3 targets small objects in cluttered images, while Thyme's high-resolution questions undergo cross-checking and expert verification, emphasizing fine-grained perception (Lai et al., 2026; Zhang et al., 2025). We first remove examples with images below one million pixels. We then filter by the base model's ability to answer without tools: Qwen3-VL generates five answers per example, and we retain examples with at most two correct answers. Together, these filters select high-resolution examples that the base model cannot consistently solve without tools, focusing training on questions that may benefit from inspecting local visual evidence. Since these questions mainly concern visual details rather than complex abstract reasoning, filtering for low without-tool accuracy provides a reasonable proxy for selecting high-VTN examples.

The resulting dataset has an average image size of 16.45M pixels, and Qwen3-VL achieves only 5.53% accuracy on these training data. Compared with the 1.5M-pixel cap on the global image during training, these images provide substantially more detail for the crop tool to access at the original resolution.

A.3 VisualProbe Annotation

We construct region annotations for VisualProbe through AI-assisted pre-annotation followed by manual verification and annotation. The annotations identify the visual evidence needed to answer each question and are used exclusively for evaluation, providing no training supervision.

AI-assisted pre-annotation. We use Gemini-3-Flash-Preview with code execution to inspect and zoom into each image, answer the question, and return a bounding box for the key evidence region. The prompt allows a crop that contains sufficient evidence without requiring a tight object boundary. We first verify the generated answer against the reference answer using a rule-based check, with Qwen3.5-Flash as a fallback judge for semantic equivalence. For answers judged correct, Gemini-3-Flash-Preview then checks whether the proposed region supports the answer, given the question, generated answer, original image with the box overlaid, and cropped image. Only boxes passing both checks are loaded as pre-annotations in the manual interface. This stage supplies candidate boxes for 408 of the 515 VisualProbe examples.

Manual verification and annotation. We develop a web interface that displays the original image alongside the question, reference answer, and a visual summary of the AI annotation. The interface overlays available pre-annotations and supports image zooming and drawing replacement boxes. Two PhD students participate in the manual annotation process. The first completes the initial annotations by inspecting each example, retaining or revising the proposed box, and drawing an evidence region when no accepted pre-annotation is available. The second then reviews the annotations to verify that the selected regions contain the visual evidence needed to answer the questions. Each saved annotation contains one bounding box with coordinates normalized to $[0, 1000]$ in $[\text{top}, \text{left}, \text{bottom}, \text{right}]$ order. During annotation, we identify 41 examples containing errors or ambiguities and exclude them from the region evaluation set, leaving 474 examples: 131 Easy, 251 Medium, and 92 Hard. Across these 474 retained examples, the annotated bounding boxes have a mean box-to-image area ratio of 0.1760%, with mean ratios of 0.2382%, 0.1776%, and 0.0833% for the Easy, Medium, and Hard subsets, respectively. Standard answer accuracy is computed on the full VisualProbe benchmark of 515 examples for fair comparison, while metrics involving region annotations, including crop localization and joint answer-and-localization accuracy, use only the 474 retained examples.

Crop localization. For example i, let A_i denote the annotated region and $\mathcal{P}_i$ the set of distinct predicted crop boxes extracted from its trajectory. We convert annotation and prediction coordinates to a common convention before comparison. To prevent overly broad crops from receiving credit, a predicted crop is eligible only if its area is at most 25% of the original image area and its width and height are each at most two thirds of the corresponding image dimension. Let $\mathcal{P}_i^{\text{elig}}$ denote the eligible crops. We define localization correctness as

$$\ell_i = \mathbf{1}\left[\exists P \in \mathcal{P}_i^{\text{elig}} : \frac{|P \cap A_i|}{\min(|P|, |A_i|)} \geq 0.5\right], \tag{A1}$$

where $|\cdot|$ denotes area. The overlap is normalized by the smaller box area, allowing the predicted crop and annotation to differ in extent. Any eligible crop in the trajectory can satisfy this criterion. Examples with no predicted crop, or only ineligible crops, receive $\ell_i = 0$.

Joint answer-and-localization accuracy. Let $a_i \in \{0, 1\}$ denote final-answer correctness under the standard answer evaluation protocol. On an annotated subset of N retained examples, we compute

$$\text{Acc}_{\text{loc}} = \frac{1}{N}\sum_{i=1}^{N} \ell_i, \qquad \text{Acc}_{\text{joint}} = \frac{1}{N}\sum_{i=1}^{N} a_i \ell_i. \tag{A2}$$

Joint accuracy therefore requires a correct answer and a qualifying crop in the same trajectory. Both metrics use the same retained examples, including those without a crop in their denominators. We compute the metrics for each evaluation repetition and report their arithmetic mean over three repetitions (Avg@3).

A.4 Comparison Works Reproduction Details

We compare against three groups of models: proprietary models GPT-5.6 (OpenAI, 2026) and Gemini 3.8 Flash (Google DeepMind, 2026); open-source base models Qwen2.5-VL-7B (Bai et al., 2025b) and Qwen3-VL-8B (Bai et al., 2025a); and visual tool-use models, including Pixel-Reasoner (Su et al., 2026), Thyme (Zhang et al., 2025), DeepEyes (Zheng et al., 2026), DeepEyesV2 (Hong et al., 2025), CodeV (Hou et al., 2026), Mini-o3 (Lai et al., 2026), and SenseNova-MARS (Chng et al., 2025). We report Avg@3 performance, and all other evaluation settings strictly follow MED (Ma et al., 2026b).

We reproduce the comparison methods only on VisualProbe to obtain answer accuracy together with crop localization and joint answer-and-localization accuracy. Baseline answer accuracies on V*Bench and HR-Bench are taken from the original papers or existing reports (Gong et al., 2026; Wang et al., 2026a). For each reproduced method, we strictly follow the settings and hyperparameters specified in its original paper, making only the adaptations needed to process VisualProbe data. To ensure consistent answer evaluation, we use the same LLM judge model as for our method.

The comparison methods use different coordinate conventions for their crop tools. We adapt each method's predicted boxes to a common coordinate order and normalize them to $[0, 1000]$ with respect to the original image dimensions before computing region metrics. All methods are then evaluated on the same annotated subset using the crop eligibility constraints, overlap criterion, and metric definitions in Appendix A.3.

In our VisualProbe reproduction, Thyme invokes the crop tool in only approximately 10% of cases. Given this limited crop usage, its region metrics would primarily reflect the absence of crop calls and offer limited insight into localization quality. We therefore mark its crop localization and joint answer-and-localization accuracies as "–" in Table 1. Besides, DeepEyesV2 (Hong et al., 2025) invokes tools through generated code and devotes part of its training to search capabilities. This broader training focus may help explain why it does not consistently outperform DeepEyes on challenging VisualProbe that requires fine-grained visual localization. Thus in our reproduction, DeepEyesV2 has lower answer accuracy on Easy and Hard and lower localization and joint accuracy on all three splits. We regard the difference in training focus as a possible explanation rather than an isolated causal finding.

B Additional Experimental Results and Analysis

B.1 Bucket Initialization Cost

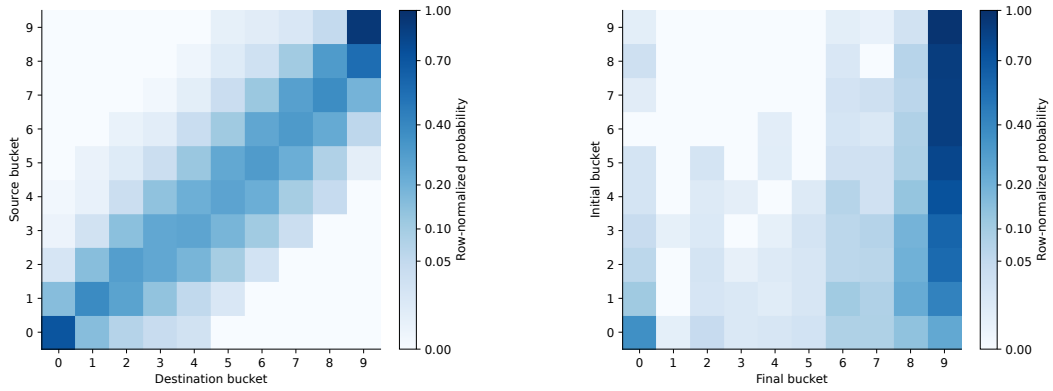
CalDS requires tool-enabled base-model rollouts before RL training to initialize the difficulty estimates and bucket assignments. This introduces an upfront computational cost, but the rollouts are independent and can be fully parallelized. In our experiments, all initialization rollouts were completed in 7 hours on two nodes, with a GPU-hour cost that is small relative to subsequent RL training. Even when the full initialization cost is included, CalDS uses only approximately $1/2.8$ of the total GPU-hours required by dynamic sampling. We emphasize that this initialization is performed only once: the resulting rollout outcomes and initial bucket assignments can be reused across all subsequent experiments. Its amortized cost is therefore negligible across the project. We will publicly release these initialization results so that follow-up work using the same setup can reuse them without repeating the initialization rollouts.

B.2 Bucket Migration Dynamics

To examine how dynamic buckets track changes in example difficulty during training, Figure A1 visualizes bucket transitions at two timescales. Buckets are ordered from hardest (0) to easiest (9). The vertical axis indicates the source bucket and the horizontal axis the destination bucket; each row is normalized to sum to one and thus represents a conditional transition distribution.

Local transitions after rollout. Figure A1(a) aggregates the bucket updates of sampled examples after each training step. For intermediate source buckets, transition mass is concentrated near the diagonal and typically shifted one or two buckets to the right. This pattern indicates that recent rollouts tend to increase estimated success rates, consistent with the policy becoming more capable during training. Bucket 0, however, has a substantial probability of remaining unchanged, showing that the hardest examples remain difficult even after being revisited. This persistent difficulty helps explain the inefficiency of uniform random sampling on our hard dataset: repeatedly sampling examples that still yield all-zero rollout groups consumes computation without providing a relative reward signal for GRPO.

Cumulative migration over training. Figure A1(b) compares initial and final bucket assignments. For most initial buckets, the largest share of examples ends in bucket 9, indicating that many initially challenging cases become reliably solvable according to their recent rollout histories. As these easy examples accumulate, uniform random sampling increasingly spends its budget on cases the policy already solves, which tend to yield all-one groups with little learning signal. CalDS continues to seek challenging examples through updated bucket assignments and adaptive sampling probabilities, maintaining attention on the remaining learning opportunities even late in training. Together, the two views motivate adapting the sampling distribution throughout training,



(a) Per-step bucket transitions

(b) Initial-to-final bucket transitions

Fig. A1: Bucket migration during CalDS training. (a) Transitions from pre-rollout assignments to updated assignments for sampled examples, aggregated across training steps. (b) Transitions from initialization to the end of training. Rows denote source buckets and columns denote destination buckets, ordered from hardest (0) to easiest (9); each row sums to one. Examples generally move toward slightly easier buckets. By the end of training, bucket 9 is the dominant destination for most initial buckets.

as both persistently hard examples and increasingly easy ones can limit the usefulness of uniformly sampled rollouts.

C Detailed Related Work

Large vision-language reasoning models. Recent VLMs combine increasingly capable visual representations with language-based reasoning. OpenAI o3 (OpenAI, 2025) combines multi-step reasoning across text, code, and images with tool-use capabilities, providing a general-purpose foundation for tool-augmented visual reasoning. For open source models, the Qwen series (Bai et al., 2025b;a; Team, 2026) support reasoning over multiple images, video, and interleaved multimodal contexts. Vision-R1 (Huang et al., 2026) develops multimodal reasoning through a combination of synthesized chain-of-thought supervision and RL. For tasks requiring fine visual details, reasoning also depends on acquiring evidence that the initial image representation may omit. V* (Wu & Xie, 2024) addresses this through guided visual search. More recent approaches interleave textual reasoning with visual operations and returned observations (Su et al., 2026; Zheng et al., 2026). Our work follows this interactive reasoning paradigm and studies how to train crop-and-zoom behavior efficiently on difficult visual questions.

Visual tool use via reinforcement learning. One approach to learning visual tool use combines supervised fine-tuning (SFT) on tool-augmented trajectories with subsequent RL. Pixel Reasoner (Su et al., 2026) uses synthesized traces to initialize visual operations and curiosity-driven rewards to encourage their exploration. Mini-o3 (Lai et al., 2026) collects diverse cold-start trajectories and introduces over-turn masking to support deeper visual search. These recipes require constructing and filtering high-quality intermediate reasoning trajectories, a costly process that is difficult to scale. DeepEyes (Zheng et al., 2026) shows that a model's existing grounding capabilities can instead support direct RL without cold-start SFT, using a conditional tool-use reward. After that, a series of works (Su et al., 2025; Chng et al., 2025; Huang et al., 2025; Dong et al., 2026; Zhao et al., 2026a; Wu et al., 2026; Helm & Gurevych, 2026; Ma et al., 2026a; Yan et al., 2026; Wang et al., 2026b; Liu et al., 2026; Jiang et al., 2026; Kim et al., 2026; Gui et al., 2026) have been introduced to further improve visual tool use for a broader range of applications. As what we have discussed, training on examples that are difficult to solve without tools is essential for incentivizing effective visual tool use. However, under common used GRPO (Shao et al., 2024), entirely unsuccessful rollout groups provide no relative reward signal. DAPO's dynamic sampling (Yu et al., 2026) addresses this through filtering and resampling, at the cost of numerous additional tool-augmented

rollouts under hard datasets. CalDS instead adapts batch difficulty before generation, enabling efficient learning from hard examples within a fixed online rollout budget.

Tool utility and faithful visual tool use. Higher answer accuracy alone does not establish that a model has learned to acquire and use better visual evidence. MED (Ma et al., 2026b) finds that gains in its studied settings mainly reflect intrinsic learning and reduced tool-induced harm. TextCall (Shao et al., 2026) retains tool-call text but omits returned images, showing that gains can persist without visual feedback. These findings motivate distinguishing successful answers, relevant crops, and dependence on returned observations. Several methods address this gap through explicit supervision of intermediate visual behavior. Liu et al. (2025) reward sufficient and minimal visual evidence; CodeV (Hou et al., 2026) provides process rewards based on tool outputs; and ViRL (Wang et al., 2025a) explicitly rewards visual rationales. LFPC (Zhao et al., 2026b) creates an information gap by reducing global-image resolution and subsequently introduces grounding supervision using annotated regions. These approaches modify the signals that guide visual actions. Related studies report similar issues and offer some insights about this problem (Li et al., 2026; Ding et al., 2026; Yang et al., 2026; Xiang et al., 2026; Hu et al., 2026). Our work complements these approaches by improving training efficiency on questions that are difficult to solve without tools. These two lines of work are thus complementary and can be combined to support efficient and faithful visual tool use.

D Derivation and Analysis of Reward-Guided Sampling

CalDS groups prompts by recent rollout success rates and controls batch difficulty through bucket sampling proportions. Reward feedback adjusts a scalar λ to favor harder or easier buckets. We derive the resulting distribution and its sensitivity to λ.

Let $\mathcal{A}$ denote the nonempty buckets, with uniform reference weights $u_b = 1/|\mathcal{A}|$. Each bucket has a fixed score $d_b \in [0, 1]$, increasing from hard to easy through a monotone transformation of the bucket's success-rate interval center (Equation (7)); this score need not equal its current success probability. Candidate proportions w_b lie in the simplex $\Delta(\mathcal{A}) = \{w : w_b \geq 0,\ \sum_{b\in\mathcal{A}} w_b = 1\}$, and $\pi(\lambda)$ denotes the selected mixture. We hold $\mathcal{A}$, u_b, and d_b fixed, omit the iteration index, and sum over active buckets.

D.1 Derivation via Lagrange Multipliers

We use the method of Lagrange multipliers to solve the constrained optimization in Equation (3), obtaining the sampling distribution in Equation (4). The objective combines a KL penalty for departure from the reference with a weighted mean score. For a finite $\lambda \in \mathbb{R}$, define

$$F_\lambda(w) = \sum_{b\in\mathcal{A}} w_b \log \frac{w_b}{u_b} + \lambda \sum_{b\in\mathcal{A}} w_b d_b, \qquad w \in \Delta(\mathcal{A}), \tag{A3}$$

with $0 \log(0/u_b) = 0$. The first term is $D_{\mathrm{KL}}(w\|u)$, while $\sum_b w_b d_b$ is smaller for harder mixtures. Thus, positive λ favors harder buckets, negative λ favors easier buckets, and $\lambda = 0$ recovers the uniform reference.

Introduce a Lagrange multiplier ν for the normalization constraint $\sum_b w_b = 1$:

$$\mathcal{L}(w, \nu) = F_\lambda(w) + \nu \left(\sum_b w_b - 1 \right). \tag{A4}$$

For an interior candidate, stationarity gives

$$\frac{\partial \mathcal{L}}{\partial w_b} = \log \frac{w_b}{u_b} + 1 + \lambda d_b + \nu = 0, \qquad w_b = u_b e^{-\lambda d_b} e^{-1-\nu}. \tag{A5}$$

Writing $Z(\lambda) = \sum_b u_b e^{-\lambda d_b} > 0$, normalization requires $e^{-1-\nu} = 1/Z(\lambda)$, yielding Equation (4). To establish global optimality, including against boundary distributions, substitute $\log \pi_b = \log u_b -$

$\lambda d_b - \log Z$ into the objective:

$$F_\lambda(w) = \sum_b w_b \log \frac{w_b}{\pi_b(\lambda)} - \log Z(\lambda)$$
$$= D_{\mathrm{KL}}(w \| \pi(\lambda)) - \log Z(\lambda) \geq -\log Z(\lambda) = F_\lambda(\pi(\lambda)). \tag{A6}$$

Equality holds only for $w = \pi(\lambda)$. This proves that the strictly positive distribution obtained from stationarity is the unique minimizer on the entire simplex.

D.2 Control Direction and Mixture Sensitivity

For any active pair b, c, the normalizer cancels to give

$$\log \frac{\pi_b(\lambda)}{\pi_c(\lambda)} = \log \frac{u_b}{u_c} - \lambda(d_b - d_c). \tag{A7}$$

With uniform reference weights, the first term vanishes. If $d_b < d_c$, increasing λ increases the relative mass of b over c. This is a relative comparison: the absolute probability of a bucket depends on its score relative to the current mixture mean.

To obtain that dependence, let $\psi(\lambda) = \log Z(\lambda)$ and $\bar{d}(\lambda) = \sum_b \pi_b(\lambda) d_b$. Direct differentiation gives

$$\psi'(\lambda) = \frac{-\sum_b u_b d_b e^{-\lambda d_b}}{Z(\lambda)} = -\bar{d}(\lambda), \qquad \psi''(\lambda) = \sum_b \pi_b d_b^2 - \bar{d}^{\,2} = \mathrm{Var}_\pi(d). \tag{A8}$$

Differentiating $\log \pi_b = \log u_b - \lambda d_b - \psi$ and then summing against d_b yields

$$\pi_b'(\lambda) = \pi_b(\lambda)(\bar{d} - d_b),$$
$$\bar{d}'(\lambda) = \sum_b d_b \pi_b(\bar{d} - d_b) = \bar{d}^{\,2} - \sum_b \pi_b d_b^2 = -\mathrm{Var}_\pi(d) \leq 0. \tag{A9}$$

Here, $\mathrm{Var}_\pi(d)$ is the variance of bucket scores under the sampling distribution. As λ increases, buckets below the mean score gain probability and the mean score decreases, shifting sampling toward harder examples. The decrease is strict whenever at least two active scores differ, since all active probabilities are positive at finite λ. If only one score is present, changing λ leaves the mixture unchanged. More generally, buckets with equal scores retain their reference probability ratio for every λ.

D.3 Sampling Implementation

Every nonempty bucket retains positive probability for finite λ_t, though it may not appear in a finite batch. In practice, we round these probabilities to per-bucket counts $a_{b,t}$ satisfying $\sum_b a_{b,t} = B$ and $0 \leq a_{b,t} \leq n_{b,t}$. We select a set $S_{b,t}$ of $a_{b,t}$ prompts from each bucket's shuffled rotating queue, yielding B distinct prompts whose source buckets are recorded before rollout.

E Derivation and Analysis of Calibrated Controller Sensitivity

E.1 Derivation of the Covariance Sensitivity

Fix an iteration and hold the nonempty active set $\mathcal{A}$, reference weights $u_b > 0$, sampling scores d_b, and calibrated forecasts q_b constant. We suppress the iteration index while differentiating with respect to λ. Define

$$Z(\lambda) = \sum_{b \in \mathcal{A}} u_b e^{-\lambda d_b}, \qquad \pi_b(\lambda) = \frac{u_b e^{-\lambda d_b}}{Z(\lambda)}, \qquad m(\lambda) = \sum_{b \in \mathcal{A}} \pi_b(\lambda) q_b. \tag{A10}$$

All sums below run over active buckets. Write $\bar{d} = \sum_b \pi_b d_b$ and $\bar{q} = \sum_b \pi_b q_b = m(\lambda)$. The forecast q_b estimates success probability, whereas d_b determines how sampling responds to λ; these quantities need not coincide.

Differentiating the normalizer gives

$$Z'(\lambda) = -\sum_b u_b d_b e^{-\lambda d_b} = -Z(\lambda)\bar{d}. \tag{A11}$$

The quotient rule then yields

$$\begin{aligned}\pi_b'(\lambda) &= \frac{-d_b u_b e^{-\lambda d_b} Z(\lambda) - u_b e^{-\lambda d_b} Z'(\lambda)}{Z(\lambda)^2} \\ &= \pi_b(\lambda)\left(-d_b - \frac{Z'(\lambda)}{Z(\lambda)}\right) = \pi_b(\lambda)(\bar{d} - d_b).\end{aligned} \tag{A12}$$

Thus, increasing λ raises the probability of buckets with scores below the current mean. Since the forecasts are held fixed, the predicted reward derivative is

$$\begin{aligned}m'(\lambda) &= \sum_b q_b \pi_b'(\lambda) = \sum_b \pi_b q_b (\bar{d} - d_b) \\ &= \bar{q}\bar{d} - \sum_b \pi_b q_b d_b = -\sum_b \pi_b (q_b - \bar{q})(d_b - \bar{d}) \\ &= -\operatorname{Cov}_{b\sim\pi(\lambda)}(q_b, d_b) \equiv -C(\lambda).\end{aligned} \tag{A13}$$

This proves Equation (8). Applying the same calculation to $\bar{d}$ gives $\bar{d}'(\lambda) = -\operatorname{Var}_\pi(d) \le 0$. The mean sampling score therefore never increases with λ, whereas predicted reward decreases locally only when $C(\lambda) > 0$. If $q_b = d_b$, the covariance reduces to $\operatorname{Var}_\pi(d)$. With center-preserving shaping, the sampling coordinates and calibrated forecasts differ, so the reward derivative requires their covariance.

E.2 Sensitivity Sign and the Controller Update

The sign of the covariance can be understood by expanding pairwise differences:

$$\begin{aligned}\frac{1}{2}\sum_{b,c} \pi_b \pi_c (q_b - q_c)(d_b - d_c) &= \frac{1}{2}\left(2\sum_b \pi_b q_b d_b - 2\bar{q}\bar{d}\right) \\ &= C(\lambda).\end{aligned} \tag{A14}$$

If forecasts preserve the hard-to-easy ordering, every product $(q_b - q_c)(d_b - d_c)$ is nonnegative, so $C \ge 0$. Under this ordering, $C > 0$ whenever at least one pair differs strictly in both score and forecast. However, the EMA calibration does not enforce monotonicity, so the covariance can be negative. Zero covariance indicates no first-order reward response; it can arise from identical forecasts, a single active bucket, or cancellation between oppositely ordered pairs.

When $C(\lambda) > 0$, a first-order Taylor expansion of the predicted reward for a small increment h gives

$$m(\lambda + h) \approx m(\lambda) + m'(\lambda)h = m(\lambda) - C(\lambda)h. \tag{A15}$$

Setting this linearized prediction equal to the target reward and solving for h yields

$$m(\lambda) - C(\lambda)h = \tau \quad \Longrightarrow \quad C(\lambda)h = m(\lambda) - \tau \quad \Longrightarrow \quad h = \frac{m(\lambda) - \tau}{C(\lambda)}. \tag{A16}$$

This increment solves the linearized target equation and is the Newton correction for $m(\lambda) - \tau = 0$; the nonlinear reward need not reach τ in one step. A larger positive sensitivity requires a smaller increment to correct the same reward error; a positive error requires increasing λ and shifting probability toward harder buckets.

The implemented controller uses the observed error $\bar{r}_t - \tau$ and evaluates sensitivity on the updated bucket state. After incorporating batch t into the forecasts and bucket membership, it holds λ_t fixed and computes

$$\widetilde{\pi}_{b,t+1} = \pi_{b,t+1}(\lambda_t), \qquad C_{t+1} = \operatorname{Cov}_{b\sim\widetilde{\pi}_{t+1}}(q_{b,t+1}, d_b) = -m_{t+1}'(\lambda_t). \tag{A17}$$

Algorithm 1: Calibrated Dynamic Difficulty Shaping (CalDS)

Input: Training set $\mathcal{D}$ with reference answers; initial policy π_{θ_0}; nonempty initial histories $\{H_{i,0}\}_{i\in\mathcal{D}}$ from tool-enabled base-model rollouts; iterations T; batch size $B \le |\mathcal{D}|$; group size G; history window X; bucket count K; target τ; calibration rate α; odd shaping exponent $\gamma > 1$; controller parameters $\eta_\lambda, \varepsilon, \Delta_{\max}, \lambda_{\max}$.

Initialize:
- Retain the latest X outcomes in each $H_{i,0}$; build $\mathcal{B}_{b,0}$ using Eq. (2).
- Shuffle each bucket's members and place them in a rotating queue.
- Set $c_b = (b + 0.5)/K$ and $d_b = [1 + (2c_b - 1)^\gamma]/2$ for every bucket b.
- Set $q_{b,0}$ to the mean $\hat{p}_{i,0}$ of bucket members, or c_b if the bucket is empty.
- Set $\lambda_0 \leftarrow 0$.

for $t = 0, \ldots, T-1$ **do**
- *Allocate the batch and record source buckets.*
- Find $\mathcal{A}_t = \{b : n_{b,t} > 0\}$; compute $\pi_{b,t}(\lambda_t)$ using Eq. (4).
- Round to integer counts $a_{b,t}$ with $\sum_b a_{b,t} = B$ and $0 \le a_{b,t} \le n_{b,t}$.
- Pop $a_{b,t}$ distinct prompts from each queue to form $S_{b,t}$; freeze these source sets.
- Set $\mathcal{D}_t \leftarrow \bigcup_b S_{b,t}$.
- *Collect tool-augmented trajectories and optimize the policy.*
- Generate G trajectories per $i \in \mathcal{D}_t$ with π_{θ_t} and crop-and-zoom feedback.
- Evaluate final answers to obtain $r_{i,g,t} \in \{0, 1\}$.
- Compute group-relative advantages; update θ_t to θ_{t+1} using Eq. (1).
- Compute $\bar{r}_t \leftarrow (BG)^{-1} \sum_{i\in\mathcal{D}_t} \sum_{g=1}^{G} r_{i,g,t}$.
- *Refresh histories, membership, and source-bucket forecasts.*
- Carry forward histories and assignments for unselected prompts; set $q_{b,t+1} \leftarrow q_{b,t}$.
- **for each** $i \in \mathcal{D}_t$ **do**
  - Append new rewards to $H_{i,t}$ and retain the latest X outcomes as $H_{i,t+1}$.
  - Recompute $b_{i,t+1}$ using Eq. (2).
  - Append i to the tail of its updated bucket's queue.
- **end for**
- Recompute $\mathcal{B}_{b,t+1}$, $n_{b,t+1}$, and $\mathcal{A}_{t+1}$.
- For each b with $a_{b,t} > 0$, update $q_{b,t+1}$ using the frozen $S_{b,t}$ and Eq. (6).
- *Adjust the sampling preference for the next batch.*
- Compute $\widetilde{\pi}_{b,t+1} \leftarrow \pi_{b,t+1}(\lambda_t)$ over $\mathcal{A}_{t+1}$.
- Compute $C_{t+1} \leftarrow \mathrm{Cov}_{b\sim\widetilde{\pi}_{t+1}}(q_{b,t+1}, d_b)$.
- Set $\Delta_t \leftarrow \mathrm{clip}(\eta_\lambda(\bar{r}_t - \tau)/(\max(C_{t+1}, 0) + \varepsilon), -\Delta_{\max}, \Delta_{\max})$.
- Set $\lambda_{t+1} \leftarrow \mathrm{clip}(\lambda_t + \Delta_t, -\lambda_{\max}, \lambda_{\max})$.

end for

Output: Trained policy π_{θ_T}.

Only buckets in $\mathcal{A}_{t+1}$ enter this calculation. Thus, the derivative uses the new state at the old control value, before computing λ_{t+1}. Replacing the predicted reward error in Equation (A16) with the observed error, scaling by η_λ, and regularizing the sensitivity denominator gives the raw increment

$$h_t^{\mathrm{raw}} = \eta_\lambda \frac{\bar{r}_t - \tau}{\max(C_{t+1}, 0) + \varepsilon}. \tag{A18}$$

The factor η_λ scales the correction, and $\varepsilon > 0$ keeps the division finite. Equations (5) then limit the increment to $[-\Delta_{\max}, \Delta_{\max}]$ and the accumulated control value to $[-\lambda_{\max}, \lambda_{\max}]$.

If $C_{t+1} < 0$, the nonnegative clamp preserves the update direction prescribed by the observed reward error, while clipping bounds its magnitude. We did not observe negative covariance in our training experiments.

F Algorithm Procedure

Algorithm 1 summarizes CalDS training. Each iteration collects exactly BG tool-augmented trajectories and reuses their rewards for policy optimization, history updates, bucket calibration, and sampling control. Calibration uses the source buckets recorded before rollout, whereas controller sensitivity uses the updated bucket state at the previous control value λ_t.

G LIMITATION AND FUTURE WORK

Our study focuses on crop-and-zoom tool, one of the most widely studied operations in visual tool use (Su et al., 2026; Zheng et al., 2026; Lai et al., 2026). The difficulty estimation and adaptive sampling mechanisms of CalDS are effective in this setting and are potentially applicable to other visual tools. However, due to limited computational resources, we have not yet extended CalDS to a broader range of visual tools for experimental evaluation. These include image search (Chng et al., 2025; Narayan et al., 2025; Zhang et al., 2026), as well as image drawing and geometric transformations such as rotation and flipping (Dong et al., 2026). Extending CalDS to tasks that require selecting and composing these tools is a promising direction for future work. Such settings introduce variation in tool execution costs, feedback reliability, and interaction length, motivating difficulty estimation and sampling strategies that account for both task success and rollout cost. Evaluating these extensions would establish whether the training efficiency gains observed for crop-and-zoom generalize to more diverse multimodal agents.

H LLM JUDGE PROMPT

The prompt of LLM judge described in Appendix A.1 is as follows. To reduce LLM evaluation costs, we first check the generated answer against the reference answer using text matching. A successful match receives a score of 1; otherwise, we invoke the LLM judge. The placeholders $\{\texttt{question}\}$, $\{\texttt{answer}\}$, and $\{\texttt{response}\}$ are replaced with the question, reference answer, and model-generated solution, respectively. The judge extracts the model's final answer and returns a binary correctness score, allowing semantically equivalent answers and a 5% relative-error tolerance for floating-point answers.

```
You are given a question and its standard answer, your task is to judge if a
model-generated solution is correct.
You first need to extract the final choice made in the model-generated
solution, and then judge if it matches the standard answer.
The <image> placeholder indicates an image in the question, but you do not
need its content.

For multiple choice questions, note that the model-generated solution may not
directly give the option letter, in this case, you need to judge from the
solution's content to find out which option it has chosen.
For floating-point answers, the model-generated final answer needs to be close
enough (5% relative error) to the standard answer to be considered correct.
Note that math expressions can have multiple equivalent forms, e.g., 0.5km and
500m are equivalent.
Other short-text answers can be considered equivalent if they are semantically
equivalent under the context of the question.

Give a score of 1 if the model-generated solution matches the standard answer,
and 0 otherwise.

[[Begin of Question]]
{question}
[[End of Question]]

[[Begin of Standard Answer]]
{answer}
[[End of Standard Answer]]

[[Begin of Model-Generated Solution]]
{response}
[[End of Model-Generated Solution]]

Your output should begin with a concise analysis, mentioning what answer the
Model-Generated Solution has finally given.
Finally give an integer score in the following json format:
```json
{{
    "score": ## an integer, 0 or 1 ##
}}
```
```

I TRAJECTORY CASES

Figures A2–A5 present four visual tool-use trajectories on the VisualProbe benchmark, generated by our model trained with CalDS. These examples illustrate how our model uses intermediate crop results to refine or redirect its visual search, inspecting relevant details before producing a final answer.

Question:
What color is “Mindpearl”?

Response 1:

I need to find the text "Mindpearl" and then determine its color. I will start by scanning the image for any text or logos that might match "Mindpearl" or be associated with it, especially looking for it on buildings or signs in the image, since that's where such text is likely to appear.

I will now use the image_crop_and_zoom_in_tool to examine the image more closely, especially focusing on the buildings and their signage, particularly in the central and lower parts of the image where such details are more likely to be visible.

Crop Tool Call 1:

bbox_2d = [0, 514, 686, 826]

Response 2:

Based on the cropped image, I now focus on the lower central area, especially looking for any text that might be associated with "Mindpearl" or a logo next to it. I will now examine the image more closely, particularly looking for any text or logos along the facades of the buildings, especially in the lower half of the image, where such details are more likely to be visible.

Crop Tool Call 2:

bbox_2d = [326, 744, 564, 826]

Response 3:

Based on the information gathered, I now focus on the specific text "Mindpearl" to inspect its color directly. I will now use the image_crop_and_zoom_in_tool to get an even closer view of the text "Mindpearl" and its associated graphic or color.

Crop Tool Call 3:

bbox_2d = [446, 772, 522, 806]

Answer:

Based on the step-by-step inspection:

- I first scanned the image for "Mindpearl" and narrowed down the location to a specific building in the lower central area.
- Upon zooming into that area, I found "Mindpearl" printed next to a small green icon or logo.
- Upon inspecting the text "Mindpearl" directly, I could clearly see that the text is black or dark gray, and it is next to a green symbol.

Therefore, the color of "Mindpearl" is black or a dark gray.

Original Image:



Crop Result 1:



Crop Result 2:



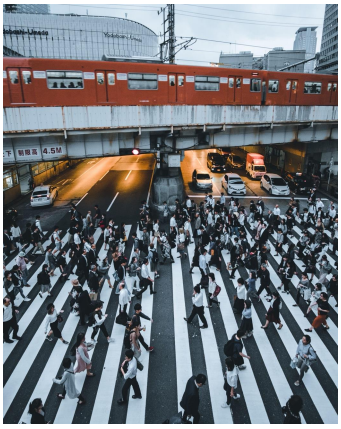
Crop Result 3:



Fig. A2: Visual tool-use trajectory of VisualProbe Benchmark (Hard subset, ID 0).

Question:
What color is the number 27?

Original Image:



Response 1:

I will use the image_crop_and_zoom_in_tool to inspect the image for any text or numbers, especially looking for "27" or any number that might be associated with it, and check its color. I will start by scanning the image for any text or numbers, especially in areas around the vehicles, signs, or along the infrastructure, since such details are often found there.

Crop Tool Call 1:

bbox_2d = [0, 334, 1000, 454]

Crop Result 1:



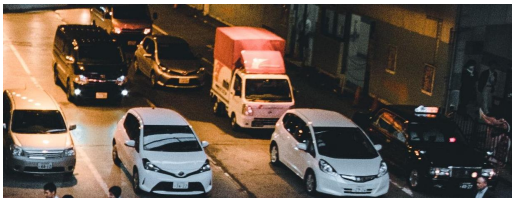
Response 2:

Based on the hint and the need to find the number "27" and its color, I will now focus on the license plates of the cars, as they are a likely place to find such numbers, especially since I noticed a few numbers on the bottom of the cars in the image, particularly on the black car on the far right and others nearby.

Crop Tool Call 2:

bbox_2d = [572, 342, 946, 456]

Crop Result 2:



Response 3:

Based on the images I just inspected, I now focus on the bottom of the black car on the far right, as I spotted what looks like the number "27" on its license plate. I will now examine that specific area more closely to confirm what I saw and to inspect the color of the number "27" precisely.

Crop Tool Call 3:

bbox_2d = [816, 396, 946, 454]

Crop Result 3:



Answer:

Based on the information gathered through the image tool, I now have a clear view of the number "27" on the license plate of the black car. I can now clearly see that the number "27" is printed in white or white/light-colored text on a black or dark-colored background.

Therefore, the color of the number “27” is white or light-colored.

Fig. A3: Visual tool-use trajectory of VisualProbe Benchmark (Hard subset, ID 1).

Question:
What is the number to the left of 1000?

Original Image:



Response 1:

I need to find a number that appears directly to the left of "1000" in the image. I will now use the image_crop_and_zoom_in_tool to examine the image for any text or numbers that might match this description, especially looking for "1000" and what is directly to its left.

Crop Tool Call 1:

bbox_2d = [806, 244, 1000, 386]

Crop Result 1:



Response 2:

Based on the cropped image, I now focus on another region of the image, especially looking for any text or numbers that might be near or next to "1000" or a similar number, since the initial view didn't show what I was looking for directly. I will now examine the image more broadly, especially looking for "1000" or numbers around it, perhaps in the text or phone numbers or other small print, especially in the lower half or along the sides of the image.

Crop Tool Call 2:

bbox_2d = [500, 326, 1000, 526]

Crop Result 2:



Response 3:

Based on the results from the previous steps, I now focus on the yellow and white poster or advertisement at the bottom right, as I noticed a phone number or a sequence of numbers there, which might be what I'm looking for. I will now examine the bottom right of the image more closely, especially looking for "1000" or any number directly next to it, particularly in the text or phone numbers.

Crop Tool Call 3:

bbox_2d = [847, 446, 1000, 526]

Crop Result 3:



Answer:

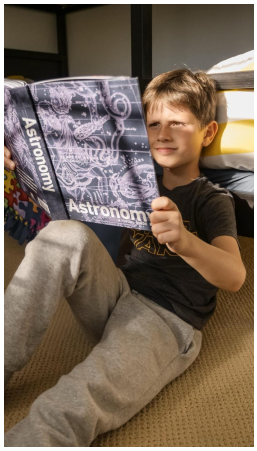
Based on the information gathered, I now see a line of text that reads "Tel: 2563 1000". In this text, "2563" is directly to the left of "1000". This matches the requirement of finding a number directly to the left of "1000".

Therefore, the number to the left of 1000 is 2563.

Fig. A4: Visual tool-use trajectory of VisualProbe Benchmark (Hard subset, ID 17).

Question:
what is the word before "as"?

Original Image:



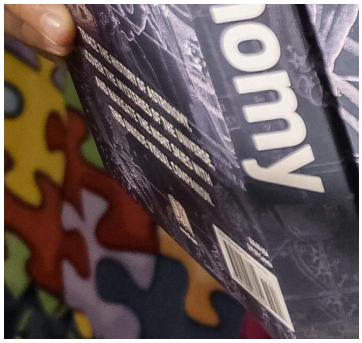
Response 1:

I need to find a specific phrase or text in the image where "as" appears, and then identify the word directly preceding "as". I will now use the image_crop_and_zoom_in_tool to examine the image for any text, especially along the book or any other printed material, looking for "as" or words around it.

Crop Tool Call 1:

bbox_2d = [0, 356, 226, 476]

Crop Result 1:



Response 2:

Based on the cropped image, I now see some white text along the spine and side of the book, but I don't see the word "as" or anything close to it in this specific area. I will now examine the front cover and the rest of the book's text, especially around the bottom and any other text that might be printed in smaller fonts or in different locations.

Crop Tool Call 2:

bbox_2d = [0, 426, 624, 517]

Crop Result 2:



Response 3:

Based on the information I just found, I now see a line of small text at the bottom of the front cover that reads "Previously published as Eyewitness Companions: Astronomy". I now focus on this line to find the word directly before "as" and confirm what I see.

Crop Tool Call 3:

bbox_2d = [176, 444, 616, 506]

Crop Result 3:



Answer:

Based on the step-by-step exploration, I now see the phrase "published as" in the small text at the bottom of the book cover. The word directly before "as" is "published".

Therefore, the word before “as” is “published”.

Fig. A5: Visual tool-use trajectory of VisualProbe Benchmark (Hard subset, ID 59).