INNEREYE: GIVING MULTIMODAL AGENTS A PRE-ACTION VISUAL STATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal agents increasingly operate in interactive environments by interpreting visual inputs, maintaining interaction context, and generating executable actions. In many such agents, the explicit decision trajectory is progressively represented through autoregressive token sequences, while visual features remain available within the underlying multimodal model. However, agent decisions are inherently step-dependent: as interaction history evolves, the visual evidence relevant to the next action may change even when the visual input itself remains unchanged. This raises an agent-level question beyond visual perception itself: how should visual information be represented for the current decision before it is expressed as an action? We introduce **InnerEye**, a multimodal agent formulation that explicitly constructs a step-specific pre-action visual state before action generation. At each step, this transient state is freshly formed from the task instruction, visual input, and current interaction history, providing a latent representation that supports the upcoming action. InnerEye progressively trains this state from visual grounding toward supporting the upcoming action. Stage 1 performs visually grounded supervised fine-tuning by aligning latent hidden states with features from a frozen vision encoder while learning expert actions. Stage 2 further optimizes the grounded state through reinforcement learning with Online Latent Action Distillation, which transfers the pre-action visual states from a full-vision branch to an auxiliary branch without direct access to image tokens on successful rollouts, providing an action-oriented training signal without altering the standard full-vision inference process. Across two Qwen3-VL scales and three benchmarks spanning multi-step visual tool use and single-step visual action prediction, InnerEye consistently improves over both standard post-training baselines and a latent visual reasoning baseline.

1 INTRODUCTION

Multimodal agents are increasingly expected to operate in interactive environments by interpreting visual inputs, maintaining interaction context, and producing executable actions (Zhou et al., 2024; Koh et al., 2024; Zheng et al., 2024). Recent advances in multimodal foundation models and reinforcement learning have improved agent capabilities across web interaction, multimodal information seeking, and computer-use scenarios (He et al., 2025; Wei et al., 2025; Wu et al., 2026). In many such agents, interaction trajectories are represented primarily through autoregressive token sequences: previous actions, tool responses, intermediate reasoning, and subsequent actions are serialized into the model context and progressively extended during interaction (Zhang & Zhang, 2024; Zhang et al., 2024). While visual features remain available to the underlying multimodal model, the agent's explicit decision trajectory is therefore largely expressed in a symbolic token space. Yet decision-making is inherently step-dependent: as the interaction history evolves, the visual evidence relevant to the next action may change even when the visual input remains unchanged. This raises a complementary question beyond visual perception itself: *how should visual information be represented for the current decision before it is expressed as an action?*

Recent advances in latent reasoning suggest that intermediate computation need not always be externalized as natural language, but can instead be carried by continuous or latent representations (Hao et al., 2024; Yang et al., 2026b). In multimodal reasoning, such representations have further been explored as a way to preserve or manipulate visual information during reasoning. This perspective

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.

motivates us to revisit the agent-level question above. Rather than generating each action directly from the multimodal context in the symbolic action space, we ask whether an agent can first form a dedicated, non-verbal visual state conditioned on the current interaction context, and then use this state to support the upcoming action. Such a state would introduce an explicit intermediate representation between multimodal context encoding and action generation, without requiring the intermediate computation to be decoded into human-readable reasoning.

Based on this perspective, we introduce **InnerEye**, a multimodal agent framework that explicitly inserts a fixed-capacity pre-action visual state before discrete action generation. At each decision step, InnerEye introduces a fixed block of latent tokens whose contextual hidden states are conditioned on the task instruction, visual input, and current interaction history. Rather than being decoded into textual explanations, these hidden states form a non-verbal latent representation that supports the upcoming action. Importantly, the pre-action visual state is transient and step-specific: it is used only for the current decision and is not propagated as persistent latent memory. At the next decision step, a fresh state is formed from the updated interaction context, allowing the same visual input to induce different latent representations as the decision context evolves. In this way, InnerEye introduces an explicit latent computation stage between multimodal context encoding and action generation.

Learning such a pre-action visual state requires progressively shaping its representation from visual grounding toward action support. We first establish visual grounding through supervised fine-tuning, where the agent learns expert actions while the contextual hidden states at the latent positions are aligned with visual representations produced by a frozen vision encoder. The interior latent positions receive no token-level language-modeling targets, allowing their hidden states to remain flexible and context-dependent, while the alignment objective anchors them to the visual input. This provides a visually grounded starting point for further optimization toward the upcoming action.

To further shape the grounded state into a representation that supports the upcoming action, we introduce **Online Latent Action Distillation (OLAD)**, an auxiliary online self-distillation objective that complements task-level policy optimization. During reinforcement learning, a full-vision branch produces the policy's action distribution together with the layer-wise hidden states of the pre-action visual state. For successful rollouts, these latent states are transferred to the corresponding latent positions of an auxiliary branch with direct access to image tokens removed. The auxiliary branch is then trained to match the full-vision action distribution. In this way, OLAD directly couples the learned latent state to action prediction while leaving the standard full-vision inference process unchanged. The auxiliary branch is used only during training and does not impose an explicit visual reconstruction objective. Together, visual grounding anchors the latent state to the visual input, while OLAD further shapes it into a pre-action representation that supports action generation.

Our contributions are threefold:

- We introduce InnerEye, a multimodal agent formulation that explicitly constructs a fixed-capacity, non-verbal, and step-specific pre-action visual state before discrete action generation. The state is transient across decision steps and is freshly formed from the current multimodal context to support the upcoming action.
- We develop a progressive two-stage learning strategy that first grounds the pre-action visual state in visual representations and then further shapes it into a representation that supports action generation. In particular, we introduce OLAD, which uses an auxiliary branch without direct access to image tokens to provide an action-oriented training signal for the latent state, without imposing an explicit visual reconstruction objective.
- We evaluate InnerEye across multi-step visual tool-use and single-step visual action-prediction settings, covering two model scales and out-of-distribution evaluation. InnerEye consistently improves performance across these settings, and extensive ablation studies further support the effectiveness of its key design choices.

2 Related Work

2.1 Multimodal Agents with Reinforcement Learning

Recent work has increasingly used reinforcement learning to train multimodal agents for sequential decision making under environment feedback. Representative directions include visual web inter-

action and online multi-turn control (Yang et al., 2026a; Wei et al., 2025), long-horizon multimodal search (Zhang et al., 2026; Yan et al., 2026), active visual reasoning with external visual operations (Zheng et al., 2025; Huang et al., 2025), and general multimodal agent learning with heterogeneous perception and interaction capabilities (Liu et al., 2025). Across these settings, reinforcement learning is primarily used to improve action selection, interaction strategy, tool invocation, and long-horizon task success.

InnerEye studies a complementary question: how an agent can form an intermediate visual state that supports the upcoming action. Rather than only optimizing the resulting action trajectory, InnerEye introduces a transient, step-specific pre-action visual state and progressively trains it from visual grounding toward an action-oriented representation. This makes the pre-action representation itself an explicit learning target while retaining the standard full-vision inference process.

2.2 Latent Reasoning in Multimodal Models

Latent reasoning explores intermediate computation beyond explicit natural-language chains. In multimodal models, recent methods introduce latent visual representations to preserve, reconstruct, or transform perceptual information during reasoning. Mirage (Yang et al., 2026b) augments decoding with latent visual tokens supervised through visual-embedding distillation, LVR (Li et al., 2026a) performs autoregressive reasoning in visual embedding space by reconstructing query-relevant visual tokens, VaLR (Jeon et al., 2026) inserts vision-aligned latent tokens before textual reasoning steps to preserve perceptual information, and ILVR (Dong et al., 2026) interleaves latent visual representations whose evolving perceptual cues are selectively distilled from intermediate images. LIVR (Li et al., 2026b) learns task-adaptive visual reasoning tokens without explicit intermediate-image supervision.

InnerEye differs primarily in the role and lifecycle of the latent representation. Prior multimodal latent-reasoning methods use latent states for purposes such as reconstructing visual targets, preserving perceptual information during reasoning, or representing evolving intermediate visual states. In contrast, InnerEye forms a fresh pre-action state for each agent decision. The state is conditioned on the current task and interaction context, grounded in the available visual features, and discarded after the corresponding action rather than propagated across decision steps. OLAD further trains this grounded state to support action prediction through an auxiliary branch without direct access to image tokens, while the standard full-vision inference process remains unchanged.

3 Methodology

3.1 Preliminaries

Problem formulation. We consider a multimodal agent that operates over one or multiple decision steps. Each task instance provides a task instruction q and a visual input v. At decision step t, the agent receives a multimodal context $c_t = \big(q,\ v,\ \mathcal{H}_t\big)$, where $\mathcal{H}_t$ denotes the ordered interaction history before the current decision:

$$\mathcal{H}_t = \big((a_1, o_1), \ldots, (a_{t-1}, o_{t-1})\big). \tag{1}$$

Here, a_i denotes the agent action at step i, and o_i is the corresponding tool or environment response. In the multi-step settings considered in this work, the visual input v remains fixed throughout the interaction, while $\mathcal{H}_t$ evolves as new actions are executed and their responses are observed. The agent then produces an action a_t, which may correspond to a tool invocation or, at the terminal step, the final output. Its distribution factorizes autoregressively over action tokens:

$$\pi_\theta(a_t \mid c_t) = \prod_{i=1}^{|a_t|} p_\theta\big(y_{t,i} \mid c_t,\ y_{t,<i}\big), \tag{2}$$

where $y_{t,1}, \ldots, y_{t,|a_t|}$ denote the tokens comprising a_t. For single-step tasks, this formulation reduces to $T = 1$ and $\mathcal{H}_1 = \varnothing$.

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) avoids learning a separate value function by estimating relative advantages within a group of sampled trajectories. For

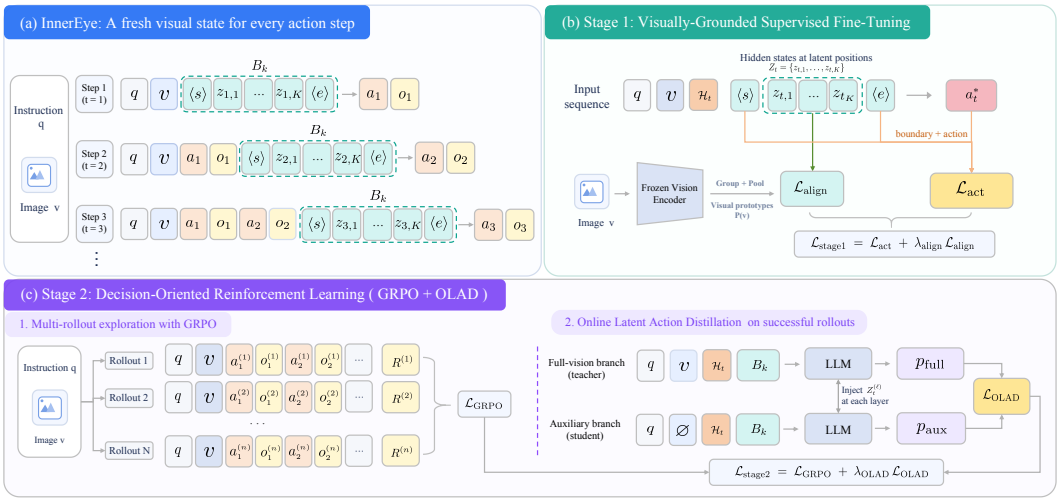


Figure 1: Overview of **InnerEye**. (a) A fresh pre-action visual state is formed at every decision step from the task instruction, fixed visual input, and interaction history; previous latent states are discarded rather than propagated across steps. (b) Stage 1 establishes visual grounding by aligning latent hidden states with pooled visual prototypes while supervising the boundary and action tokens with cross-entropy. (c) Stage 2 combines GRPO with OLAD: GRPO optimizes task-level behavior over sampled rollouts, while OLAD uses an auxiliary branch without direct access to image tokens to provide an action-oriented training signal to the pre-action visual state on successful rollouts.

a given task instance, we sample N complete trajectories $\{\tau^{(1)}, \ldots, \tau^{(N)}\}$ from the rollout policy $\pi_{\theta_{\text{old}}}$. Each trajectory receives a scalar reward $R^{(n)}$, and its group-relative advantage is computed as $\hat{A}^{(n)} = \frac{R^{(n)} - \mu_R}{\sigma_R + \epsilon_A}$, where μ_R and σ_R are the group mean and standard deviation. Let $r_{t,i}^{(n)} = \frac{\pi_\theta\left(y_{t,i}^{(n)} | c_t^{(n)}, y_{t,<i}^{(n)}\right)}{\pi_{\theta_{\text{old}}}\left(y_{t,i}^{(n)} | c_t^{(n)}, y_{t,<i}^{(n)}\right)}$. The policy is optimized by minimizing

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_{n,t,i}\left[\min\left(r_{t,i}^{(n)}\hat{A}^{(n)}, \text{clip}\left(r_{t,i}^{(n)}, 1-\epsilon, 1+\epsilon\right)\hat{A}^{(n)}\right)\right] + \beta D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}], \quad (3)$$

where the expectation is taken over action tokens across rollout steps and trajectories, ϵ is the clipping coefficient, and β controls KL regularization toward the frozen reference policy π_{ref}.

3.2 INNEREYE

The key idea of InnerEye is to introduce a fixed-capacity pre-action visual state between the multimodal context and action generation at every decision step, providing a non-verbal intermediate representation that supports the upcoming action. Concretely, we introduce a fixed latent token block consisting of K interior latent tokens enclosed by two boundary tokens, where we set $K = 4$:

$$B_K = \langle\text{latent_start}\rangle \underbrace{\langle\text{latent}\rangle \cdots \langle\text{latent}\rangle}_{K} \langle\text{latent_end}\rangle. \quad (4)$$

At step t, B_K is inserted between the multimodal context c_t and the action sequence, so that every action token can attend to the latent positions through standard causal attention. Although the token identities in B_K are fixed, their contextual hidden states depend on the current multimodal context c_t. We denote the hidden state at the k-th interior latent position by $z_{t,k}$ and define $Z_t = \{z_{t,1}, \ldots, z_{t,K}\}$ as the pre-action visual state at step t.

For multi-step interactions, each decision step forms its own pre-action visual state. Previous latent states are not retained in the interaction history: $\mathcal{H}_{t+1}$ contains only previous actions and their corresponding tool or environment responses, and a fresh copy of B_K is inserted at the next step to form a new context-dependent state. The vision encoder remains frozen throughout training, so both training stages focus on shaping how the language model forms and uses the pre-action visual state from the available visual features and interaction context.

3.2.1 STAGE 1: VISUALLY-GROUNDED SUPERVISED FINE-TUNING

The first stage aims to establish the pre-action visual state as a visually grounded internal representation while preserving the agent's ability to imitate expert actions. Given an expert trajectory $\big((c_1, a_1^*), \ldots, (c_T, a_T^*)\big)$, we decompose it into step-level training samples. At step t, the model processes the current multimodal context c_t, the latent block B_K, and the expert action a_t^*. The interaction history in $\mathcal{H}_t$ is retained as context, while latent states from previous steps are not carried forward. Consequently, a fresh pre-action visual state is formed at every decision step from the updated context rather than being propagated as persistent latent memory.

Stage 1 is optimized with two complementary objectives. The first is an action supervision loss based on standard autoregressive cross-entropy. We apply supervision to the two boundary tokens, ⟨latent_start⟩ and ⟨latent_end⟩, as well as to all tokens in the expert action a_t^*, while masking the K interior latent positions:

$$\mathcal{L}_{\text{act}} \;=\; -\sum_{i \in \mathcal{S}_t} \log\, p_\theta\big(y_i \mid c_t,\; y_{<i}\big), \tag{5}$$

where y_i indexes tokens in the sequence following c_t, and $\mathcal{S}_t$ denotes the supervised positions corresponding to the two boundary tokens and the expert action tokens. The boundary tokens provide structural supervision for entering and exiting the latent computation region, with ⟨latent_end⟩ marking the transition to action generation. In contrast, the K interior latent positions receive no token-level language-modeling targets. Although they share the same latent token identity, their contextual hidden states are not tied to predefined discrete semantic targets and can therefore vary with the current multimodal context.

The second objective explicitly grounds these latent representations in the visual input. Let $V(v) \in \mathbb{R}^{M \times d}$ denote the M visual tokens produced by the frozen vision encoder and its merger module. We partition these tokens into K contiguous groups and average-pool each group to obtain a set of visual prototypes $P(v) = \{p_1, \ldots, p_K\}$, where $p_k \in \mathbb{R}^d$. Denoting the decoder hidden state at the k-th interior latent position at step t as $z_{t,k}$, we minimize the cosine distance between each latent state and its corresponding visual prototype:

$$\mathcal{L}_{\text{align}} \;=\; \frac{1}{K} \sum_{k=1}^{K} \big(1 - \cos(z_{t,k},\; p_k)\big). \tag{6}$$

This alignment objective anchors the latent hidden states to the visual representation without imposing token-level semantic targets on the interior latent positions, allowing them to remain flexible while being explicitly grounded in the visual input.

The overall Stage 1 objective is

$$\mathcal{L}_{\text{stage1}} \;=\; \mathcal{L}_{\text{act}} \;+\; \lambda_{\text{align}}\, \mathcal{L}_{\text{align}}, \tag{7}$$

where λ_{align} controls the strength of visual alignment and is set to 0.5 in our experiments. Together, these objectives establish the pre-action visual state as a visually grounded representation that remains free from direct token-level language supervision at the interior latent positions.

3.2.2 STAGE 2: DECISION-ORIENTED REINFORCEMENT LEARNING

Stage 1 establishes the pre-action visual state as a visually grounded internal representation. Visual grounding alone, however, does not ensure that the pre-action visual state is shaped to support the upcoming action. In Stage 2, we further optimize the grounded state so that it can better support action prediction under the current multimodal context.

For each task instance, the current policy samples N complete trajectories $\{\tau^{(1)}, \ldots, \tau^{(N)}\}$. Each trajectory consists of a sequence of agent actions. At every decision step, a fresh pre-action visual state is formed at the latent positions of B_K, while sampling is performed only over the action tokens. Each completed trajectory receives a reward

$$R^{(n)} = R^{(n)}_{\text{acc}} + 0.1\, R^{(n)}_{\text{format}},$$

where $R^{(n)}_{\text{acc}} \in \{0, 1\}$ evaluates the correctness of the final task outcome, and $R^{(n)}_{\text{format}} \in \{0, 1\}$ indicates whether the generated output satisfies the required parsing format. We then optimize the policy using the standard GRPO objective defined in equation 3.

Table 1: Main results on ToolVQA and GTA benchmarks. Accuracy is the primary evaluation metric; tool-call statistics are included for completeness following the benchmark protocol. Best accuracy within each base-model group is shown in bold.

	ToolVQA			GTA		
Model	Accuracy	Tool Call	Tool Error	Accuracy	Tool Call	Tool Error
GPT-4o	34.96	2003	1804	41.52	-	-
GPT-3.5-Turbo	18.37	3178	1321	23.62	-	-
Claude-3.5-sonnet	30.52	585	524	-	-	-
Base model: Qwen3-VL-2B-Instruct						
Vanilla	11.46	0	0	29.07	0	0
Vanilla w/ tool	8.44	8306	2868	17.44	964	198
LIVR	58.04	3902	115	34.30	440	145
SFT	55.32	4131	24	39.53	503	47
GRPO	57.44	4010	27	42.63	474	42
InnerEye	**59.70**	4570	23	**44.38**	451	39
Base model: Qwen3-VL-8B-Instruct						
Vanilla	16.20	0	0	45.35	0	0
Vanilla w/ tool	14.23	5219	763	37.79	391	43
LIVR	62.53	3936	39	36.63	291	125
SFT	60.56	4328	23	47.67	423	35
GRPO	62.45	4349	20	48.26	494	36
InnerEye	**64.32**	4397	23	**52.91**	487	31

While GRPO optimizes task-level behavior through online exploration, it does not explicitly provide an action-oriented learning signal to the pre-action visual state. To complement this objective, we introduce OLAD, an auxiliary online self-distillation objective that further shapes the grounded pre-action visual state toward supporting the upcoming action. We apply OLAD only to rollouts that are both format-valid and task-correct, so that successful trajectories provide the distillation signal.

In the full-vision pass, we perform teacher forcing with context c_t and latent block B_K. From every decoder layer $\ell \in \{1, \dots, L\}$, we extract the hidden states at the K interior latent positions:

$$Z_t^{(\ell)} = \{z_{t,1}^{(\ell)}, \dots, z_{t,K}^{(\ell)}\} \in \mathbb{R}^{K \times d}, \tag{8}$$

together with the corresponding action-token distribution p_{full}. This distribution represents the current policy's action prediction under full visual access and serves as the online distillation target.

In the auxiliary pass, we construct a vision-removed context $\tilde{c}_t = (q, \varnothing, \mathcal{H}_t)$ by removing all image patch tokens while retaining the task instruction and interaction history. At each decoder layer ℓ, the full-vision latent hidden states are transferred to the corresponding interior latent positions:

$$\tilde{z}_{t,k}^{(\ell)} \leftarrow z_{t,k}^{(\ell)}. \tag{9}$$

The auxiliary branch therefore has no direct access to image tokens, and the transferred latent states provide its only direct pathway for visual information from the current input. The branch then produces an action-token distribution p_{aux}. This branch is used only during training and does not alter the standard full-vision inference process of InnerEye.

At each decision step of a selected rollout, OLAD minimizes the divergence between the full-vision and auxiliary action distributions:

$$\mathcal{L}_{\text{OLAD}} = \frac{1}{|\mathcal{A}_t|} \sum_{i \in \mathcal{A}_t} D_{\text{KL}}\Big(p_{\text{full}}\big(\cdot \mid c_t,\ B_K,\ y_{t,<i}\big) \ \Big\|\ p_{\text{aux}}\big(\cdot \mid \tilde{c}_t,\ Z_t^{(1:L)},\ y_{t,<i}\big)\Big), \tag{10}$$

where $\mathcal{A}_t$ denotes the set of action-token positions at step t. By matching the action distribution produced under full visual access, OLAD provides an action-oriented training signal that encourages the grounded pre-action visual state to better support prediction of the upcoming action.

The overall Stage 2 objective combines task-level policy optimization with latent action distillation:

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{GRPO}} + \lambda_{\text{OLAD}}\, \mathcal{L}_{\text{OLAD}}, \tag{11}$$

Table 2: Main results on AndroidControl-CL benchmark. We report accuracy on seven sub-tasks and their average. Commu, Educa, Enter, Product, Shop, Tool, and Travel denote Communication, Education, Entertainment, Productivity, Shopping, Tools, and Travel & Local, respectively.

Method	Commu	Educa	Enter	Product	Shop	Tool	Travel	Average
Base model: Qwen3-VL-2B-Instruct								
Vanilla	21.88	19.22	21.86	21.53	21.13	22.41	19.34	21.05
Few-shot	33.70	28.40	30.94	31.30	32.37	34.04	27.50	31.18
LIVR	71.04	63.06	62.32	67.09	68.30	72.06	61.52	66.48
SFT	69.72	59.18	65.24	68.42	65.64	72.26	64.13	66.37
GRPO	71.23	61.52	66.88	68.66	67.49	74.07	67.92	68.25
InnerEye	**72.74**	**63.21**	**68.04**	**72.99**	**69.84**	**75.65**	**68.77**	**70.18**
Base model: Qwen3-VL-8B-Instruct								
Vanilla	50.83	44.06	47.83	50.00	41.22	52.47	48.63	47.86
Few-shot	56.24	50.00	55.31	56.71	50.93	64.40	55.90	55.64
LIVR	71.60	64.58	67.06	72.25	70.39	76.33	69.96	70.31
SFT	70.83	62.63	68.26	71.14	68.73	75.46	68.59	69.38
GRPO	72.01	64.94	68.34	72.07	70.49	76.14	69.64	70.52
InnerEye	**73.85**	**65.73**	**71.12**	**73.05**	**72.30**	**78.98**	**72.91**	**72.56**

where λ_{OLAD} controls the strength of the distillation objective, and we set $\lambda_{\text{OLAD}} = 0.03$. Together, the two objectives combine task-level policy optimization with action-oriented shaping of the pre-action visual state.

4 Experiments

4.1 Experimental Setup

We instantiate InnerEye on Qwen3-VL (Bai et al., 2025) at two scales (2B and 8B) and evaluate it across two complementary settings. For multi-step evaluation, the agent is trained on the ToolVQA (Yin et al., 2025) training split and tested on both the ToolVQA test split and GTA (Wang et al., 2024), with GTA serving as an out-of-distribution benchmark. We compare against closed-source models and baselines built on the same Qwen3-VL backbone, including Vanilla, SFT, GRPO, and LIVR (Li et al., 2026b), a latent visual reasoning method without explicit intermediate-image supervision. GRPO is initialized from the corresponding SFT checkpoint and further optimized with reinforcement learning under the same task setting. We report accuracy as the primary metric. For single-step evaluation, the agent is trained and tested on AndroidControl-CL (Yao et al., 2026), a mobile device control benchmark comprising seven sub-tasks. Baselines include Vanilla, few-shot prompting, SFT, GRPO, and LIVR. For GRPO and InnerEye, the main results are averaged over three random seeds and all ablation studies are conducted with seed 42.

Throughout both stages, the vision encoder and its merger remain frozen; only the LLM decoder is updated. Stage 1 uses a learning rate of 1×10^{-5} with a global batch size of 64, while Stage 2 lowers the learning rate to 1×10^{-6}, increases the global batch size to 128, and sets the rollout group size to N=8. All training is conducted on 4×NVIDIA RTX PRO 6000 GPUs using DeepSpeed (Rasley et al., 2020) ZeRO-2 with gradient accumulation. Further details are deferred to Appendix A.1.

4.2 Main Results

Multi-step setting. Table 1 summarizes the results on the multi-step benchmarks. InnerEye consistently achieves the highest accuracy among methods built on the same Qwen3-VL backbone across both model scales and benchmarks. With Qwen3-VL-2B-Instruct, InnerEye reaches 59.70% on ToolVQA and 44.38% on GTA, outperforming GRPO by 2.26 and 1.75 percentage points, respectively. With Qwen3-VL-8B-Instruct, InnerEye achieves 64.32% on ToolVQA and 52.91% on GTA, yielding gains of 1.87 and 4.65 points over GRPO. InnerEye also consistently outperforms LIVR across both model scales and benchmarks. On ToolVQA, InnerEye improves over LIVR by 1.66

Table 3: Ablation study of latent capacity, training design, and latent-state interventions in InnerEye. All ablation experiments use Qwen3-VL-2B-Instruct with seed 42.

Ablation	ToolVQA	GTA	AndroidControl-CL	Average
Latent capacity				
$K = 1$	58.84	41.86	69.01	56.57
$K = 2$	**59.89**	43.02	70.07	57.66
$K = 4$ (Ours)	59.83	**44.19**	**70.13**	**58.05**
$K = 8$	59.27	41.86	69.83	56.99
$K = 16$	59.15	40.12	69.28	56.18
Training design				
w/o Stage 2	56.50	41.28	67.63	55.14
w/o accuracy gate	59.40	43.02	69.91	57.44
w/o OLAD	58.96	40.70	69.44	56.37
w/o visual alignment	58.41	40.70	69.84	56.32
Latent block only	58.16	40.12	69.43	55.90
Latent-state intervention				
Masked latent	56.31	38.95	57.72	50.99
Noise latent	55.95	37.79	53.49	49.08
Previous-step latent	59.14	41.86	–	–
Step-shuffled latent	59.02	41.27	–	–
Cross-sample latent	57.85	40.12	68.85	55.61

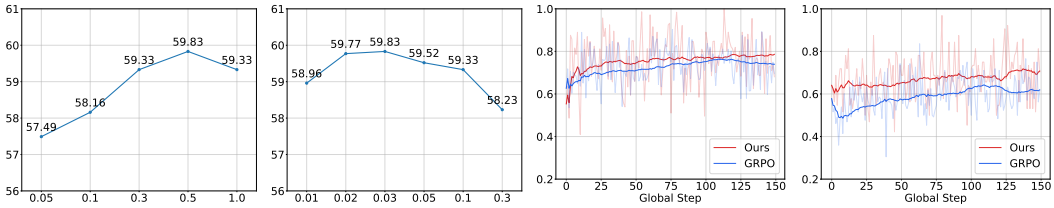
and 1.79 points with the 2B and 8B models, respectively. The gap is larger on the out-of-distribution GTA benchmark, where InnerEye exceeds LIVR by 10.08 and 16.28 points. These results suggest that explicitly forming and training a step-specific pre-action visual state provides benefits beyond latent visual reasoning alone, particularly in the multi-step out-of-distribution setting.

Single-step setting. Table 2 reports the results on AndroidControl-CL. InnerEye again achieves the highest average accuracy across both model scales. With Qwen3-VL-2B-Instruct, InnerEye reaches 70.18%, compared with 66.37% for SFT, 68.25% for GRPO, and 66.48% for LIVR. With Qwen3-VL-8B-Instruct, InnerEye achieves 72.56%, improving over SFT, GRPO, and LIVR by 3.18, 2.04, and 2.25 percentage points, respectively. The improvements are also consistent across subtasks. Together with the multi-step results, these findings show that the benefit of the pre-action visual state is not limited to interaction histories spanning multiple decisions; it also extends to direct visual action prediction in the single-step setting.

4.3 Ablation Study

Latent capacity. All ablation studies are conducted with Qwen3-VL-2B-Instruct using seed 42. We first examine the effect of the number of interior latent tokens K. As shown in Table 3, increasing K from 1 to 4 generally improves performance, while further increasing the latent capacity provides no additional benefit and leads to degradation. Among the evaluated settings, $K = 2$ and $K = 4$ perform similarly on ToolVQA, whereas $K = 4$ achieves the highest average accuracy. These results suggest that InnerEye does not require a large latent capacity to obtain its gains.

Training design. We next ablate the major components of the progressive training strategy. Removing Stage 2 reduces the average accuracy from 58.05% to 55.14%, showing that Stage 1 alone does not fully realize the benefit of the pre-action visual state. Removing OLAD also degrades performance, supporting its role in further shaping the grounded state to support the upcoming action. Removing visual alignment in Stage 1 lowers the average accuracy to 56.32%. Importantly, the latent block only variant removes both visual alignment and OLAD while preserving the same latent-token architecture and capacity as the full model. Its average accuracy drops to 55.90%, indicating that the gains cannot be explained solely by inserting additional latent computation positions before action generation. Finally, removing the accuracy gate, i.e., applying OLAD to all format-valid rollouts regardless of task correctness, also degrades performance, suggesting that restricting OLAD to successful rollouts provides a more reliable distillation signal.



(a) λ_{align} (b) λ_{OLAD} (c) AndroidControl-CL (d) ToolVQA

Figure 2: Additional ablation analyses. (a-b) Sensitivity to the Stage 1 alignment weight λ_{align} and the Stage 2 OLAD weight λ_{OLAD} on ToolVQA. (c-d) Accuracy reward during Stage 2 training for InnerEye and standard GRPO. All results use Qwen3-VL-2B-Instruct with seed 42.

Latent-state interventions. Beyond ablating the training objectives, we directly intervene on the learned pre-action visual state at inference time to examine whether its content contributes to action generation. Masking the interior latent states or replacing them with norm-matched Gaussian noise consistently degrades performance across all three benchmarks. On AndroidControl-CL, accuracy drops from 70.13% to 57.72% under masking and to 53.49% under noise replacement, even though the original visual input remains available. These results indicate that the trained policy actively uses the information encoded in the latent states rather than treating them as unused intermediate positions. We further examine whether the correspondence between the pre-action visual state and the current decision context matters. On the multi-step benchmarks, replacing the current state with the previous-step state or shuffling states across steps within the same trajectory consistently reduces performance. Replacing it with a state from a different sample also degrades performance, reducing the average accuracy to 55.61%. Together, these interventions show that preserving both step and sample correspondence is important for action generation, providing evidence that the useful content of the pre-action visual state depends on the current interaction context.

Hyperparameters. We further analyze the sensitivity of InnerEye to the auxiliary-loss weights used. As shown in Fig. 2(a-b), performance first improves and then declines as either λ_{align} or λ_{OLAD} increases. This suggests that both excessively weak and excessively strong auxiliary supervision can be suboptimal. Among the evaluated settings, $\lambda_{\text{align}} = 0.5$ and $\lambda_{\text{OLAD}} = 0.03$ achieve the highest accuracy, respectively, indicating that visual grounding and OLAD are most effective when appropriately balanced with the corresponding action-learning objectives.

Training dynamics. Figures 2(c-d) compare the Stage 2 optimization trajectories of InnerEye and standard GRPO on AndroidControl-CL and ToolVQA. We report both the raw task-accuracy reward observed during training and its smoothed trend for clearer visualization. InnerEye maintains a higher accuracy reward over most of the optimization process and reaches a higher final reward on both benchmarks, with a larger gap on ToolVQA.

5 Conclusion

We presented InnerEye, a multimodal agent formulation that introduces a non-verbal, step-specific pre-action visual state before discrete action generation. At each decision step, InnerEye forms a transient pre-action visual state from the current multimodal context, providing an intermediate representation that supports the upcoming action. We further developed a two-stage training strategy that first grounds this state in visual representations through supervised fine-tuning and then shapes it through reinforcement learning with GRPO and OLAD into a more action-oriented representation. Experiments across multi-step visual tool-use and single-step visual action-prediction benchmarks show consistent improvements across two model scales, while ablation studies support the effectiveness of the key design choices. These results provide evidence that explicitly learning a transient pre-action visual state can benefit multimodal agent decision making in the evaluated settings and motivate further study of richer latent representations for longer-horizon interactive environments, offering a simple direction for structuring latent computation before action generation.

AI USE STATEMENT

We used generative AI tools for checking syntactic and grammatical errors and improving the overall readability of the manuscript. We carefully reviewed and verified all AI-assisted work and take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide detailed information to facilitate reproduction of our results. The model configurations, benchmark settings, baselines, and evaluation protocols are described in Section 4.1. Detailed training configurations and hyperparameters for both training stages are provided in Appendix A.1, while the benchmark-specific evaluation protocols and system prompts are given in Appendices A.3 and A.4. The inference-time latent-state intervention procedures are documented in Appendix A.5. Finally, complete pseudocode for the two-stage training procedure is provided in Appendix B. An anonymous repository containing the core implementation of InnerEye is also provided here.

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*, 2025.

Shuai Dong, Siyuan Wang, Xingyu Liu, Chenglin Li, Haowen Hou, and Zhongyu Wei. Interleaved latent visual reasoning with selective perceptual modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29316–29335, 2026.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27545–27564, 2025.

Zeyi Huang, Yuyang Ji, Anirudh Sundara Rajan, Zefan Cai, Wen Xiao, Haohan Wang, Junjie Hu, and Yong Jae Lee. Visualtoolagent (vista): A reinforcement learning framework for visual tool selection. *arXiv preprint arXiv:2505.20289*, 2025.

Byungwoo Jeon, Yoonwoo Jeong, Hyunseok Lee, Minsu Cho, and Jinwoo Shin. Vision-aligned latent reasoning for multi-modal large language model. *arXiv preprint arXiv:2602.04476*, 2026.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 148076–148090, 2026a.

Kelvin Li, Chuyi Shang, Leonid Karlinsky, Rogerio Feris, Trevor Darrell, and Roei Herzig. Latent implicit visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33457–33466, 2026b.

Ziyu Liu, Yuhang Zang, Yushan Zou, Zijian Liang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual agentic reinforcement fine-tuning. *arXiv preprint arXiv:2505.14246*, 2025.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3505–3506, 2020.

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.

Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. Gta: a benchmark for general tool agents. *Advances in Neural Information Processing Systems*, 37: 75749–75790, 2024.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 7909–7928. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.401. URL https://doi.org/10.18653/v1/2025.emnlp-main.401.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 2456–2487. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.114. URL https://doi.org/10.18653/v1/2026.acl-long.114.

Wentao Yan, Shengqin Wang, Huichi Zhou, Yihang Chen, Kun Shao, Yuan Xie, and Zhizhong Zhang. Prommsearchagent: A generalizable multimodal search agent trained with process-oriented rewards. *arXiv preprint arXiv:2604.20486*, 2026.

Rui Yang, Qianhui Wu, Yuxi Chen, Hao Bai, Wenlin Yao, Hao Cheng, Baolin Peng, Huan Zhang, Tong Zhang, and Jianfeng Gao. Openwebrl: Demystifying online multi-turn reinforcement learning for visual web agents. *arXiv preprint arXiv:2606.02031*, 2026a.

Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33510–33520, 2026b.

Zhenquan Yao, Zitong Huang, Yihan Zeng, Jianhua Han, Hang Xu, Chun-Mei Feng, Jianwei Ma, and Wangmeng Zuo. Cgl: Advancing continual gui learning via reinforcement fine-tuning. *arXiv preprint arXiv:2603.02951*, 2026.

Shaofeng Yin, Ting Lei, and Yang Liu. Toolvqa: A dataset for multi-step reasoning vqa with external tools. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4424–4433. IEEE, 2025.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12016–12031, 2024.

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.

Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3132–3149, 2024.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. *CoRR*, abs/2505.14362, 2025. doi: 10.48550/ARXIV.2505.14362. URL https://doi.org/10.48550/arXiv.2505.14362.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

A Implementation and Evaluation Details

A.1 Training Configuration

We provide the detailed training configurations of InnerEye in Tables 4 and 5. Across both stages, the vision encoder and visual merger are kept frozen, and only the LLM decoder is optimized. All training runs are conducted on four NVIDIA RTX PRO 6000 GPUs with DeepSpeed ZeRO-2 and bfloat16 precision. Unless otherwise specified, we use $K = 4$, $\lambda_{\text{align}} = 0.5$, and $\lambda_{\text{OLAD}} = 0.03$ throughout training. During evaluation, we use greedy decoding for all methods.

Table 4: Training hyperparameters for Stage 1.

Configuration	Stage 1
Trainable parameters	LLM decoder
Frozen parameters	Vision encoder + visual merger
Global batch size	64
Learning rate	1×10^{-5}
DeepSpeed ZeRO stage	2
Learning-rate schedule	Warmup + cosine decay
Warmup ratio	0.03
Weight decay	0.1
Training epochs	2
Optimizer	AdamW
Precision	bfloat16

Table 5: Training hyperparameters for Stage 2.

Configuration	Stage 2
Trainable parameters	LLM decoder
Frozen parameters	Vision encoder + visual merger
Global batch size	128
Rollout group size N	8
Learning rate	1×10^{-6}
DeepSpeed ZeRO stage	2
Learning-rate schedule	Warmup + cosine decay
Warmup ratio	0.03
Weight decay	0.1
Training steps	150
Optimizer	AdamW
Precision	bfloat16
Rollout temperature	1.0
Rollout top-p	1.0
Rollout top-k	0
clipping coefficient ϵ	0.2
KL coefficient β	0.02

When computing the OLAD objective, we stop gradients through the full-vision target distribution p_{full}, while retaining gradients through the injected latent states $Z_t^{(1:L)}$. Therefore, p_{full} is treated as a fixed target in $D_{\text{KL}}(p_{\text{full}} \| p_{\text{aux}})$, whereas the latent states remain differentiable when they are injected into the auxiliary branch. As a result, $\mathcal{L}_{\text{OLAD}}$ updates the auxiliary prediction pathway as well as the upstream representations that produce $Z_t^{(1:L)}$, but does not backpropagate through the full-vision target distribution itself.

SFT is trained on the same expert data using the corresponding Qwen3-VL backbone. GRPO is initialized from the SFT checkpoint and uses the same reinforcement-learning configuration. For LIVR, we reimplemented the method following the original paper. Following the configuration reported in the paper, we use 16 latent visual reasoning tokens and adopt the same two-stage training procedure and hyperparameter settings.

A.2 Multi-Seed Robustness

Due to computational constraints, we conduct multi-seed evaluation for GRPO and InnerEye, which form the primary comparison for assessing the robustness of our reinforcement-learning improvements. We use three random seeds, 42, 43, and 44, and keep all other training configurations unchanged. Table 6 reports the mean and standard deviation of the primary evaluation metrics across runs. For AndroidControl-CL, the reported metric is the average accuracy over the seven sub-tasks.

Table 6: Multi-seed results over three random seeds. We report mean $\pm$ standard deviation of accuracy.

Method	ToolVQA	GTA	AndroidControl-CL
Base model: Qwen3-VL-2B-Instruct			
GRPO	57.44 ± 0.12	42.63 ± 0.33	68.25 ± 0.29
InnerEye	$\mathbf{59.70 \pm 0.13}$	$\mathbf{44.38 \pm 0.33}$	$\mathbf{70.18 \pm 0.07}$
Base model: Qwen3-VL-8B-Instruct			
GRPO	62.45 ± 0.22	48.26 ± 0.00	70.52 ± 0.17
InnerEye	$\mathbf{64.32 \pm 0.28}$	$\mathbf{52.91 \pm 0.58}$	$\mathbf{72.56 \pm 0.20}$

A.3 Benchmark and Evaluation Protocol

We evaluate InnerEye under both multi-step and single-step settings, following the task-specific protocols of the corresponding benchmarks.

Multi-step evaluation. ToolVQA and GTA are used for multi-step evaluation. Each task instance is initialized with a fixed visual input and a task instruction, and the visual input remains unchanged throughout the entire interaction trajectory. The agent may invoke tools over multiple decision steps, with each tool response appended to the interaction history and provided as context for subsequent decisions. We set the maximum number of tool calls to five. After collecting sufficient evidence, the agent terminates the trajectory by producing a final answer enclosed by <answer> and </answer>.

Training is performed exclusively on the ToolVQA training split. We evaluate on the ToolVQA test split for in-domain performance, while GTA is used only for evaluation and serves as an out-of-distribution benchmark. The primary metric is final task accuracy. A prediction is considered correct if the content of the final answer matches the ground-truth answer under string matching; intermediate tool calls do not directly affect the accuracy metric.

Single-step evaluation. AndroidControl-CL is used for single-step evaluation. Each sample requires the agent to predict one next action from the task instruction, screenshot, and the action context provided by the benchmark, after which the sample terminates. For click actions, a prediction is considered correct if the predicted click coordinate falls inside the corresponding ground-truth bounding box. For non-click operations, the predicted operation must exactly satisfy the required action specification. AndroidControl-CL contains seven task categories: Communication, Education, Entertainment, Productivity, Shopping, Tools, and Travel_and_local. We report the accuracy of each category together with their unweighted average.

A.4 Prompt and Action Format

We use task-specific system prompts for the multi-step and single-step settings to reflect their different interaction protocols. Within each setting, all methods involving training use the same system prompt. Prompting-only baselines use separate prompting configurations, which are also provided below.

Multi-step setting. For the multi-step ToolVQA and GTA benchmarks, the agent may either invoke a tool or terminate the trajectory with a final answer. Tool calls must follow the prescribed structured format so that they can be parsed and executed by the environment, while final answers are enclosed by <answer> and </answer>. The system prompt used by methods involving training is:

System prompt for trained methods (ToolVQA / GTA)

You are a multimodal tool−use agent. Use tools when necessary to solve the visual question. After collecting enough evidence, provide the final answer in <answer>...</answer>. If you need a tool, output exactly one tool call in the format <tool_call>{"name": "ToolName", "arguments": {...}}</ tool_call>. For image tools, use "image": "current_image".

During interaction, each valid tool call is executed by the environment and the corresponding tool response is appended to the interaction history for the next decision step. The trajectory terminates once the agent produces a final answer. The format reward used in Stage 2 evaluates whether the generated output can be successfully parsed according to this interaction protocol.

Multi-step prompting baselines. For the prompting-only baselines, we directly evaluate the corresponding Qwen3-VL-Instruct model without additional training. Vanilla uses the following system prompt:

Vanilla system prompt (ToolVQA / GTA)

You are a helpful multimodal assistant. Answer the visual question based on the image. Provide the final answer in <answer>...</answer>. Inside the tags, write only the shortest direct answer.

Vanilla w/ tool additionally exposes the available tool interface through the following system prompt:

Vanilla w/ tool system prompt (ToolVQA / GTA)

You are a multimodal question−answering agent. You may answer directly or call one of the available tools. Never invent a tool or argument name.
Use exactly one valid JSON tool call at a time:
<tool_call>{"name":"TOOL_NAME","arguments":{"ARGUMENT_NAME":VALUE}}</tool_call>
Argument names are case−sensitive. Every JSON string value must use double quotes. For an image argument, the value must be the JSON string "current_image".
After tool observations are sufficient, answer exactly with <answer>FINAL_ANSWER</answer>.
Available tools:
[The tool list returned by the tool server is dynamically inserted here. Each tool entry contains its name , description, and arguments.]

For Vanilla w/ tool, the available-tool section is dynamically populated with the tool names, descriptions, and argument schemas returned by the tool server. The inserted schema is authoritative for the available tool names and arguments.

Single-step setting. For AndroidControl-CL, each sample requires the agent to predict a single next GUI action from the task instruction, current screenshot, and previous-action history provided by the benchmark input. No additional environment interaction is performed after the predicted action. The system prompt used by methods involving training is:

System prompt for trained methods (AndroidControl-CL)

You are a useful GUI agent. You are given a task and a screenshot of the screen. You need to perform the next action command to complete the task according to the given UI screenshot, goal, and previous action list. You need to provide the next action command.

Single-step prompting baselines. For the Vanilla baseline, we directly evaluate the corresponding Qwen3-VL-Instruct model without additional training using the following system prompt:

Vanilla system prompt (AndroidControl-CL)

You are a GUI control agent. Given a screenshot, a goal, and an optional previous−action history, predict exactly one next action that best advances or completes the goal.
Legal actions and required formats:
– Click(x,y): tap a visible target at normalized screenshot coordinates x,y in [0,1].
– Long_press(x,y): long−press a visible target at normalized coordinates x,y in [0,1].
– Input_text(text): type the exact required text into the focused field.
– Open(app): open the named application.
– Scroll(direction): scroll in exactly one direction: up, down, left, or right.
– Back(): go to the previous screen.
– Home(): go to the device home screen.
– Wait(): wait for the current screen to finish loading or updating.
– Finish(): use only when the goal is already completed and no further action is needed.
Choose coordinates from the screenshot when using Click or Long_press. Use the goal, current screenshot, and previous actions together. Output only one action command in exactly one of the formats above.

The Few-shot baseline uses the same system prompt as Vanilla and additionally provides five demonstration examples. For all AndroidControl-CL methods, the generated action is evaluated directly against the benchmark target under its prescribed action representation.

A.5 Latent-State Intervention Protocol

We further conduct inference-time interventions on the learned pre-action visual state to examine how the latent content affects agent decisions. All interventions are applied only to the interior latent positions; the boundary tokens ⟨latent_start⟩ and ⟨latent_end⟩ remain unchanged.

Masked latent. For the masking intervention, the hidden states of all interior latent tokens are set to zero before they are used for action generation.

Noise latent. For the noise intervention, each interior latent state is replaced by Gaussian noise whose norm is matched to that of the original latent representation. This preserves the representation scale while disrupting its learned content.

Previous-step latent. For multi-step trajectories, the current pre-action latent state is replaced by the latent state from the immediately preceding decision step. The first decision step is left unchanged because no previous latent state is available.

Step-shuffled latent. For the random intervention, the latent states collected along the trajectory are randomly permuted across decision steps and reassigned to different steps. This preserves the set

of latent representations within the trajectory while disrupting their alignment with the corresponding decision steps.

These interventions leave the original visual input and all non-latent tokens unchanged, allowing us to isolate the effect of the learned pre-action latent representations on action generation.

Cross-sample latent. For the cross-sample intervention, the current pre-action latent state is replaced by a latent state obtained from a different task instance. The replacement latent state is sampled from another example and substituted at the corresponding latent positions, while the original visual input and all non-latent tokens of the current sample remain unchanged. This intervention disrupts the correspondence between the latent representation and the current sample while preserving the latent-state structure.

A.6 Detailed AndroidControl-CL ablations

Table 7: Detailed ablation results on AndroidControl-CL. We report accuracy for each task category and the average over the seven categories. All results are obtained with Qwen3-VL-2B-Instruct using seed 42.

Ablation	Commu.	Educa.	Enter.	Product.	Shop.	Tool	Travel	Average
Latent capacity								
$K = 1$	71.16	60.80	66.44	72.09	69.82	74.59	68.18	69.01
$K = 2$	73.26	61.66	68.84	71.48	69.25	77.11	68.86	70.07
$K = 4$ (Ours)	72.71	63.39	68.78	72.92	68.78	75.56	68.79	70.13
$K = 8$	71.93	61.66	68.44	72.03	69.21	75.56	69.96	69.83
$K = 16$	71.16	61.77	67.47	70.75	69.73	74.59	69.48	69.28
Training design								
Ours	72.71	63.39	68.78	72.92	68.78	75.56	68.79	70.13
w/o Stage 2	71.71	58.32	66.10	70.20	67.40	74.59	65.09	67.63
w/o accuracy gate	72.71	62.63	67.12	72.53	70.49	74.78	69.07	69.91
w/o OLAD	71.93	60.91	68.15	72.86	69.21	74.68	68.31	69.44
w/o visual alignment	72.38	63.71	67.98	71.14	69.30	75.46	68.93	69.84
Latent block only	72.15	63.39	68.09	71.14	68.02	74.98	68.24	69.43
Latent-state intervention								
Masked latent	66.07	46.54	56.79	65.87	49.45	65.86	53.49	57.72
Noise latent	62.10	43.30	53.14	61.60	43.65	61.69	48.97	53.49
Cross-sample latent	71.49	61.12	65.74	69.92	69.39	75.94	68.38	68.85

Table 7 provides the per-category AndroidControl-CL results corresponding to the aggregate ablations reported in Table 3. The training ablations generally reduce the overall average, with some variation across individual categories, while latent masking and noise replacement lead to substantially larger drops across all seven tasks. These detailed results confirm that the trends observed in the averaged score are not driven by a single category.

A.7 Training Cost

We report the wall-clock cost of Stage 2 training on Qwen3-VL-2B-Instruct using four NVIDIA RTX PRO 6000 GPUs. As shown in Table 8, standard GRPO requires 4.31 hours, while InnerEye requires 8.47 hours under the same hardware configuration, corresponding to approximately $1.97\times$ the training time. The additional cost arises from the auxiliary OLAD computation and is incurred only during training; the auxiliary vision-removed branch is not used at inference time.

B Training Algorithm Pseudocode

Algorithms 1 and 2 summarize the complete training procedures of InnerEye. Stage 1 performs visually grounded supervised fine-tuning, where the latent positions are explicitly aligned with pooled visual representations while expert actions are learned through token-level supervision. Stage 2

Table 8: Stage 2 wall-clock training cost on Qwen3-VL-2B-Instruct using four NVIDIA RTX PRO 6000 GPUs.

Method	Training time (h)	Relative cost
GRPO	4.31	1.00×
InnerEye	8.47	1.97×

Algorithm 1 Stage 1: Visually Grounded Supervised Fine-Tuning

Require: SFT dataset $\mathcal{D}_{\mathrm{SFT}}$, latent size K, alignment weight λ_{align}
Ensure: Updated parameters θ
  for each training batch from $\mathcal{D}_{\mathrm{SFT}}$ **do**
    $\mathcal{L}_{\mathrm{stage1}} \leftarrow 0$
    for each trajectory $\big((c_1, a_1^*), \ldots, (c_T, a_T^*)\big)$ in the batch **do**
      $V(v) \leftarrow \mathrm{VisionEncoder}(v)$
      $P(v) \leftarrow \mathrm{GroupPool}(V(v), K)$
      for $t = 1, \ldots, T$ **do**
        $c_t \leftarrow (q, v, \mathcal{H}_t)$
        Insert latent block B_K
        $Z_t \leftarrow \mathrm{Forward}_\theta(c_t, B_K, a_t^*)$
        Compute $\mathcal{L}_{\mathrm{act}}^{(t)}$ on boundary and action-token positions
        $\mathcal{L}_{\mathrm{align}}^{(t)} \leftarrow \dfrac{1}{K} \displaystyle\sum_{k=1}^{K} (1 - \cos(z_{t,k}, p_k))$
        $\mathcal{L}_{\mathrm{stage1}} \leftarrow \mathcal{L}_{\mathrm{stage1}} + \mathcal{L}_{\mathrm{act}}^{(t)} + \lambda_{\mathrm{align}} \mathcal{L}_{\mathrm{align}}^{(t)}$
      end for
    end for
    Normalize $\mathcal{L}_{\mathrm{stage1}}$ over decision steps
    Update θ using $\mathcal{L}_{\mathrm{stage1}}$
  end for

continues from the Stage 1 checkpoint and combines GRPO with OLAD. All sampled rollouts contribute to the GRPO objective, whereas OLAD is applied only to trajectories that are both task-correct and format-valid.

In Stage 1, the visual encoder is applied once to the fixed visual input of each trajectory, and the resulting visual tokens are grouped into K pooled prototypes for latent alignment. At every decision step, a fresh latent block B_K is inserted and the corresponding contextual hidden states Z_t are formed from the current multimodal context. Cross-entropy supervision is applied only to the boundary and expert action-token positions, while the interior latent positions receive no token-level language-modeling targets. The latent hidden states are instead explicitly grounded through the visual alignment objective.

C LIMITATIONS

Our study is currently limited to settings where the same visual input is maintained across action steps, while the interaction history evolves over time. Accordingly, the current evaluation does not cover interactive settings in which visual observations change after each action. Extending InnerEye to such settings with dynamically changing visual observations is an important direction for future work, where the pre-action visual state may need to be updated or coordinated across successive observations.

Algorithm 2 Stage 2: Decision-Oriented Reinforcement Learning

Require: Policy π_θ, reference policy π_{ref}, rollout group size N, OLAD weight λ_{OLAD}
Ensure: Updated parameters θ

for each task instance (q, v) **do**
  Sample $\{\tau^{(1)}, \ldots, \tau^{(N)}\} \sim \pi_{\theta_{\text{old}}}$
  for $n = 1, \ldots, N$ **do**
    Evaluate $\tau^{(n)}$
    $R^{(n)} \leftarrow R^{(n)}_{\text{acc}} + 0.1 R^{(n)}_{\text{format}}$
  end for
  Compute group-relative advantages $\{\hat{A}^{(n)}\}_{n=1}^{N}$
  Compute $\mathcal{L}_{\text{GRPO}}$ using all N rollouts
  $\mathcal{S} \leftarrow \left\{ n \mid R^{(n)}_{\text{acc}} = 1 \wedge R^{(n)}_{\text{format}} = 1 \right\}$
  $\mathcal{L}_{\text{OLAD}} \leftarrow 0$
  $M \leftarrow 0$
  for $n \in \mathcal{S}$ **do**
    for each decision step t in $\tau^{(n)}$ **do**
      $c_t^{(n)} \leftarrow (q, v, \mathcal{H}_t^{(n)})$
      $\left(\{Z_t^{(\ell,n)}\}_{\ell=1}^{L}, p_{\text{full}} \right) \leftarrow \text{FullVisionForward}\left(c_t^{(n)}, B_K, a_t^{(n)} \right)$
      $\tilde{c}_t^{(n)} \leftarrow (q, \varnothing, \mathcal{H}_t^{(n)})$
      $p_{\text{aux}} \leftarrow \text{AuxiliaryForward}\left(\tilde{c}_t^{(n)}, B_K, a_t^{(n)}, \{Z_t^{(\ell,n)}\}_{\ell=1}^{L} \right)$
      $\mathcal{L}_{\text{OLAD}} \leftarrow \mathcal{L}_{\text{OLAD}} + \dfrac{1}{|\mathcal{A}_t^{(n)}|} \displaystyle\sum_{i \in \mathcal{A}_t^{(n)}} D_{\text{KL}}\left(p_{\text{full},i} \,\|\, p_{\text{aux},i} \right)$
      $M \leftarrow M + 1$
    end for
  end for
  if $M > 0$ **then**
    $\mathcal{L}_{\text{OLAD}} \leftarrow \mathcal{L}_{\text{OLAD}} / M$
  end if
  $\mathcal{L}_{\text{stage2}} \leftarrow \mathcal{L}_{\text{GRPO}} + \lambda_{\text{OLAD}} \mathcal{L}_{\text{OLAD}}$
  Update θ using $\mathcal{L}_{\text{stage2}}$
end for