DRIVE: DUAL REFLECTION WITH INTERLEAVED VISUAL EVIDENCE FOR EMBODIED POINTING

Anonymous authors
Paper under double-blind review

ABSTRACT

A model may identify the intended visual target yet predict a coordinate that falls elsewhere. We present DRIVE (Dual Reflection with Interleaved Visual Evidence), a two-stage reinforcement-learning framework designed to strengthen the connection between spatial reasoning and coordinate prediction. A shared policy first proposes target points, then reflects on their placement through rendered feedback: it states a spatial expectation before rendering and diagnoses the observed marker afterward. When the diagnosis identifies a mismatch, the policy revises the coordinates and inspects them again. Training uses final localization outcomes to optimize a selected response conditioned on this reflection history. On the 300-example VABench-P test set, the complete protocol achieves 71.67% final-hit accuracy, compared with 67.33% for direct inference with the same trained checkpoint. Inference ablations examine how spatial expectations, placement diagnoses, and rendered markers contribute to the trained policy's predictions.

1 INTRODUCTION

Visual grounding localizes targets in an image based on language instructions (Kazemzadeh et al., 2014; Mao et al., 2015). These targets may be objects, actionable parts, or placement regions satisfying spatial constraints (Yuan et al., 2024; 2025b). In modular robotic systems, the resulting interaction points can guide downstream planning and control (Yuan et al., 2024; 2025b).

Aligning numerical coordinates with their corresponding image locations remains a challenge for vision-language models (Yuan et al., 2025a; Zhao et al., 2025). For example, when asked to *point to the cup handle*, a model may recognize the handle to the right of the cup while returning a point near the rim. Numerical candidates therefore create a specific verification problem: where does the proposed coordinate fall, and does that location satisfy the instruction? Figure 1 illustrates this mismatch between the intended target and the rendered coordinate.

Prior work uses visual aids and rendered feedback to connect coordinate predictions with image locations and support iterative refinement (Nasiriany et al., 2024; Yuan et al., 2025a; Yang & Huang, 2025; Zhao et al., 2025). DRIVE interleaves textual reflection and rendered visual evidence in a two-stage training framework. Stage 1 produces an independent point proposal. Stage 2 pairs a *spatial expectation* before rendering with a *placement diagnosis* afterward, making the relationship between a numerical coordinate and its visible location explicit.

During training rollouts, the policy can revise a point when its observed placement differs from the intended target. The expectation, rendered image, diagnosis, and revised candidate remain in the interaction history. An outcome-based reinforcement-learning objective optimizes the shared policy using a selected response conditioned on this history. This design uses reflection and correction as training context for learning spatially grounded point predictions.

We evaluate the trained model on the 300-example VABench-P test set (Yuan et al., 2025a). The complete protocol achieves 71.67% final-hit accuracy, compared with 67.33% for direct inference using the same checkpoint. We also examine the effects of the two reflection components and the visual marker through inference ablations.

Our main contributions are summarized as follows:

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: **Target understanding and coordinate placement can diverge.** In this schematic example, the intended target is the cup handle, but the predicted coordinate $(820, 180)$ falls near the cup rim. Rendering the coordinate makes the spatial mismatch visible. DRIVE uses expectations and placement diagnoses around such rendered candidates during training. Coordinates are normalized to $[0, 1000]$. The right panel magnifies the boxed region; its dashed outline indicates the target handle.

- **Two-stage reflection training.** We combine independent point proposals with reflection on rendered candidates to connect coordinate prediction with spatial understanding.
- **Expectation–diagnosis interaction.** The policy predicts a candidate's spatial placement, inspects its rendered location, and uses the resulting diagnosis to retain or revise the coordinates.
- **Outcome-based policy optimization and evaluation.** We train the shared policy with group-relative response-level credit and evaluate its visual-pointing performance and sensitivity to reflection components.

2 Related Work

Language-conditioned grounding and pointing. Language-conditioned grounding has been studied through referring-expression datasets (Kazemzadeh et al., 2014; Mao et al., 2015; Yu et al., 2016), structured visual representations (Yu et al., 2018), and transformer-based localization and detection (Deng et al., 2021; Kamath et al., 2021; Liu et al., 2023). Subsequent work develops grounded multimodal representations and spatial understanding (Peng et al., 2023; Wang et al., 2024; Chen et al., 2024; Zhou et al., 2025). Point-based interfaces extend these capabilities to interaction-location prediction, general-purpose visual pointing, and embodied action (Yuan et al., 2024; Deitke et al., 2024; Yuan et al., 2025b; Cheng et al., 2025).

Visual feedback around candidate locations. Prior work associates visible marks with spatial referents (Yang et al., 2023), enables reasoning through drawn artifacts (Hu et al., 2024), and composes executable visual operations (Gupta & Kembhavi, 2022; Surís et al., 2023). Candidate actions or locations can be overlaid for iterative selection (Nasiriany et al., 2024), while self-consistency connects coordinate generation with visual understanding (Yuan et al., 2025a). Poivre trains point

visualization and refinement using a process reward (Yang & Huang, 2025), and GUI-Cursor uses rendered cursor feedback and movement history for grounding (Zhao et al., 2025). DRIVE structures the comparison around a pre-render spatial expectation and a post-render placement diagnosis, using the resulting history as context for outcome-based training.

Interactive reasoning and learning to revise. Interactive approaches interleave reasoning and actions (Yao et al., 2022), learn API use (Schick et al., 2023), or organize iterative verbal feedback (Madaan et al., 2023; Shinn et al., 2023). SCoRe trains self-correction through multi-turn reinforcement learning on self-generated traces (Kumar et al., 2024). Multimodal studies use verifiable perception or grounding rewards (Liu et al., 2025; Bai et al., 2025b) and incorporate visual observation or grounded reasoning (Zheng et al., 2025; Sarch et al., 2025). VAGEN couples state estimation and transition modeling with dense turn-level rewards and turn-aware credit assignment (Wang et al., 2025). DRIVE assigns the localization-outcome advantage to a selected response conditioned on the preceding spatial reflections.

Outcome supervision for spatial reflection. Process supervision evaluates intermediate reasoning steps, whereas outcome supervision evaluates final answers (Lightman et al., 2023). Group-relative optimization and related reinforcement-learning approaches support learning from verifiable episode outcomes (Shao et al., 2024; DeepSeek-AI et al., 2025; Ahmadian et al., 2024). Studies of self-correction examine the role of checking information (Huang et al., 2023; Gou et al., 2023; Cheng et al., 2026) and trajectory selection (Tripathy & Krishnan, 2026). DRIVE uses final localization outcomes to train responses conditioned on expectations, rendered observations, and placement diagnoses, without separately scoring the intermediate reflection text.

3 METHOD

DRIVE trains a shared policy through two-stage rollouts: independent point proposal followed by reflection on rendered candidate locations. We describe the rollout interaction and then the outcome-based policy update.

3.1 TASK AND TWO-STAGE TRAINING ROLLOUTS

We study image–language localization following referring-expression and visual-pointing tasks (Kazemzadeh et al., 2014; Mao et al., 2015; Yu et al., 2016; Yuan et al., 2024; 2025b; Cheng et al., 2025). Given an image I and instruction q, the policy predicts points in the specified object, functional part, or placement region. Episode n returns an ordered list $P_n = [p_{n,1}, \ldots, p_{n,L_n}]$ of L_n points, with normalized image coordinates $p_{n,\ell} = (x_{n,\ell}, y_{n,\ell}) \in [0, 1000]^2$. Single-target instructions request one point. The evaluator maps these coordinates to pixel locations and checks whether they lie within the union M of target masks.

Each *training rollout* has two stages. Stage 1 generates a point proposal P_n^{prop}, and Stage 2 uses an interactive process to produce the final point list P_n^{f}. Both stages use the same policy π_θ with separate contexts. The final point list is evaluated against the target masks to obtain the training reward. Figure 2 illustrates the two-stage rollout and the resulting policy update. At evaluation, the trained policy uses the same two-stage interaction without parameter updates.

3.2 DUAL REFLECTION AROUND CANDIDATE COORDINATES

Independent Point Proposal. Stage 1 uses a dedicated grounding prompt, separating initial localization from reflection and tool-use instructions. The context is then rebuilt from (I, q). Coordinates extracted from a complete `answer` block are serialized as an ordered hint instructing Stage 2 to use them as its initial candidate. Stage 1 reasoning is not transferred. If extraction fails, the hint is omitted and Stage 2 generates a candidate from the task. The list actually submitted for its first rendering is P_n^0. Appendix B specifies the prompts and coordinate interface.

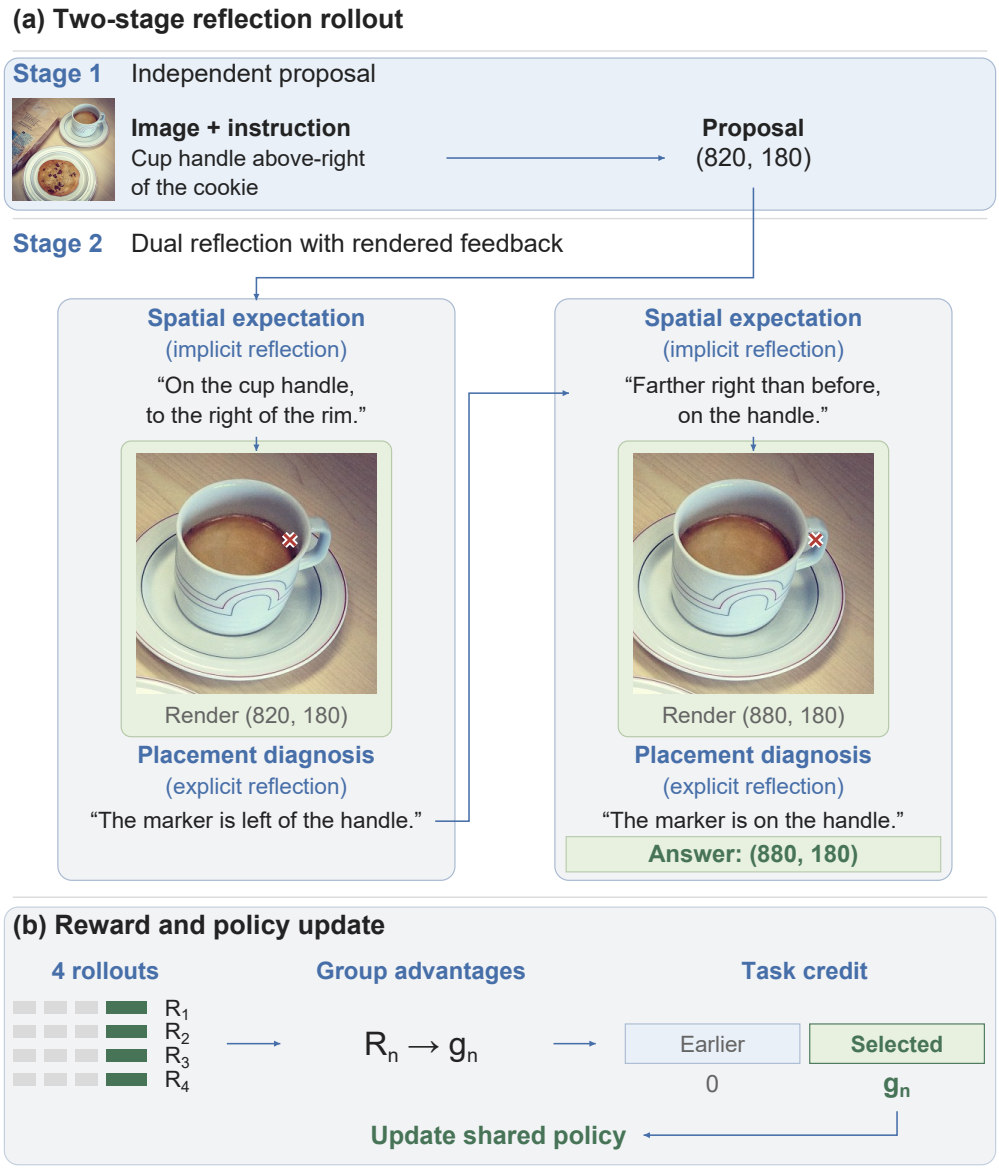


Figure 2: **Two-stage reflection training (schematic).** (a) Stage 1 proposes a point. Stage 2 states a spatial expectation before rendering (implicit reflection), diagnoses the observed placement afterward (explicit reflection), and revises the point when needed. Coordinates are normalized to $[0, 1000]$. (b) Four rollouts per task yield outcome rewards and group-relative advantages. All retained tokens in the selected response receive the task advantage; earlier responses provide context. PPO updates the policy shared by both stages. KL and entropy apply to all retained Stage 2 responses (Section 3.3).

Spatial Expectation Before Rendering. For candidate P_n^k, the policy predicts where each point will appear relative to the target before viewing the current rendering. The `implicit_reflection` field pairs each point with this spatial expectation, making the anticipated relationship between its coordinates and the target explicit. The tool then returns

$$\widetilde{I}_n^k = V(I, P_n^k). \tag{1}$$

Here V overlays markers on the original image. The response includes a generic inspection prompt, without annotation masks, correctness labels, or correction directions.

Placement Diagnosis After Rendering. After observing $\widetilde{I}_n^k$, the policy describes where each marker falls relative to the target and judges whether it satisfies the instruction. The `explicit_reflection` field records this placement diagnosis and explains whether to retain the point or revise its coordinates.

Coordinate Revision. When a candidate is judged incorrect, the policy proposes revised coordinates and repeats the expectation–rendering–diagnosis sequence. The expectation, rendered image, diagnosis, and revised candidate remain in the same interaction history, allowing the policy to compare predicted and observed spatial relations. When all candidates are judged correct, the policy returns the inspected points. Turn and repetition limits bound the rollout; the final point list P_n^{f} is parsed from the returned answer.

3.3 Training

At update cycle i, the sampling policy $\pi_{\bar{\theta}}$, with $\bar{\theta} = \theta_i$, generates four two-stage rollouts per task. After policy optimization, the updated parameters are used by both stages in subsequent rollouts.

Localization Outcome Reward. Point quality is the fraction of predicted points within the target-mask union:

$$Q(P, M) = \frac{1}{|P|} \sum_{p \in P} \mathbf{1}[p \in M], \qquad Q(\varnothing, M) = 0. \tag{2}$$

Points are evaluated independently, without one-to-one matching to target instances. Let $h_n \in \{0, 1\}$ indicate whether the serialized rollout text contains a complete `tool_call` tag pair. The point reward is

$$r_n^{\mathrm{point}} = h_n Q(P_n^{\mathrm{f}}, M). \tag{3}$$

Let $v_n \in \{0, 1\}$ indicate whether the final answer is valid, and let $b_n \in \{0, 1\}$ indicate whether two consecutive tool calls contain identical point lists after rounding. The episode reward is

$$R_n = \begin{cases} r_n^{\mathrm{point}} - \lambda_{\mathrm{reg}} d_n - \lambda_{\mathrm{rep}} b_n, & v_n = 1, \\ -\lambda_{\mathrm{miss}} - \lambda_{\mathrm{rep}} b_n, & v_n = 0, \end{cases} \tag{4}$$

where

$$d_n = \mathbf{1}[Q(P_n^{\mathrm{prop}}, M) > 0 \ \wedge \ r_n^{\mathrm{point}} = 0]. \tag{5}$$

The proposal-reference penalty discourages losing a hit from the independent proposal at a valid endpoint. We use $\lambda_{\mathrm{reg}} = \lambda_{\mathrm{miss}} = 0.10$ and $\lambda_{\mathrm{rep}} = 0.05$. Rewards are computed after sampling and are not supplied as intermediate correctness feedback.

Group-Relative Task Advantage. Following within-group outcome normalization (Shao et al., 2024), episode n in task group $\mathcal{G}(n)$ receives

$$g_n = \frac{R_n - \mathrm{mean}_{a \in \mathcal{G}(n)} R_a}{\mathrm{std}_{a \in \mathcal{G}(n)} R_a + \varepsilon}. \tag{6}$$

Here, $\mathcal{G}(n)$ contains episodes sampled for the same task, and ε ensures numerical stability.

Credit Assignment to a Selected Turn. Let $o_{n,t}$ identify the response turn containing token t. Select $o_n^\star$ as the last nonempty response matching the loop's answer pattern, falling back to the last nonempty generated response. For retained Stage 2 response mask $m_{n,t}^{\mathrm{resp}}$, define

$$m_{n,t}^{\mathrm{credit}} = m_{n,t}^{\mathrm{resp}} \mathbf{1}[o_{n,t} = o_n^\star], \qquad A_{n,t}^{\mathrm{task}} = m_{n,t}^{\mathrm{credit}} g_n. \tag{7}$$

All retained tokens in the selected response receive the same task advantage g_n, including reasoning, fields, and answer tokens. Other responses provide context and receive zero direct task advantage. Stage 1 and tool-returned tokens are excluded from this optimization sequence. Both stages use the updated shared parameters on subsequent rollouts. Appendix C specifies token selection and loss reduction.

Policy Optimization Objective. For generated token $y_{n,t}$ and the context $c_{n,t}$ available when it was sampled, define the policy probability ratio

$$\rho_{n,t}(\theta) = \frac{\pi_\theta(y_{n,t} \mid c_{n,t})}{\pi_{\bar{\theta}}(y_{n,t} \mid c_{n,t})}. \tag{8}$$

We minimize a dual-clipped PPO task loss (Schulman et al., 2017; Ye et al., 2019), with fixed-reference KL regularization and entropy:

$$\mathcal{L}(\theta) = \mathbb{E}_{m^{\text{resp}}}[\ell_{\text{dc}}(\rho_{n,t}(\theta), A^{\text{task}}_{n,t})] + \beta_{\text{KL}}\mathcal{L}_{\text{KL}} - \beta_{\text{ent}}\mathcal{H}(\pi_\theta). \tag{9}$$

The objective is averaged over retained Stage 2 response tokens, including those with zero task advantage. The KL reference policy remains fixed throughout training, whereas the sampling policy is refreshed between update cycles. Appendices A and C provide implementation details.

4 Experiments

We evaluate the visual-pointing performance of the trained DRIVE policy and the roles of spatial expectation, placement diagnosis, and rendered feedback.

4.1 Experimental Setup

Data and model. We first equip Qwen3-VL-8B (Bai et al., 2025a) with basic visual grounding capabilities following the training procedure of Embodied-R1 (Yuan et al., 2025b). We then perform reinforcement learning with DRIVE on 10,000 examples sampled from the Embodied-R1 dataset. Checkpoint selection uses an independent validation split held out from the training data. We evaluate on the 300-example VABench-P test set (Yuan et al., 2025a). Appendix D.1 describes data preparation and target annotations.

Implementation. We implement DRIVE using `verl` (Sheng et al., 2024). All DRIVE inference conditions (B0 and A1–A5) use the step-20 checkpoint selected on the held-out validation split, with temperature-zero decoding and one episode per test example. Training hyperparameters, interaction budgets, and decoding settings are provided in Appendix C.4.

Inference conditions. We compare the complete protocol (B0) with direct prediction (A5) and four inference ablations: removing the visual marker (A1), spatial expectation (A2), placement diagnosis (A3), or both reflection components (A4). A1 retains the original image, coordinates, and inspection instructions. A2–A4 alter the corresponding prompt bundles and output requirements. A5 generates a direct answer without tools. Each condition is evaluated separately; B0 and A1–A4 generate their own Stage 1 proposals. Appendix B.6 specifies the interventions.

Metrics. We report final-hit rate: the fraction of examples with a positive evaluator point reward, based on any predicted point lying in the target region under the condition's evaluation rules. Terminal validity and output cardinality are auxiliary diagnostics. Training instead rewards the fraction of predicted points that hit. Appendix D.2 defines the evaluation rules. Differences are accompanied by 95% paired-bootstrap intervals over the same 300 examples. Appendix D specifies evaluation and resampling details.

Table 1: **Visual-pointing results and inference ablations.** B0 and A1–A5 use the same trained checkpoint on 300 test examples; Embodied-R1 is a published reference result. Initial hit, final hit, and terminal validity are percentages. The difference and 95% paired-bootstrap interval are percentage points for B0 minus the comparison condition.

Condition	Initial	Final	Valid	Difference [95% CI]
B0: complete interaction	68.67	**71.67**	100.00	–
Embodied-R1 (reported) (Yuan et al., 2025b)	–	66.00	–	–
A5: single-turn direct	67.33	67.33	95.33	4.33 [1.67, 7.33]
A1: no visual marker	67.33	70.33	99.33	1.33 [−1.00, 3.67]
A2: no expectation	70.33	45.33	90.33	26.33 [21.00, 31.67]
A3: no diagnosis	68.33	66.67	96.00	5.00 [2.00, 8.33]
A4: neither reflection	68.67	67.33	99.00	4.33 [1.33, 7.67]

4.2 Main Results

DRIVE achieves 71.67% final-hit accuracy (215/300) on VABench-P, compared with the 66.00% reported for Embodied-R1 (Yuan et al., 2025b) on this benchmark. This corresponds to a difference of 5.67 percentage points.

For the same DRIVE checkpoint, single-turn direct inference achieves 67.33% (202/300), while the complete two-stage protocol achieves 71.67%, an improvement of 4.33 percentage points with a 95% paired-bootstrap confidence interval of [1.67, 7.33] (Table 1).

4.3 Inference Ablations

Reflection components. Removing spatial expectation reduces final-hit rate to 45.33%, while removing placement diagnosis gives 66.67%. Removing both components yields 67.33%, higher than removing expectation alone, indicating sensitivity to the combined reflection prompt and output schema.

Visual marker. Removing the overlay gives 70.33% final hits, a difference of 1.33 percentage points from B0, with a paired interval of [−1.00, 3.67]. The interval includes zero; the estimated marker effect is smaller than the effects of the reflection-prompt interventions.

Preservation of initial hits. Table 2 separates preserved hits, lost hits, and improved outcomes within each condition. Without spatial expectation, the initial-hit rate is 70.33%, yet 80 of the 211 initially hitting examples end without a final hit. The complete protocol loses only one of its 206 initial hits, while A3 and A4 lose 12 and nine, respectively. These transitions compare recorded proposal and final-output hits, including the effects of answer validity and point-list cardinality.

Table 2: **Initial-to-final result transitions across inference conditions.** H denotes a recorded hit and M a non-hit, including missing answers. Each row contains 300 examples and compares that condition's own initial proposal with its final output; proposals are generated independently across conditions. A5 is omitted because its initial and final outputs are identical by construction.

Condition	H→H	H→M	M→H	M→M
B0: complete interaction	205	**1**	10	84
A1: no visual marker	200	2	11	87
A2: no expectation	131	80	5	84
A3: no diagnosis	193	12	7	88
A4: neither reflection	197	9	5	89

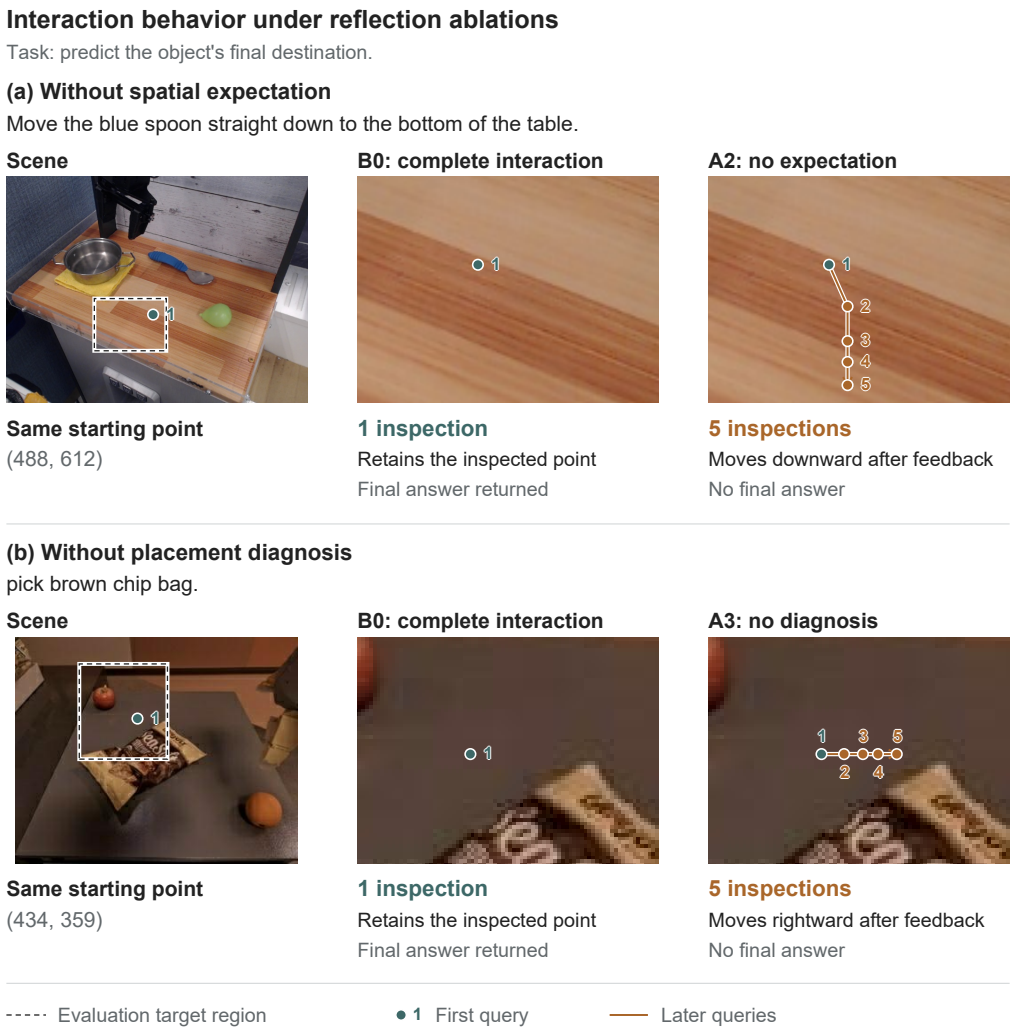


Figure 3: **Interaction behavior under reflection ablations.** Each task asks for the object's final destination. With the same checkpoint and initial proposal, B0 inspects and retains the point; A2 (no expectation) queries progressively downward, and A3 (no diagnosis) rightward. Each ablated trajectory makes five calls without a final answer. Numbers mark query order in matching detail views. All queries lie within the dashed evaluation target region. Markers use logged coordinates from rows 48 and 298 (zero-based).

4.4 Behavior Analysis

Realized interaction. B0 makes 303 parsed tool calls over 300 episodes, averaging 1.01 calls per example. The final list matches the first rendered list in 297 cases. Most B0 episodes therefore use a single inspection. Appendix E.2 reports output cardinalities, and Appendix D.5 analyzes agreement between placement diagnoses and endpoint hits.

Qualitative examples. Figure 3 illustrates how reflection ablations change the decision to retain or revise an inspected point. In these matched examples, B0 retains its initial candidate while the ablated policy continues moving it after feedback. Additional B0 examples appear in Appendix E.1.

5 DISCUSSION

DRIVE connects a numerical coordinate to its visual referent through a pre-render expectation and a post-render diagnosis. Coordinate revision repeats this comparison at a new location. During training, these observations and reflections form the context for the response receiving the localization reward.

The complete interaction achieves the highest final-hit rate among the tested inference conditions. The ablations show that both reflection components affect the trained policy's predictions. Broader spatial-reasoning benchmarks and matched training ablations would further evaluate how this training design affects generalization and the contribution of each learning component.

6 CONCLUSION

DRIVE is a two-stage reflection-training framework for visual pointing. It combines independent coordinate proposals with spatial expectations, rendered observations, placement diagnoses, and coordinate revision in a shared policy. Outcome-based reinforcement learning trains the policy on responses conditioned on these spatial interactions. On VABench-P, the complete protocol achieves 71.67% final-hit accuracy, exceeding direct inference with the same trained checkpoint by 4.33 percentage points.

AI USE STATEMENT

OpenAI Codex assisted with manuscript editing and experimental analysis, including interpretation of results. The authors take responsibility for the final text, analyses, and claims.

REPRODUCIBILITY STATEMENT

Appendices A–E provide the rollout protocol and prompts, optimization objective, training and decoding settings, evaluation procedure, and trajectory analyses.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, et al. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. *arXiv:2402.14740*, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, et al. Qwen3-VL Technical Report. *arXiv:2511.21631*, 2025a.

Sule Bai, Mingxing Li, Yong Liu, Jing Tang, Haoji Zhang, Lei Sun, et al. UniVG-R1: Reasoning Guided Universal Visual Grounding with Reinforcement Learning. *arXiv:2505.14231*, 2025b.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, et al. SpatialVLM: Endowing Vision-Language Models with Spatial Reasoning Capabilities. *arXiv:2401.12168*, 2024.

Long Cheng, Jiafei Duan, Yi Ru Wang, Haoquan Fang, Boyang Li, Yushan Huang, et al. PointArena: Probing Multimodal Grounding Through Language-Guided Pointing. *arXiv:2505.09990*, 2025.

Yu Cheng, Arushi Goel, and Hakan Bilen. Think, Look, and Revise: Inconsistency-Aware Visual Self-Correction in MLLMs. *arXiv:2608.29374*, 2026.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv:2501.12948*, 2025.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, et al. Molmo and PixMo: Open Weights and Open Data for State-of-the-Art Vision-Language Models. *arXiv:2409.17146*, 2024.

Jiajun Deng, Zhengyuan Yang, Tianlang Chen, Wengang Zhou, and Houqiang Li. TransVG: End-to-End Visual Grounding with Transformers. *arXiv:2104.08541*, 2021.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, et al. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. *arXiv:2305.11738*, 2023.

Tanmay Gupta and Aniruddha Kembhavi. Visual Programming: Compositional visual reasoning without training. *arXiv:2211.11559*, 2022.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, et al. Visual Sketchpad: Sketching as a Visual Chain of Thought for Multimodal Language Models. *arXiv:2406.09403*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, et al. Large Language Models Cannot Self-Correct Reasoning Yet. *arXiv:2310.01798*, 2023.

Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. MDETR – Modulated Detection for End-to-End Multi-Modal Understanding. *arXiv:2104.12763*, 2021.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to Objects in Photographs of Natural Scenes. *Proceedings of EMNLP*, 2014. URL https://aclanthology.org/D14-1086/.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, et al. Training Language Models to Self-Correct via Reinforcement Learning. *arXiv:2409.12917*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, et al. Let's Verify Step by Step. *arXiv:2305.20050*, 2023.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, et al. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. *arXiv:2303.05499*, 2023.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, et al. Visual-RFT: Visual Reinforcement Fine-Tuning. *arXiv:2503.01785*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, et al. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv:2303.17651*, 2023.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. Generation and Comprehension of Unambiguous Object Descriptions. *arXiv:1511.02283*, 2015.

Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, et al. PIVOT: Iterative Visual Prompting Elicits Actionable Knowledge for VLMs. *arXiv:2402.07872*, 2024.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, et al. Kosmos-2: Grounding Multimodal Large Language Models to the World. *arXiv:2306.14824*, 2023.

Gabriel Sarch, Snigdha Saha, Naitik Khandelwal, Ayush Jain, Michael J. Tarr, Aviral Kumar, et al. Grounded Reinforcement Learning for Visual Reasoning. *arXiv:2505.23678*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, et al. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv:2302.04761*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, et al. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv:2409.19256*, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv:2303.11366*, 2023.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual Inference via Python Execution for Reasoning. *arXiv:2303.08128*, 2023.

Animesh Tripathy and Aswanth Krishnan. Iterative Visual Thinking and the Self-Correction Mirage in VLM Grounding. *arXiv:2606.13156*, 2026.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, et al. VAGEN: Reinforcing World Model Reasoning for Multi-Turn VLM Agents. *arXiv:2510.16907*, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, et al. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv:2409.12191*, 2024.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V. *arXiv:2310.11441*, 2023.

Wenjie Yang and Zengfeng Huang. Poivre: Self-Refining Visual Pointing with Reinforcement Learning. *arXiv:2509.23746*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, et al. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv:2210.03629*, 2022.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, et al. Mastering Complex Control in MOBA Games with Deep Reinforcement Learning. *arXiv:1912.09729*, 2019.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. Modeling Context in Referring Expressions. *arXiv:1608.00272*, 2016.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, et al. MAttNet: Modular Attention Network for Referring Expression Comprehension. *arXiv:1801.08186*, 2018.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, et al. RoboPoint: A Vision-Language Model for Spatial Affordance Prediction for Robotics. *arXiv:2406.10721*, 2024.

Yifu Yuan, Haiqin Cui, Yibin Chen, Zibin Dong, Fei Ni, Longxin Kou, et al. From Seeing to Doing: Bridging Reasoning and Decision for Robotic Manipulation. *arXiv:2505.08548*, 2025a.

Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, et al. Embodied-R1: Reinforced Embodied Reasoning for General Robotic Manipulation. *arXiv:2508.13998*, 2025b.

Yu Zhao, Wei-Ning Chen, Huseyin Atahan Inan, Samuel Kessler, Lu Wang, Lukas Wutschitz, et al. Learning GUI Grounding with Spatial Reasoning from Visual Feedback. *arXiv:2509.21552*, 2025.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, et al. DeepEyes: Incentivizing "Thinking with Images" via Reinforcement Learning. *arXiv:2505.14362*, 2025.

Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, et al. RoboRefer: Towards Spatial Referring with Reasoning in Vision-Language Models for Robotics. *arXiv:2506.04308*, 2025.

A Policy Optimization Details

Using the probability ratio in Equation (8), write $\rho = \rho_{n,t}(\theta)$ and $A = A^{\text{task}}_{n,t}$, and define $L_1 = -\rho A$ and $L_2 = -\operatorname{clip}(\rho, 1-\epsilon, 1+\epsilon)A$. The dual-clipped task loss (Schulman et al., 2017; Ye et al., 2019) is

$$\ell_{\text{dc}}(\rho, A) = \begin{cases} \max(L_1, L_2), & A \geq 0, \\ \min(-cA, \max(L_1, L_2)), & A < 0. \end{cases} \tag{10}$$

The clipping parameters are $\epsilon = 0.2$ and $c = 3$. The task advantage is treated as a stop-gradient coefficient. The KL and entropy coefficients are both 0.001. Appendix C specifies the numerical estimators and microbatch reduction.

B Prompts and Interaction Protocol

B.1 Isolated Proposal and Context Construction

Stage 1 uses the following grounding prompt (the model-identity phrase is omitted):

```
You are a vision-language model for embodied point grounding.

Given an image and a user instruction, provide one or more 2D point
coordinates for the described target region. Think about the reasoning
process first, then provide the final answer.

The reasoning process and answer must be enclosed within <thinking>
</thinking> and <answer> </answer> tags. The answer consists only of
coordinate points in this format:
<thinking> reasoning process here </thinking><answer><point>[[x1, y1], [
    x2,
y2], ...]</point></answer>
```

The user message contains the image followed by the task question. Proposal generation inherits the episode's sampling parameters and sets a separate maximum of 1,024 generated tokens. Training repeats each task four times before invoking the agent loop, so each episode samples its own proposal and continuation. Evaluation requests one episode per example. The sampling settings for each phase appear in Appendix C.4.

The proposal helper reads the first complete `answer` block and, when present, its nested `point` block. It accepts a numeric pair or a list of pairs; malformed list entries are skipped. Extracted coordinates are rounded to integers and serialized as an ordered hint instructing the actor to use them as its initial candidate and call the tool before answering. With no extracted points, the hint is omitted and the actor generates a candidate from the image and question. The actor produces P_n^0 from this rebuilt context.

The rebuilt context contains the global interaction prompt, original image, question, initial-turn instructions, optional coordinate hint, and tool schema. Proposal reasoning is not copied into this context or optimized as part of the continuation.

B.2 Global Interaction Contract

Experiments use Chinese global, initial-turn, and post-tool prompts. The English specification below preserves their field names and interaction rules, including the ablation variants.

The global prompt asks the actor to localize points satisfying the image and question, inspect them with tool feedback, and enclose its final answer in `<answer>`. Coordinates are normalized integers between 0 and 1,000. A single-target placement task requests one destination point; multiple points are requested only for multiple independent targets. Point count and order are to remain consistent across fields. The prompt permits only these XML sections:

```
<thinking> <relation> <box> <affordance> <tool_call>
<explicit_reflection> <implicit_reflection> <conclusion> <answer>
```

`thinking` briefly states task understanding, the current point list, or the next action. `relation` contains a list of object relations; `box` contains region dictionaries with `bbox_2d` and `label`. Either may contain `[]` when unhelpful. `affordance` and `answer` contain only ordered point lists. `conclusion` briefly summarizes the diagnosis and next step. The reflection schemas are:

```
<implicit_reflection>
[
  {"point": [x, y], "expectation": "Expected placement of this point"}
]
</implicit_reflection>

<explicit_reflection>
[
  {
    "point": [x, y],
    "state": "correct",
    "image_analyze": "Observed placement in the latest feedback image",
    "candidate": [x_next, y_next],
    "candidate_reason": "Reason to retain or update this point"
  }
]
</explicit_reflection>
```

The expectation list is point-aligned and may contain only `point` and `expectation`; it must not contain a correctness state or revised candidate. The diagnosis list is aligned with the latest submitted points. Its `state` must be `correct` or `incorrect`, and its image analysis must describe the latest feedback rather than copy the pre-render expectation. These states are predictions by the actor.

B.3 Initial-Turn and Post-Tool Templates

On the original image, the initial-turn prompt requires nonempty sections in this order:

```
<thinking>...</thinking>
<relation>...</relation>
<box>...</box>
<affordance>[[x, y], ...]</affordance>
<implicit_reflection>...</implicit_reflection>
<tool_call>{"name":"draw_point_and_feedback",
            "arguments":{"points":[[x,y], ...]}}</tool_call>
```

It prohibits an initial `answer`, requires normalized integer coordinates, and requires tool-call points to equal the candidate list. It reiterates the single-destination constraint for single-target tasks.

After feedback, the local prompt requests a fresh diagnosis using the latest feedback image. The next assistant response must begin with `explicit_reflection`, `conclusion`, and `thinking`, in that order. It then selects exactly one branch:

```
If any diagnosed point has state "incorrect":
  <affordance>ordered list of diagnosis candidates</affordance>
  <implicit_reflection>new point-aligned expectations</
    implicit_reflection>
  <tool_call>draw the new candidate list</tool_call>
  Do not emit <answer>.

If every diagnosed point has state "correct":
  <answer>the same ordered points just diagnosed</answer>
  Do not emit new candidates, expectations, or tool calls.
```

The global prompt supplies schematic examples of these two branches using candidate $(780, 620)$ and revision $(792, 615)$.

B.4 Renderer and Coordinate Parsing

The drawing tool receives the original image and a normalized coordinate list through the required `points` argument of `draw_point_and_feedback`. For image dimensions (W, H), it computes

$$(u, v) = \left(\text{round } \frac{xW}{1000}, \text{round } \frac{yH}{1000} \right) \tag{11}$$

and clips the drawing location to $[0, W-1] \times [0, H-1]$. The coordinate labels retain the submitted values, rounded to integers. The strict-mask evaluator instead decodes to mask dimensions (W_M, H_M) as

$$(u_M, v_M) = \left(\text{round } \frac{x(W_M - 1)}{1000}, \text{round } \frac{y(H_M - 1)}{1000} \right). \tag{12}$$

It counts decoded out-of-bounds points as misses without clipping them onto the mask.

The tool rejects empty lists, malformed pairs, and lists longer than 32, returning an error message without a marked image. It draws a circle of radius seven pixels with a crosshair, cycles colors across points, and adds coordinate labels. Each rendering starts from the original image. The response contains the marked image and a generic inspection instruction.

The final-answer parser reads the first complete `answer` block in the scored assistant text. It accepts numeric pairs or lists of pairs, optionally enclosed in `point` tags. Malformed or empty outputs yield zero point quality. The proposal reward is computed from the stored Stage 1 answer.

B.5 Interaction Budgets

Both the training launcher and the evaluation-only runner configure at most five user turns in Stage 2, one processed tool call per assistant turn, and a 6,000-token continuation cap. The user-turn counter starts at zero and increments for a tool response or an interaction follow-up. The continuation cap covers the concatenated Stage 2 response region, including inserted feedback positions whose optimization mask is zero. The final returned token arrays are sliced to that cap. The independent Stage 1 request has its own 1,024-token cap. Table 4 separates these shared budgets from phase-specific sampling.

Consecutive parsed drawing calls are compared as ordered, rounded integer tuples. On the second consecutive identical list, it sets the episode-level repeat flag and suppresses further tool execution. In the interaction-follow-up branch, the user-turn guard requests an answer when at least four user turns have already been consumed. Its answer-only instruction is:

```
You must output <answer>[[x,y]]</answer> now. Tool calls are disabled.
```

Tool calls generated after this instruction are ignored. Consecutive tool calls can exhaust the turn budget before this branch is reached, ending the rollout without a final answer. The final-answer parser checks coordinate syntax; the prompt specifies consistency with the inspected points.

B.6 Inference Ablation Implementation

The no-marker condition changes the rendered image while preserving image access, the tool interface, and inspection text. The no-expectation, no-diagnosis, and neither-reflection variants each replace a bundle comprising global, initial-turn, and post-tool instructions. The no-expectation variant retains task reasoning, relation, region, candidate, and diagnosis fields while excluding the expectation section both initially and on later tool turns. The no-diagnosis variant removes the post-render diagnosis requirements; the neither-reflection bundle removes both families of fields. Direct prediction replaces the entire interaction with a single image-and-query response.

B0 and A1–A4 share the Stage 1 grounding routine before applying condition-specific Stage 2 instructions. Each condition generates its own proposals using the decoding settings in Table 4. Initial-hit columns and transition statistics use these proposals. A5 uses its direct output for both initial and final diagnostics.

C Training and Optimization Details

C.1 Training Cycle and Parameter Roles

At update cycle i, let θ_i be the current actor parameters and set $\bar{\theta} = \theta_i$ during rollout. The same serving actor supplies the isolated proposals and samples the interactive continuations. After rollout, the trainer evaluates rewards, builds turn masks and group-relative advantages, and computes old-policy log-probabilities before optimizing the shared actor. A fixed reference policy supplies the KL term.

Algorithm 1 One DRIVE rollout and update cycle

Require: Actor θ_i, task batch, evaluator-side annotations
1: Set $\bar{\theta} \leftarrow \theta_i$ for rollout and old-policy scoring
2: Repeat each task four times to form an episode group
3: For each episode, generate an isolated proposal and extract coordinate hints
4: Sample expectation–rendering–diagnosis continuations within the budgets
5: Record generated-turn spans, response masks, and terminal metadata
6: Compute protocol-qualified point rewards and Eq. (4)
7: Normalize rewards within prompt groups using Eq. (6)
8: Select credit turns and construct detached advantages using Eq. (7)
9: Compute old log-probabilities under the rollout-visible contexts
10: Optimize the shared actor using Eq. (9)
11: **return** updated actor θ_{i+1}

C.2 Credit Selection and Reduction

For Equation (6), each task group contains four episodes. We use the sample standard deviation and $\varepsilon = 10^{-6}$, with no additional normalization across responses.

The credit turn is the last nonempty generated response containing a complete `answer` tag pair, or the last nonempty response if no such pair is found. Response tokens and turn masks are truncated to the continuation cap. If truncation removes the selected turn, selection falls back to the last active turn and then to all retained response positions. An empty response receives zero task advantage.

All retained tokens in the chosen turn share the trajectory coefficient. Original prompt and inserted tool/follow-up tokens have no generated-response mask. Losses use `token-mean` aggregation. For microbatch $\mathcal{B}_j$, define

$$\mathrm{TM}_{\mathcal{B}_j}(z) = \frac{\sum_{n \in \mathcal{B}_j, t} m^{\mathrm{resp}}_{n,t} z_{n,t}}{\sum_{n \in \mathcal{B}_j, t} m^{\mathrm{resp}}_{n,t} + 10^{-8}}. \tag{13}$$

The task, KL, and entropy terms each use this operator. The denominator includes all retained response tokens, including those with zero task advantage. With fixed microbatches, gradients accumulate from $K^{-1} \sum_{j=1}^{K} \mathcal{L}_{\mathcal{B}_j}$ for each local PPO minibatch, followed by distributed gradient reduction. Local minibatches contain eight responses and microbatches contain two, giving $K = 4$. Dynamic batching is disabled.

The fixed reference is initialized from the same model as the starting actor.

C.3 Numerical Policy Objective

For current and old token log-probabilities ℓ^θ and ℓ^{old}, the ratio is implemented as

$$\rho = \exp(\text{clip}(\ell^\theta - \ell^{\text{old}}, -20, 20)). \tag{14}$$

The dual-clipped task loss and its clipping constants are given in Appendix A. For reference-policy log-probability ℓ^{ref}, the configured numerical KL estimator is

$$\delta = \text{clip}(\ell^{\text{ref}} - \ell^\theta, -20, 20), \qquad \mathcal{L}_{\text{KL}} = \mathbb{E}_{m^{\text{resp}}}[\text{clip}(e^\delta - \delta - 1, -10, 10)]. \tag{15}$$

The actor performs one PPO epoch using stored old-policy log-probabilities and clips the gradient norm at 1.0.

C.4 Training Configuration

Table 3: **Training settings.** Phase-specific decoding and shared episode budgets appear in Table 4.

Setting	Value
RL training examples / test examples	10,000 / 300
Prompts per batch / responses per prompt	48 / 4
Learning rate	10^{-6}
Launcher PPO minibatch / microbatch per GPU	8 / 2
PPO epochs / clipping / dual clipping	1 / 0.2 / 3
KL coefficient / entropy coefficient	0.001 / 0.001
Prompt cap / continuation cap	10,240 / 6,000 tokens
Isolated-proposal cap	1,024 tokens
Model context cap	40,960 tokens
Training temperature / data sampler seed	1.0 / 1
Loss aggregation / gradient norm cap	Token mean / 1.0
Updates / evaluation interval	100 / 20
Validation-selected checkpoint	Step 20
Nodes / GPUs per node	2 / 8
Tensor / sequence parallelism	4 / 4
Degradation / missing / repeat penalty	0.10 / 0.10 / 0.05

The step-20 checkpoint is selected using an independent validation split held out from the training data.

Table 4: **Training and evaluation episode settings.** Both stages share the phase-specific decoding parameters.

Setting	Training	Evaluation
Episodes per task	4	1
Temperature	1.0	0
Top-p / top-k	1.0 / -1	1.0 / -1
Stage 1 generated-token cap	1,024	1,024
Stage 2 continuation cap	6,000	6,000
Stage 2 maximum user turns	5	5
Separate assistant-turn cap	Unset	Unset
Processed tool calls per assistant turn	1	1

C.5 Reward Evaluation

The evaluator computes point quality from the target-mask union. The indicator h_n in Eq. (3) matches complete `tool_call` tag pairs in serialized rollout text, including serialized instructions; it is a text-level gate. The direct-prediction condition bypasses this gate. Final-answer validity v_n is determined by terminal metadata and coordinate parsing. Empty or unparseable answers receive zero point quality, and invalid endpoints receive the missing-answer penalty in Eq. (4). The repeated-call penalty is applied once per rollout. Intermediate reflection descriptions receive no separate reward.

D Evaluation Details and Trajectory Analysis

D.1 Datasets and Annotation Geometry

The RL training set contains 10,000 examples sampled from the referring-expression grounding (REG) subset of Embodied-R1. An independent validation split held out from the training data is used to select the step-20 checkpoint. Training polygons are converted to enclosing bounding boxes and rasterized into rectangular masks. Test evaluation uses the supplied target masks of the 300-example VABench-P test set.

D.2 Hit, Validity, and Output States

The geometric any-point diagnostic is

$$h(P, M) = \mathbf{1}[Q(P, M) > 0]. \tag{16}$$

For singleton outputs this coincides with point quality; for multiple points, it measures whether any point hits, while training uses the fraction of points that hit. The reported final-hit label is $\mathbf{1}[r_n^{\text{point}} > 0]$ for B0 and A1–A4, and $h(P_n^{\text{f}}, M)$ for direct prediction (A5). Terminal validity is a separate syntax diagnostic. Appendix C defines the rollout-text gate.

Let N_{ab} count records with initial state a and final state b, where $a, b \in \{H, M\}$. The output recovery and regression rates are

$$\text{Recovery} = \frac{N_{MH}}{N_{MH} + N_{MM}}, \qquad \text{Regression} = \frac{N_{HM}}{N_{HH} + N_{HM}}. \tag{17}$$

For a common record cohort, the consistency identity is

$$\text{Hit}_f - \text{Hit}_{\text{prop}} = (N_{MH} - N_{HM})/N. \tag{18}$$

H and M denote recorded hits and non-hits, including unavailable answers. States compare the independent proposal with the final output. A5 uses the same direct output for both states.

D.3 Pairing and Bootstrap Computation

We pair the 300 test examples across all six conditions by task question and ordered target-mask paths.

For each condition a, define the paired difference $d_i = h_i^{B0} - h_i^a$. We sample 300 row indices with replacement 10,000 times using NumPy's `default_rng(0)`, reinitializing the generator for each comparison. The mean difference and the 2.5/97.5 percentiles of resampled mean differences yield Table 1. The same resampled indices are applied to both conditions. Intervals quantify paired example-level uncertainty at the selected checkpoint.

D.4 Trajectory Parsing

Trajectory analysis separates the saved Stage 1 proposal from the interaction and groups turns by role. We extract the proposal's complete `answer` block when present, then parse `draw_point_and_feedback` requests, final-answer JSON, and reflection JSON from assistant turns. Point-list equality requires identical cardinality, ordering, and coordinate values.

D.5 Terminal-Diagnosis Analysis

We parse the last assistant diagnosis and final answer, selecting records where (i) the answer has one point, (ii) the diagnosis has one item, (iii) the diagnosis point equals the final point and the last called point, and (iv) the state is either `correct` or `incorrect`. We compare that state with the stored final-hit field. B0 selects 278 records; 22 are excluded by the singleton requirement. A1 selects 280; 18 fail the singleton requirement and two lack a necessary parsed component. All selected records have valid terminals.

Table 5: **Last-diagnosis agreement with singleton endpoint hits.** Counts use the selection rules above and the evaluator's endpoint labels.

Property	B0	A1: no marker
Aligned singleton endpoints	278	280
Diagnosed correct, recorded hit	196	194
Diagnosed correct, recorded miss	82	86
Diagnosed incorrect, recorded hit	0	0
Diagnosed incorrect, recorded miss	0	0
Agreement (%)	70.50	69.29

Every selected terminal diagnosis is positive, consistent with the prompt's instruction to answer after a positive judgment. Agreement with the evaluator's endpoint labels is 70.50% for B0 and 69.29% for A1.

E Additional Results and Trajectories

E.1 Additional Qualitative Examples

Figure 4 shows the policy's spatial expectations and placement diagnoses for two visual-pointing tasks, together with their evaluated final points.

(a) Lid onto pot



Expectation: "Expected to fall in the opening of the silver pot."
Diagnosis: state = correct; marker described as inside the opening.
Final: [[180, 340]]; recorded hit.

(b) Spoon left of cheese



Expectation: "Expected to fall in the region left of the cheese."
Diagnosis: state = correct; marker described as left of the cheese.
Final: [[228, 385]]; recorded miss.

Figure 4: **Example interactions.** B0 rows 5 and 152 (zero-based). Expectations are translated from Chinese; diagnoses are summarized. Images are reconstructed from source scenes and logged tool coordinates. Image labels use pixel coordinates; textual outputs use normalized coordinates.

E.2 Output Cardinality

Table 6 summarizes point-list cardinality across the proposal, first rendering candidate, and final answer.

Table 6: **Output point-count distributions for B0.** Columns give extracted list lengths; zero denotes an unavailable proposal.

Prediction	0	1	2	4	8	10
Stage 1 proposal	15	250	1	2	23	9
First rendering candidate	0	278	1	2	15	4
Final answer	0	278	1	2	15	4

E.3 Derived Output-State Rates

Table 7: **Rates derived from the initial-to-final output states.** Transition counts for B0 and A1–A4 are given in Table 2; A5 uses the same direct output at both endpoints. Rates use 300 records per condition and are percentages. "No answer" is the separately stored missing-terminal diagnostic.

Condition	Recovery	Regression	No answer
B0	10.64	0.49	0.00
A1	11.22	0.99	0.67
A2	5.62	37.91	9.67
A3	7.37	5.85	4.00
A4	5.32	4.37	1.00
A5	0.00	0.00	4.67

E.4 Coordinate-Update Trajectories

Table 8: **Coordinate revisions in B0.** These three trajectories each make two tool calls, and their final answers match the second call. Coordinates are normalized; hit states compare the Stage 1 proposal with the final output.

Row	Point	First call	Second call/final	States
179	1	(594,720)	(650,725)	M↛M
255	1	(284,382)	(300,382)	H↛H
	2	(350,406)	(360,406)	
	3	(381,414)	(390,414)	
	4	(381,414)	(390,414)	
	5	(381,414)	(390,414)	
	6	(381,414)	(390,414)	
	7	(381,414)	(390,414)	
	8	(381,414)	(390,414)	
	9	(381,414)	(390,414)	
	10	(381,414)	(390,414)	
264	1	(117,308)	(117,308)	M↛M
	2	(205,308)	(195,308)	
	3	(227,308)	(215,308)	
	4	(152,308)	(152,308)	
	5	(183,308)	(175,308)	
	6	(138,308)	(138,308)	
	7	(102,308)	(110,308)	
	8	(138,308)	(138,308)	