NAVIGATE: NEXT-ACTION VERIFIABLE AND FLEXIBLE INCENTIVES WITH GUIDED ADAPTIVE TOOL EXECUTION FOR MULTIMODAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal Large Language Models (MLLMs) remain brittle on tasks requiring multi-step perception–reasoning and tool use. Existing approaches degrade perceptual accuracy, overfit to tool-call patterns, or generalize poorly to multi-step rollouts—manifesting as irrelevant or skipped tool invocations. We present NAVIGATE, a multi-stage framework that teaches MLLMs to plan next actions, call perception specialists, and verify outcomes end-to-end. The first stage performs tool-aware adaptive sampling: a curriculum that weighs trajectories by tool usage, compositionality, and perceptual hardness, progressing from easy grounding to complex multi-tool reasoning. The second stage augments supervised fine-tuning (SFT) with a planner head that forecasts action sequences and regresses to action embeddings to respect similarity among tools. The final stage applies reinforcement learning (RL) with verifiable (terminal accuracy, format) and flexible (step-wise consistency, embedding-transport discrepancy, variance) rewards to optimize interactive rollouts. On multiple benchmarks, NAVIGATE yields around 10% gains over SFT and RL baselines across different backbones. We also conduct a series of ablation experiments and analyses to gain deeper insight into our framework.

1 INTRODUCTION

When humans face problems that exceed their immediate perceptual or computational bandwidth, they rarely rely on "thinking harder" alone (Clark & Chalmers, 1998)—they delegate: we zoom in to read tiny text, use a calculator for arithmetic, or consult a specialist when details matter. This perspective is increasingly relevant for multimodal large language models (MLLMs), which are expected to solve tasks spanning basic visual question answering (Tong et al., 2024b) to multi-step geometric and numerical reasoning (Hao et al., 2025) grounded in images. In principle, external vision and language specialists (e.g., OCR, localization, calculators, retrieval) can compensate for brittle perception and provide checkable intermediate evidence, allowing an MLLM to focus on planning, composition, and verification rather than internalizing every perceptual skill. Yet, despite rapid progress, today's MLLMs (Bai et al., 2025; Chen et al., 2024b; Liu et al., 2023) remain fragile precisely where tool assistance should help most: multi-step perception–reasoning with interactive tool use. In real scenarios—such as reading a gas price panel, extracting the correct price via OCR, and then computing a budgeted quantity—success requires the model to (i) recognize that a tool is needed, (ii) choose the right tool, (iii) sequence tool calls coherently over several steps, and (iv) integrate observations into a correct final answer. Smaller open-source models often exhibit failure modes like skipping necessary tools, calling irrelevant tools, or overfitting to superficial tool-call patterns that do not generalize.

A common response to address multimodal tool calling is to scale supervised fine-tuning (SFT) traces (Ma et al., 2025b; Hu et al., 2024). However, these approaches are misaligned with interactive execution (see Fig. 1 for LATTE (Ma et al., 2025a)): optimizing next-token likelihood alone may encourage imitation of tool traces rather than robust decision-making, and can even degrade perceptual accuracy or generalization. Reinforcement learning (RL) (Zheng et al., 2025b; Su et al., 2025a; Song et al., 2026; Lu et al., 2025a) offers a path to align behavior with end-task success, but naïvely applying RL to multimodal tool rollouts is challenging: rewards are often sparse, credit assignment

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

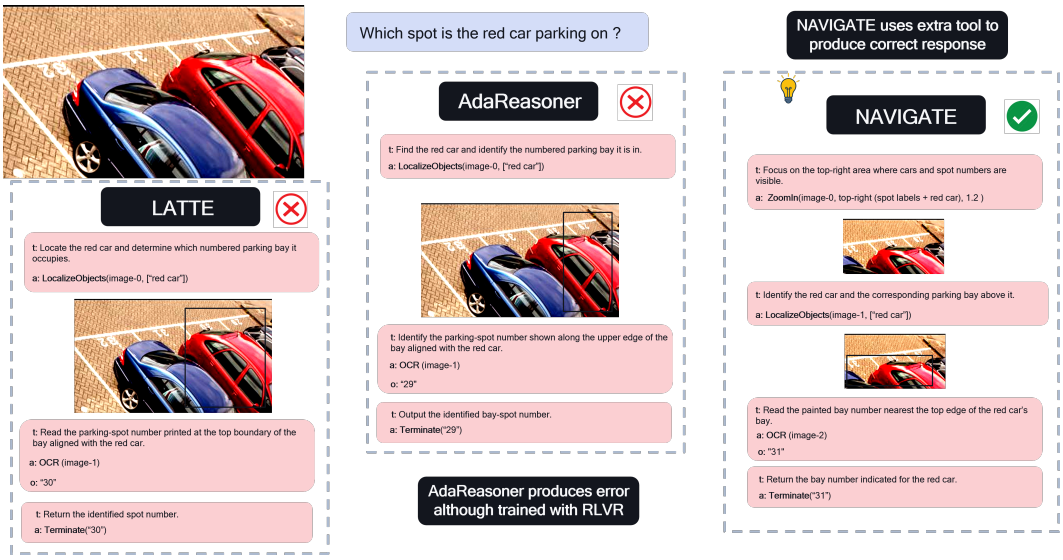


Figure 1: A visual example shows how competing methods like LATTE and AdaReasoner fail to invoke the correct tool and produce an incorrect answer. Our proposed framework NAVIGATE produces the correct answer, after invoking the correct tool.

spans multiple steps, and without structure, the policy can drift towards brittle heuristics (e.g., tool spamming or premature termination) (see Fig. 1 for AdaReasoner (Song et al., 2026)).

We also argue that effective multimodal tool orchestration hinges on a deceptively simple capability that is often under-specified: predicting the right next actions—and doing so in a way that remains verifiable at the end yet flexible during the rollout. Concretely, a tool-using MLLM should behave like an agent: it should plan which tool to invoke next, execute the call, interpret the observation, and repeat—while continuously maintaining internal consistency and resisting spurious tool patterns.

Multimodal tool orchestration creates an uneven, unstable learning problem because tool trajectories vary greatly in frequency, length, structure, perceptual difficulty, and reasoning depth. Training on the full distribution too early often yields brittle policies that overfit shallow cues or mismanage tools during long rollouts. A curriculum is therefore essential: begin with short, high-signal trajectories showing basic tool use, then gradually introduce harder samples with more tools, deeper dependencies, and multi-step reasoning. This progression stabilizes optimization and reduces failure modes by first establishing reliable perception and tool triggering before teaching robust tool composition.

To this end, we introduce NAVIGATE (shown in Fig. 2), a multi-stage framework that combines (i) a tool-aware curriculum that stabilizes grounding early, (ii) explicit next-action prediction to internalize planning, and (iii) RL with verifiable and flexible rewards to improve interactive success without overfitting to trace templates. The contributions are detailed as follows:

- *Tool-aware adaptive sampling:* NAVIGATE scores trajectories by length, compositionality, and perceptual/reasoning hardness, then trains with an easy-to-hard curriculum (Bengio et al., 2009; Kumar et al., 2010)—starting on short, high-signal tool uses and progressing to longer, compositional multi-tool chains.
- *Supervised fine-tuning with next-action prediction:* NAVIGATE augments the MLLM with a planning head that predicts future tool-action sequences, providing look-ahead supervision.
- *Reinforcement learning with verifiable and flexible rewards:* NAVIGATE uses GRPO (Mroueh et al., 2025)-style rollouts with verifiable rewards (terminal correctness, format compliance) and flexible regularizers (step consistency, embedding-transport, variance control) to promote coherent multi-step behavior, while preserving end-task accountability.
- Across multiple multimodal benchmarks, NAVIGATE consistently outperforms instruction-tuning and distillation baselines, lifting averages by around 10% and delivering strong gains in tool-intensive settings. Extensive ablations further dissect each component's contribution and behavior.

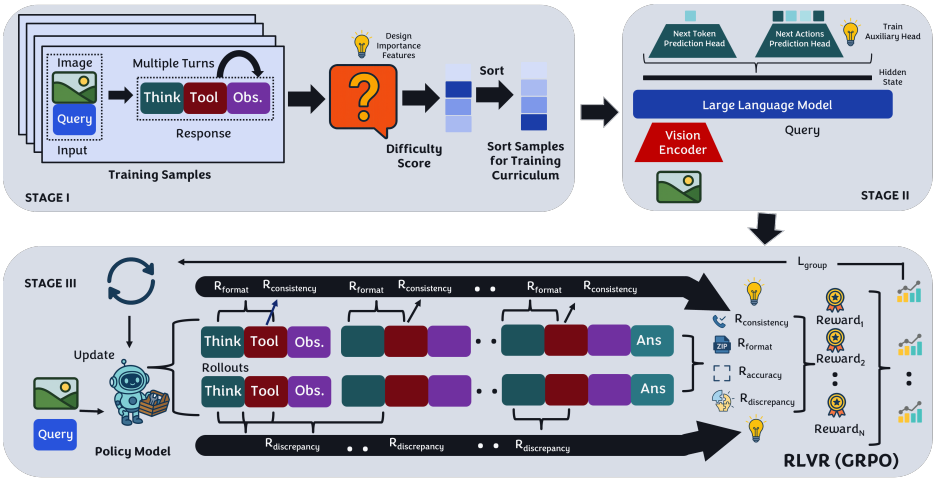


Figure 2: Our framework NAVIGATE consists of: (a) Tool-Aware Adaptive Sampling, (b) Supervised Fine-tuning with Next Action Prediction, and (c) Reinforcement Learning using Adaptive Tool Usage.

2 Related Work

2.1 Reinforcement Learning for Multimodal Reasoning

The trajectory of multimodal reasoning research has been heavily steered by the influence of DeepSeek-R1 (Guo et al., 2025). Its central insight—that rule-oriented Group Relative Policy Optimization (GRPO) can instill robust, structured reasoning behaviors in large language models—sparked a wave of interest in adapting this paradigm beyond pure text. Since then, a diverse collection of studies has broadened the scope of R1-style methods. For example, (Zhou et al., 2025a) captured the hallmark "phase-shift" phenomenon in multimodal reasoning; R1-OneVision (Yang et al., 2025) organized cross-modal reasoning into a more systematic pipeline; and work on video reasoning (Feng et al., 2025; Li et al., 2025) extended these ideas to handle temporal structure. Additional research (Huang et al., 2025; Shen et al., 2025; Lu et al., 2025b) has shown that these approaches can perform competitively across a wide variety of multimodal benchmarks. Despite these advancements, existing R1-based frameworks share a notable limitation: the rule-based reward signals they rely on primarily shape how the model reasons, not how well it perceives. When visual understanding is imperfect, those perceptual errors propagate through the reasoning chain, ultimately producing hallucinations or unreliable outcomes. NAVIGATE is designed specifically to overcome this weakness. By injecting precise perceptual cues from specialized tools, it ensures that the reasoning process is anchored to an accurate and faithful interpretation of the visual inputs.

2.2 Multimodal Reasoning with Tool Calling

Efforts to equip MLLMs with effective tool-use abilities have expanded rapidly. Early initiatives include LLaVA-Plus (Liu et al., 2024) which created a dedicated server architecture that MLLMs can query for external functionalities. Parallel work emphasized data generation: CogCoM (Qi et al., 2024) proposed a framework centered on six manipulation patterns and produced synthetic chains of manipulation, while TACO (Ma et al., 2025a) assembled extensive reasoning trajectories drawn from interactions with numerous visual tools. One branch trains models to reason through code execution (Zhang et al., 2025b; Zhao et al., 2025), enabling precise visual analysis at cost of scalability—heavy computational tools or expert models remain difficult to integrate in such environments. Another branch leverages lightweight perceptual tools, such as zooming or local inspection modules (Wang et al., 2025; Zheng et al., 2025b; Su et al., 2025a; Zhu et al., 2025b; Su et al., 2025b), which can sharpen visual understanding but generally operate as single, isolated actions rather than coordinated sequences. Although AdaReasoner (Song et al., 2026) moves towards dynamic selection among tools, its reasoning horizon is short and does not accommodate longer, adaptive workflows. Similarly, recent large-scale agentic training efforts (Lu et al., 2025a) still rely on fixed tool interfaces, constraining flexibility. NAVIGATE is introduced to overcome these limitations by enabling models to plan across multiple turns and assemble complex, context-dependent tool sequences, offering more capable and adaptive framework for MLLM tool use.

3 PROPOSED FRAMEWORK

We equip MLLMs to solve complex multimodal tasks by orchestrating vision and language tools, using data that (i) triggers specialists and (ii) reasons over their outputs, iterated until termination. Following (Ma et al., 2025a), each sample is a trajectory $\mathcal{T}$ for a query–image pair: $\mathcal{T} = (S_1, \ldots, S_n), \quad S_i = (\tau_i, a_i, o_i)$. Here, τ_i is thinking trace (words), a_i action selects specialist tool, and o_i is tool's output. The model predicts τ_i and a_i; the loss applies only to these, while o_i (from frozen specialists) is excluded. NAVIGATE leverages these inputs and trajectories to train a tool-using MLLM for complex reasoning.

3.0.1 STAGE I: TOOL-AWARE ADAPTIVE SAMPLING

Let each training sample have an associated LATTE trace. For the i^{th} training sample, we have the corresponding input $(Q^{(i)}, I^{(i)})$, where $Q^{(i)}$ is the input query and $I^{(i)}$ is the input image. Let the associated trajectory or trace be defined as:

$$\mathcal{T}^{(i)} = \big((\tau_1^{(i)}, a_1^{(i)}, o_1^{(i)}), (\tau_2^{(i)}, a_2^{(i)}, o_2^{(i)}), \ldots, (\tau_{n^{(i)}}^{(i)}, a_{n^{(i)}}^{(i)}, o_{n^{(i)}}^{(i)})\big), \tag{1}$$

Here, $\tau_*^{(i)}$ is the thinking trace, $a_*^{(i)}$ is the action i.e. the specialist tool-call and $o_*^{(i)}$ is the observation produced from the specialist. During training, we have following tuples associated with sample i such that $x^{(i)} = \big(Q^{(i)}, I^{(i)}, \mathcal{T}^{(i)}\big)$. Associated with each training sample, we proceed to compute a difficulty score that will facilitate sampling over the curriculum.

For sample x, we define normalized features $\mathbf{z}(x) \in [0,1]^6 : \mathbf{z}(x) = \big[\hat{m}(x), \hat{u}(x), \hat{L}(x), \hat{R}(x), \hat{K}(x), \hat{\ell}(x)\big]$ where $\hat{m}$ = number of tool calls, $\hat{u}$ = number of *distinct* tools, $\hat{L}$ = number of turns in the trace, $\hat{R}$ = reasoning hardness (e.g., length of thoughts, presence of math via CALCULATE/SOLVEMATHEQUATION, multi-image), $\hat{K}$ = knowledge reliance (QUERYKNOWLEDGEBASE/QUERYLANGUAGEMODEL presence), and $\hat{\ell}$ = likelihood hardness (details disclosed in supplementary). We use normalized features to obtain difficulty score $D_s(x)$, modulated by non-negative weight w_s. This weight varies with each training step s, such that $D_s(x) = w_s^\top z(x)$, with $\|w_s\|_1 = 1$. We design modulating weight such that we initially emphasize perception and basic tool orchestration. Later, we can emphasize compositionality and reasoning.

We use difficulty score $D_s(x)$ for a time-indexed sampling distribution $p_s(x)$ over the dataset $\mathcal{D}$ that: (a) targets a difficulty band that increases over time, (b) preserves tool diversity to counter imbalance, and (c) adapts to model weaknesses via $\hat{\ell}_s(x)$. We consider the ring sampler as the default. Let the target difficulty be $\mu_s \in [0,1]$ with bandwidth $\sigma_s > 0$. The stage–indexed sampling distribution is

$$\boxed{p_s(x) \;\propto\; \tilde{p}(x)\; \exp\!\left(-\frac{(D_s(x) - \mu_s)^2}{2\sigma_s^2}\right),} \tag{2}$$

where $\tilde{p}(x)$ is a base distribution (uniform over LATTE–useful sources) (Ma et al., 2025a). Based on this, we deterministically divide the whole training process into four stages. The division is done based on sorting of each sample based on their $p_s(x)$. For these four stages, we would have differing values of w_s, μ_s and σ_s. This would result in four stages with the following characteristics: (a) Stage 1 (Boot): Emphasize short traces (low $\hat{m}$, $\hat{u}$,$\hat{L}$), basic perception tasks (b) Stage 2 (Perception $\rightarrow$ Reasoning) Increased importance of $\hat{u}$, $\hat{L}$, multi-image and math tasks (c) Stage 3 (Compositional Tools) These are target traces with more tool integration. (d) Stage 4 (Robustness & Edge Cases) Include more weight on knowledge query $\hat{K}$, longer traces $\hat{L}$, diverse tools and hard OCR cases.

3.0.2 STAGE II: SUPERVISED FINE-TUNING WITH NEXT ACTION PREDICTION

We augment the MLLM with a *flat, parallel action head* that forecasts distribution over A tool actions across T steps ($T \geq n^{(i)}$ for sample i). Acting as *implicit planner*, it maps hidden states to time-indexed plan of shape $B \times L \times T \times A$, trained with masked cross-entropy (CE). This addresses SFT-only limits in dynamic, multi-step invocation—preserving cold-start benefits while enabling interactive end-task optimization (Su et al., 2025b; Liu et al., 2024; Zhou et al., 2025b).

We first redefine the next-token prediction loss using a standard LM head as $Z^{\mathrm{lm}} = HW_{\mathrm{lm}} + b_{\mathrm{lm}}$, where $W_{\mathrm{lm}} \in \mathbb{R}^{d \times V}$, $b_{\mathrm{lm}} \in \mathbb{R}^V$, and $Z^{\mathrm{lm}} \in \mathbb{R}^{B \times L \times V}$. where H is hidden state, d is hidden

dimension and V is vocabulary size. With teacher forcing, we define targets $Y \in \{0, \ldots, V-1\}^{B\times L}$ as input ids shifted by $+1$ position and dropping final token from loss. Using masked positions $M \in \{0,1\}^{B\times L}$, the NTP CE loss is

$$\mathcal{L}_{\text{NTP}} = -\frac{1}{\sum M}\sum_{b=1}^{B}\sum_{\ell=1}^{L} M_{b,\ell}\ \log\Big(\text{softmax}\big(Z^{\text{lm}}_{b,\ell,:}\big)_{Y_{b,\ell}}\Big). \tag{3}$$

This is conventional masked cross-entropy loss used for backbone training in LLMs and MLLMs.

We attach a *planning* head that projects H to $A \times T$ logits per token and reshapes to (T, A):

$$Z^{\text{plan}} = H\,W_{\text{plan}} + b_{\text{plan}},\quad W_{\text{plan}} \in \mathbb{R}^{d\times(AT)},\ b_{\text{plan}} \in \mathbb{R}^{AT}, Z^{\text{plan}} \in \mathbb{R}^{B\times L\times(AT)} \xrightarrow{\text{reshape}} \mathbb{R}^{B\times L\times T\times A}. \tag{4}$$

Let $p_{b,\ell,t,:} = \text{softmax}\big(Z^{\text{plan}}_{b,\ell,t,:}\big) \in \Delta^{A-1}$. be time-indexed probability. Given trajectory labels $Y^{\text{plan}} \in \{0,..,A-1\}^{B\times L\times T}$ and horizon mask $M \in \{0,1\}^{B\times L\times T}$, the masked CE is

$$\mathcal{L}_{\text{CE(plan)}} = -\frac{1}{\sum M}\sum_{b,\ell,t} M_{b,\ell,t}\ \log\Big(p_{b,\ell,t,\,Y^{\text{plan}}_{b,\ell,t}}\Big). \tag{5}$$

This head learns explicit look-ahead plan over tool actions and timesteps, complementing token-level NTP and matching cold-start SFT setup in (Su et al., 2025b; Wu et al., 2025). It causally predicts only future actions, padding unused steps with TERMINATE action. Because cross-entropy treats actions as orthogonal despite semantic overlap, we instead regress directly to the tool-action embeddings.

$$\widehat{E} = \text{reshape}\big(H\,W_{\text{emb}} + b_{\text{emb}}\big) \in \mathbb{R}^{B\times L\times T\times E}, W_{\text{emb}} \in \mathbb{R}^{d\times(TE)},\ b_{\text{emb}} \in \mathbb{R}^{TE}. \tag{6}$$

We denote $\widehat{e}_{b,\ell,t} \in \mathbb{R}^E$ the predicted word embedding for batch b, position ℓ, timestep t. E is the dimension of the word embedding space. Based on this, we compute the regression loss as follows:

$$\boxed{\mathcal{L}_{\cos} = \frac{1}{\sum M}\sum_{b,\ell,t} M_{b,\ell,t}\ \Big(1 - \cos\big(\widehat{e}_{b,\ell,t},\ e_{b,\ell,t}\big)\Big), \cos(x,y) = \frac{x^\top y}{\|x\|\,\|y\|}.} \tag{7}$$

$e_{b,\ell,t}$ is embedding of ground truth tool action at batch b, position ℓ, timestep t.

3.0.3 STAGE III: REINFORCEMENT LEARNING USING ADAPTIVE TOOL USAGE

SFT does not directly supervise the final answer. To address this limitation, we introduce Group Relative Policy Optimization (GRPO), extended to support rollouts from vision tools. GRPO samples G trajectories from the current frozen policy, where the trajectory for the i^{th} sample is defined as $\big((\hat{\tau}_{i,1}, \hat{a}_{i,1}, \hat{o}_{i,1}), \ldots, (\hat{\tau}_{i,n_i}, \hat{a}_{i,n_i}, \hat{o}_{i,n_i})\big)$

For every timestep t, we have step-level importance ratio $r_{i,t}$, which equals likelihood ratio between current model and old model. Furthermore, we obtain group relative advantages $\hat{A}_{i,t}$ that can be derived from reward $\hat{R}_{i,t}$ within a sampled batch. Using this, the GRPO training objective can be computed as below where importance ratio is $r_{i,t}(\theta)$.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\{\rho_i\},\{\omega_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{n_i}\sum_{t=1}^{n_i}\min\Big(r_{i,t}(\theta)\,\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big)\,\hat{A}_{i,t}\Big)\right] - \beta\, D_{\text{KL}}\big[\pi_\theta \,\|\, \pi_{\text{ref}}\big], \tag{8}$$

$$r_{i,t}(\theta) = \frac{\pi_\theta(\tau_{i,t}, a_{i,t} \mid q,\ \tau_{i,<t}, a_{i,<t}, o_{i,<t})}{\pi_{\theta_{\text{old}}}(\tau_{i,t}, a_{i,t} \mid q,\ \tau_{i,<t}, a_{i,<t}, o_{i,<t})}.$$

Here, ϵ controls clipping range and β scales KL penalty against a reference policy π_{ref}. To compute advantages, we need to compute following set of rewards:

Accuracy and Format Reward For the accuracy reward, we employ a simple, rule-driven accuracy signal. For the i-th item with ground truth o_{i,n_i} and model prediction $\hat{o}_{i,n_i}$, the terminal reward is $\hat{R}^{\text{acc}}_{i,t} = R_i\big(o_{i,n_i}, \hat{o}_{i,n_i}\big)$, which is 1 if $\text{is_same}\big(o_{i,n_i}, \hat{o}_{i,n_i}\big)$ and 0 otherwise, where $\text{is_same}(\cdot,\cdot)$ checks equivalence of o_{i,n_i} and $\hat{o}_{i,n_i}$; $\hat{o}_{i,n_i}$ is produced by `TERMINATE`. This reward is sparse, computed only at the final step. We also use intermediate rewards—format, consistency, and transport—and avoid step-wise accuracy to prevent overfitting and preserve cross-dataset generalization. For the format reward $\hat{R}^{form}_{i,t}$, we consider a reward value of 1 when the format is followed while we assign a value of 0 when the format is not followed as introduced in (Ma et al., 2025a).

Consistency reward For the consistency reward, we employ a simple, rule-driven signal, where we match the actions of the token prediction head and that of the action prediction head just before action tool is called. For the i^{th} sample and the t^{th} time-step with action prediction on token head as $\hat{a}_{i,t}$ and that of the action prediction head be $\hat{a}^{act}_{i,t}$, the reward is defined as

$$\boxed{\hat{R}^{cons}_{i,t} = R_{i,t}\big(\hat{a}_{i,t}, \hat{a}^{act}_{i,t}\big) = \begin{cases} +1, & \text{if is_same}\big(\hat{a}_{i,t}, \hat{a}^{act}_{i,t}\big), \\ 0, & \text{otherwise}, \end{cases}} \tag{9}$$

where $\text{is_same}(\cdot,\cdot)$ is a function that checks whether $\hat{a}_{i,t}$ and $\hat{a}^{act}_{i,t}$ are equivalent. It is important to note that this reward is applied on each step individually and averaged out to obtain the final reward.

Discrepancy Reward We compute a transport cost between input and output token embeddings at each trajectory step. For sample i and timestep t: inputs $Ip_{i,t} \in \mathbb{R}^{n_{Ip} \times d}$ concatenate vision+query; outputs $Op_{i,t} \in \mathbb{R}^{n_{Op} \times d}$ concatenate thinking+action. Let $Ip_{i,t} = \{x_k\}_{k=1}^{n_{Ip}} \subset \mathbb{R}^d$ and $Op_{i,t} = \{y_\ell\}_{\ell=1}^{n_{Op}} \subset \mathbb{R}^d$, with weights $u \in \Delta^{n_{Ip}}$, $v \in \Delta^{n_{Op}}$ (uniform or attention-derived). All embeddings are ℓ_2-normalized to produce $\tilde{x}_k$ and $\tilde{y}_l$, respectively. Then we can proceed to describe maximum mean discrepancy (MMD). For describing MMD, we define a kernel function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that is positive definite. This is used to create a biased estimator such that

$$\widehat{\text{MMD}}^2(Ip_{i,t}, Op_{i,t}) = \sum_{k,k'} u_k u_{k'}\, \phi\big(\tilde{x}_k, \tilde{x}_{k'}\big) + \sum_{l,l'} v_l v_{l'}\, \phi\big(\tilde{y}_l, \tilde{y}_{l'}\big) - 2\sum_{k,l} u_k v_l\, \phi\big(\tilde{x}_k, \tilde{y}_l\big). \tag{10}$$

The weights u_k, v_l are uniform i.e. $\frac{1}{n_{Ip}}$, $\frac{1}{n_{Op}}$, with kernel as RBF and per-step MMD reward is

$$\boxed{\hat{R}^{MMD}_{i,t} = 1 - \min\Big(1,\ 0.5 * \widehat{\text{MMD}}^2\Big).} \tag{11}$$

We also penalize variance across timesteps, ensuring stable and grounded output semantics. This regularization encourages shorter, more concise responses that maintain consistent meaning.

$$\boxed{\hat{R}^{var}_{i,t} \ = \ 1 - \Omega(\Sigma_{i,t}, \Sigma_{i,t-1}) \in \ [0,1].} \tag{12}$$

where $\Sigma_{i,t}$ is the covariance obtained from $Op_{i,t}$ and $\Omega(\cdot)$ is a discrepancy metric computed between covariances across timesteps $t-1$ and t. We consider the following discrepancy metric as default.

$$\Omega(\Sigma_{i,t}, \Sigma_{i,t-1}) \ = \ \frac{\|\Sigma_{i,t} - \Sigma_{i,t-1}\|_F}{\|\Sigma_{i,t}\|_F + \|\Sigma_{i,t-1}\|_F} \ \in \ [0,1]. \tag{13}$$

The total reward is then computed as gated aggregation over the format reward as follows

$$\boxed{\hat{R}_{i,t} = \hat{R}^{form}_{i,t} \times (\hat{R}^{acc}_{i,t} + \hat{R}^{cons}_{i,t} + \hat{R}^{MMD}_{i,t} + \hat{R}^{var}_{i,t})} \tag{14}$$

The gated format reward enforces the format in model responses enforced through SFT cold-starting.

4 Experiments

We follow the experimental setup of (Ma et al., 2025a) and evaluate small 7–8B multimodal models on six benchmarks. Since our reasoning traces use multiple images, we adopt multi-image–capable models, primarily Mantis-8B-SigLIP-LLaMA-3 (Jiang et al., 2024), Mantis-8B-CLIP-LLaMA-3 (Jiang et al., 2024) and LLaVA-OneVision-7B (Li et al., 2024) to demonstrate generalizability. Models are fine-tuned from pretrained or instruction-tuned checkpoints to analyze where LATTE-traces provide gains. Using hyperparameters from prior work, we train both the LM and projector for one epoch with a $1e$-5 learning rate on A100 or H100 GPUs. Our evaluation spans three perception benchmarks (RealWorldQA (xAI, 2024), CV-Bench (Tong et al., 2024a), BLINK (Fu et al., 2024)) and three that assess reasoning (MathVista (Lu et al., 2024), MMStar (Chen et al., 2024a), MMVet (Yu et al., 2023)). Following VLMEvalKit (Duan et al., 2024), we use GPT-4-turbo (OpenAI, 2023) as an LLM judge to score open-ended answers between 0 and 1. Additional details are provided in the supplementary.

Table 1: **Comparison studies.** All models were trained with 98K LATTE trace version and evaluated with both Perception and Perception + Reasoning benchmarks. * implies zero-shot performance.

Method	Perception				Perception + Reasoning				Overall
	BLINK	CV-Bench	RealWorldQA	Avg	MathVista	MMStar	MMVet	Avg	Avg
GPT 5* (OpenAI, 2025)	56.3	62.2	54.1	57.5	47.8	56.2	58.9	54.3	55.9
Claude 4.5 Sonnet* (Anthropic, 2025)	57.0	61.2	54.3	57.5	48.0	56.0	58.5	54.2	55.8
Gemini 2.5 Pro* (Comanici et al., 2025)	58.3	63.4	55.0	58.9	48.2	56.5	59.0	54.6	56.8
Qwen 2.5 VL 72B* (Bai et al., 2025)	50.1	56.3	49.3	51.9	40.2	45.9	50.2	45.4	48.7
InternVL3 78B* (Zhu et al., 2025a)	51.3	58.2	50.1	53.2	42.6	47.9	53.2	47.9	50.6
Vanilla IT (Liu et al., 2023)	44.1	49.2	41.4	44.9	31.0	39.7	27.8	32.8	38.9
VPD (Hu et al., 2024)	41.6	48.8	44.8	45.1	33.0	41.1	32.8	35.7	40.4
LLaVa-CoT (Xu et al., 2025)	42.2	40.4	38.0	40.2	36.7	44.6	40.2	40.5	40.4
LATTE (Ma et al., 2025a)	46.4	54.0	42.0	47.5	36.9	44.2	47.9	43.0	45.2
LLaVA-Plus (Liu et al., 2024)	44.2	51.6	42.5	46.1	34.4	41.5	45.6	40.5	43.3
CogCoM (Qi et al., 2024)	44.0	51.8	42.3	46.0	33.8	41.3	44.7	39.9	43.0
MatchTIR (Qu et al., 2026)	46.8	54.5	45.4	48.9	36.5	44.8	47.8	43.0	46.0
AdaReasoner (Song et al., 2026)	47.5	55.0	45.7	49.4	37.9	45.3	49.0	44.1	46.8
PixelReasoner (Su et al., 2025a)	46.7	54.5	45.0	48.7	36.2	44.8	47.5	42.8	45.8
DeepEyesv2 (Hong et al., 2025)	47.0	54.6	45.6	49.1	37.2	44.7	48.2	43.4	46.2
Visual-ARFT (Liu et al., 2025)	46.9	54.5	45.3	48.9	36.6	44.4	47.6	42.9	45.9
Vista-R1 (Lu et al., 2025a)	47.2	54.9	45.2	49.1	37.7	45.1	48.8	43.9	46.5
V-ToolRL (Su et al., 2025b)	47.1	54.0	45.4	48.8	36.8	44.2	47.9	43.0	45.9
R1-VL (Zhang et al., 2025a)	46.8	54.5	45.3	48.9	35.4	44.1	46.6	42.0	45.5
R1-Onevision (Yang et al., 2025)	47.0	54.7	45.6	49.1	37.4	44.8	48.3	43.5	46.3
Perception-R1 (Yu et al., 2025a)	46.9	54.6	45.4	49.0	36.3	44.6	47.6	42.8	45.9
CodeV (Hou et al., 2026)	48.6	56.4	47.1	50.7	39.8	46.8	50.4	45.7	48.2
ACTIVE-o3 (Zhu et al., 2025b)	49.7	57.2	48.9	51.9	38.4	46.1	49.5	44.7	48.3
ARM-Thinker (Ding et al., 2026)	49.2	56.8	47.8	51.3	39.5	47.0	50.1	45.5	48.4
NAVIGATE w/o Stage I	51.2	58.7	49.4	53.1	41.3	48.5	52.4	47.4	50.2
NAVIGATE w/o Stage II	51.4	58.9	49.8	53.4	41.6	48.6	52.8	47.6	50.5
NAVIGATE w/o Stage III	50.9	58.3	49.3	52.8	40.6	48.7	51.7	47.0	49.9
NAVIGATE w/o Stage I & II	49.9	57.7	48.1	51.9	39.5	47.2	49.9	45.5	48.7
NAVIGATE w/o Stage II & III	47.6	55.4	46.2	49.7	37.2	45.7	48.6	43.9	46.8
NAVIGATE w/o Stage I & III	48.5	56.3	46.8	50.5	38.1	46.6	49.5	44.7	47.6
NAVIGATE w/ Decoupled Planner	51.6	59.5	50.2	53.8	41.5	49.2	52.6	47.8	50.8
NAVIGATE	52.1	59.9	50.5	**54.2** (+4.8)	42.2	49.8	53.3	**48.4** (+4.3)	**51.3** (+4.5)

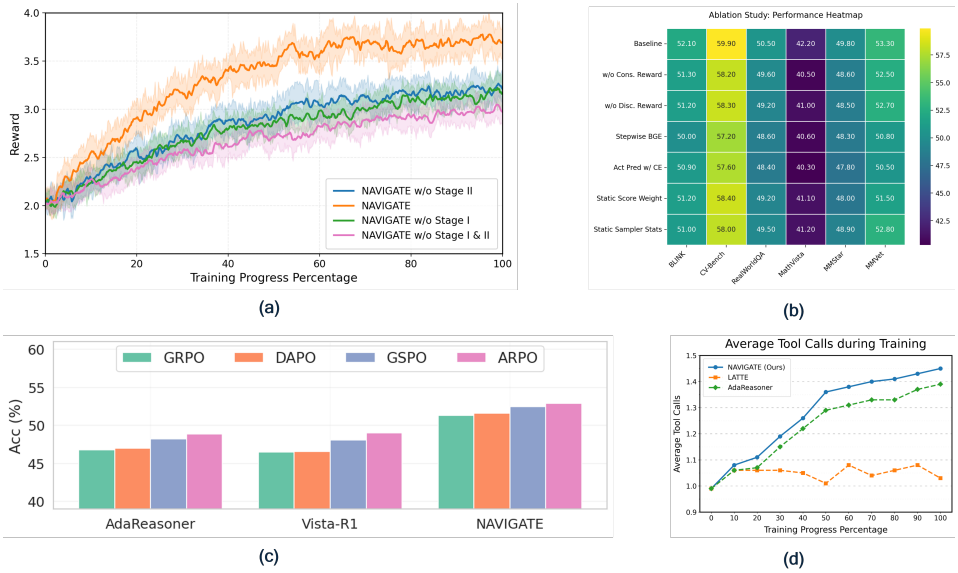


Figure 3: (a) Evolution of rewards with training for different ablations of NAVIGATE. (b) Accuracy variation for different ablations of NAVIGATE. (c) Effect of using different RL algorithms on competing methods. (d) Variation of average number of tool calls as training progresses.

Table 1 compares diverse multimodal baselines, including instruction-tuned models, perception and RL-style methods, and leading proprietary and open-source systems, against NAVIGATE and its ablations. NAVIGATE achieves the best open-source results, reaching 54.2 Perception accuracy, 48.4 Perception+Reasoning accuracy, and 51.3 overall. Although proprietary models remain ahead, NAVIGATE narrows the gap across both settings, improving both visual recognition and reasoning-intensive tasks. Ablation results show that removing any stage degrades performance: Stage I, Stage II and III as a whole contribute a lot to reasoning and other perception tasks. Pairwise-stage variants also under-perform the full three-stage pipeline, highlighting their complementarity. Finally, an integrated planner outperforms a decoupled alternative, suggesting that tighter planning-learning integration improves credit assignment.

Figure 3 (a) shows that NAVIGATE learns faster and achieves a higher final reward than all ablations. Removing either training stage reduces learning speed and performance, while removing both produces the weakest results, demonstrating that the stages are complementary. Figure 3 (b) reports accuracy across evaluation settings. NAVIGATE consistently outperforms its ablations, including variants that remove reward components or modify action prediction and curriculum sampling. Stepwise BGE-based verifiable rewards (Xiao et al., 2023) also reduce performance, likely due to overfitting. Figure 3 (c) compares GRPO, DAPO (Yu et al., 2025b), GSPO (Zheng et al., 2025a), and ARPO (Dong et al., 2025) across three systems. Differences among objectives are modest, whereas NAVIGATE consistently surpasses AdaReasoner and Vista-R1, suggesting that framework design matters more than the choice of RL objective. Finally, Figure 3 (d) shows that NAVIGATE increases tool use early and stabilizes at a higher level than the baselines. Together, these results indicate that staged training improves reward, accuracy, and proactive tool use.

Table 2: Ablations with different model backbones. * implies that the starting checkpoint is MantisPretrained. ** implies that the starting checkpoint is MantisInstruct-tuned. *** implies that the starting checkpoint is LLaVA-OVStage1.5.

Backbone	Method	Percept.	P. + Reason.	Overall
InternVL3-8B (Zhu et al., 2025a)	Vanilla IT	51.5	45.1	48.3
	LATTE	54.1	54.5	54.3
	AdaReasoner	56.3	56.8	56.6
	NAVIGATE	59.0	59.8	**59.4**
Qwen2.5VL-3B (Bai et al., 2025)	Vanilla IT	54.3	49.6	52.0
	LATTE	53.4	55.7	54.6
	AdaReasoner	55.0	57.2	56.1
	NAVIGATE	58.2	60.1	**59.2**
Qwen2.5VL-7B (Bai et al., 2025)	Vanilla IT	53.7	52.5	53.1
	LATTE	56.9	57.7	57.3
	AdaReasoner	58.2	58.9	58.6
	NAVIGATE	61.0	61.5	**61.3**
LLaMA3-8B / CLIP* (Grattafiori et al., 2024; Radford et al., 2021)	Vanilla IT	50.2	32.9	41.6
	LATTE	52.6	40.8	46.7
	AdaReasoner	54.5	42.2	48.3
	NAVIGATE	56.8	44.6	**50.7**
LLaMA3-8B / SigLIP* (Grattafiori et al., 2024; Zhai et al., 2023)	Vanilla IT	49.3	34.9	42.1
	LATTE	52.9	41.6	47.2
	AdaReasoner	54.8	43.9	49.4
	NAVIGATE	56.9	46.2	**51.6**
LLaMA3-8B / SigLIP** (Grattafiori et al., 2024; Zhai et al., 2023)	Vanilla IT	50.7	35.5	43.1
	LATTE	51.7	44.7	48.2
	AdaReasoner	53.8	46.8	50.3
	NAVIGATE	55.6	48.2	**51.9**
Qwen2-7B / SigLIP*** (Yang et al., 2024; Zhai et al., 2023)	Vanilla IT	55.0	43.9	49.5
	LATTE	58.0	51.2	54.6
	AdaReasoner	60.1	53.4	56.8
	NAVIGATE	62.6	55.6	**59.1**
LLAVA1.5-7B (Liu et al., 2023)	Vanilla IT	52.4	51.0	51.7
	LATTE	55.2	56.5	55.8
	AdaReasoner	57.2	58.0	57.6
	NAVIGATE	60.0	59.5	**59.7**
GLM-Edge-V-2B (Glm et al., 2024)	Vanilla IT	50.2	45.3	47.7
	LATTE	49.5	51.4	50.4
	AdaReasoner	51.2	53.6	52.4
	NAVIGATE	54.2	56.2	**55.2**
Qwen3.5-4B (Qwen Team, 2026)	Vanilla IT	56.0	52.5	54.3
	LATTE	56.5	58.3	57.4
	AdaReasoner	58.0	59.7	58.9
	NAVIGATE	**61.1**	**62.5**	**61.8**

Table 2 compares training strategies across several vision–language backbones and initialization regimes with 293K LATTE trace. Across all backbones, NAVIGATE (ours) consistently achieves best results, outperforming Vanilla IT, LATTE, and AdaReasoner baselines on both perception and reasoning metrics, and thus on overall score. The improvements hold regardless of the starting checkpoint, indicating that our procedure is largely initialization-agnostic and transfers reliably across architectures and model sizes. NAVIGATE delivers strong gains on reasoning and perception, indicating that the objective strengthens multi-step reasoning while also improving perceptual grounding. Although larger backbones achieve higher absolute scores, smaller models show clearest relative boosts, suggesting NAVIGATE is an effective, architecture-agnostic training recipe.

Figure 4 (a) shows a positive correlation between planning quality, measured by top-k action accuracy and sequence-level F1, and task accuracy for NAVIGATE without Stages I and III. This relationship indicates that planner confidence is a useful proxy for downstream success and can guide tool-budget allocation or confidence-based routing. Figure 4 (b) shows that increasing the inference-time tool budget improves all methods, with the largest gains occurring within the first one or two calls. NAVIGATE consistently outperforms LATTE and AdaReasoner, demonstrating more effective use of limited tool calls. Figure 4 (c) compares reward-aggregation strategies. The default gated mechanism achieves the highest accuracy by enforcing the format reward before incorporating other components, while mean-based alternatives perform worse. Figure 4 (d) analyzes errors on MMVet. Model-perception errors decrease from 50% for LATTE to 46% for AdaReasoner and 37% for NAVIGATE, whereas irrelevant-tool usage increases from 28% to 36% and 45%, respectively. Other tool-call and tool-result errors remain relatively small. Thus, as reasoning and perception improve, tool selection and routing become the primary bottlenecks. Finally, Figure 4 (e) shows accuracy throughout fine-tuning. NAVIGATE with Stages I and II performs best across training, while the ablations remain above LATTE but converge to lower plateaus. These results demonstrate complementary contributions from each stage and greater training efficiency for NAVIGATE.

Figure 5 considers the query, "What percentage does salary contribute to total income?" Given Salary = 30,000 and Total income = 53,350, NAVIGATE extracts the values, computes their ratio, rounds it, and returns 56.23%. Removing any stage causes failure: without Stage I, incorrect extraction yields 100.00%; without Stage II, improper tool sequencing also produces 100.00%; and without Stage III, missing verification accepts an implausible 331.49%. This example highlights the complementary roles of grounded extraction, tool-use planning, and final verification.

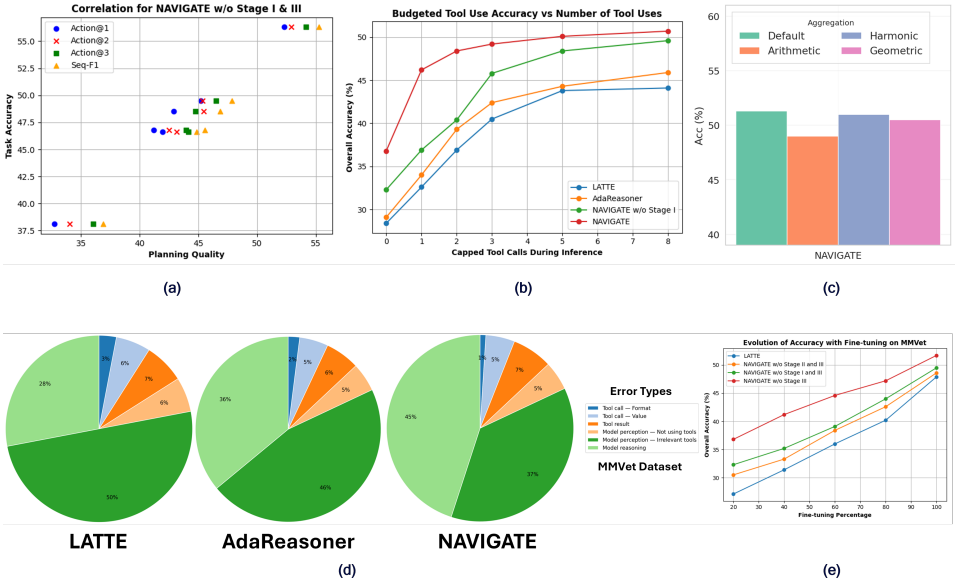


Figure 4: (a) Plot showing how final accuracy is correlated with planning quality. (b) Overall accuracy with respect to number of capped tool calls during inference. (c) Effect of different aggregation mechanisms over default gated mechanism. (d) Error distribution across different methods. (e) Accuracy evolution with training steps.

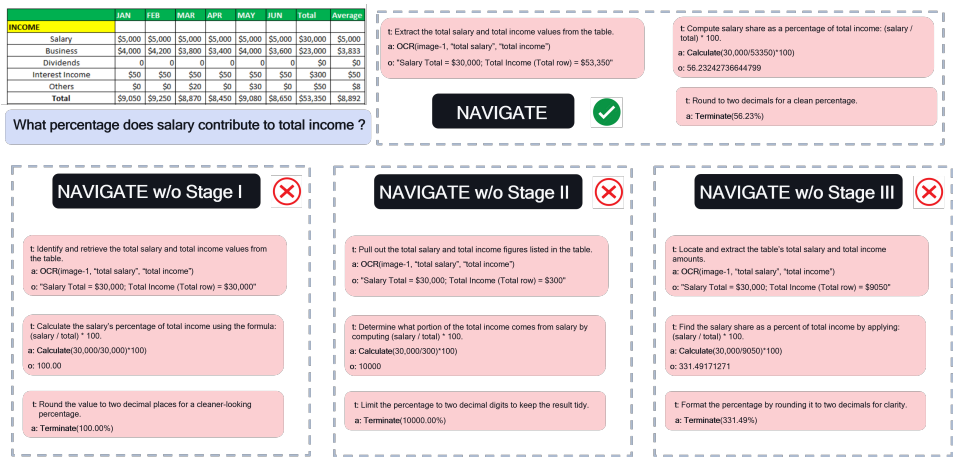


Figure 5: We demonstrate visually the importance of each stage in the NAVIGATE framework.

5 Conclusion

We introduced NAVIGATE, a three-stage framework for multimodal tool-augmented reasoning: (i) tool-aware adaptive sampling to stabilize perceptual grounding, (ii) SFT with a next-action planner head (optionally embedding-regressed) to internalize look-ahead sequencing, and (iii) GRPO with verifiable (accuracy, format) and flexible (consistency, embedding-transport/MMD, variance) rewards to align rollouts with end-task success while curbing skipped or irrelevant tool calls. Across BLINK, CV-Bench, RealWorldQA, MathVista, MMStar, and MM-Vet, NAVIGATE yields strong gains over existing SFT and RL baselines. Ablations confirm each stage's necessity; the integrated planner outperforms a decoupled variant, and gated aggregation improves fusion under small tool budgets. NAVIGATE thus offers a scalable recipe for verifiable, flexible tool orchestration in MLLMs; remaining gaps to frontier systems motivate work on larger backbones, richer math-vision curricula, and stronger planning-aware rewards. In the future work, we will explore cost-aware tool budgeting, dynamic tool discovery beyond fixed APIs.

REFERENCES

Anthropic. Claude sonnet 4.5 system card. System card, Anthropic, September 2025. URL https://www-cdn.anthropic.com/963373e433e489a87a10c823c52a0a013e9172dd.pdf.

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *arXiv preprint arXiv:2403.20330*, 2024a.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24185–24198, 2024b.

Andy Clark and David Chalmers. The extended mind. *analysis*, 58(1):7–19, 1998.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, and et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Technical report, Google DeepMind, 2025. URL https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf. Accessed: 2026-03-01.

Shengyuan Ding, Xinyu Fang, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiangyu Zhao, Haodong Duan, Xiaoyi Dong, Jianze Liang, Bin Wang, et al. Arm-thinker: Reinforcing multimodal generative reward models with agentic tool use and visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22195–22205, 2026.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 11198–11201, 2024.

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *arXiv preprint arXiv:2503.21776*, 2025.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. *arXiv preprint arXiv:2404.12390*, 2024.

Team Glm, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yunzhuo Hao, Jiawei Gu, Huichen Will Wang, Linjie Li, Zhengyuan Yang, Lijuan Wang, and Yu Cheng. Can mllms reason in multimodality? emma: An enhanced multimodal reasoning benchmark. *arXiv preprint arXiv:2501.05444*, 2025.

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

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

Yushi Hu, Otilia Stretcu, Chun-Ta Lu, Krishnamurthy Viswanathan, Kenji Hata, Enming Luo, Ranjay Krishna, and Ariel Fuxman. Visual program distillation: Distilling tools and programmatic reasoning into vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9590–9601, 2024.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

Dongfu Jiang, Xuan He, Huaye Zeng, Con Wei, Max Ku, Qian Liu, and Wenhu Chen. Mantis: Interleaved multi-image instruction tuning. *arXiv preprint arXiv:2405.01483*, 2024.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Xinhao Li, Ziang Yan, Desen Meng, Lu Dong, Xiangyu Zeng, Yinan He, Yali Wang, Yu Qiao, Yi Wang, and Limin Wang. Videochat-r1: Enhancing spatio-temporal perception via reinforcement fine-tuning. *arXiv preprint arXiv:2504.06958*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, et al. Llava-plus: Learning to use tools for creating multimodal agents. In *European conference on computer vision*, pp. 126–142. Springer, 2024.

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.

Meng Lu, Ran Xu, Yi Fang, Wenxuan Zhang, Yue Yu, Gaurav Srivastava, Yuchen Zhuang, Mohamed Elhoseiny, Charles Fleming, Carl Yang, et al. Scaling agentic reinforcement learning for tool-integrated reasoning in vlms. *arXiv preprint arXiv:2511.19773*, 2025a.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*, 2025b.

Zixian Ma, Jianguo Zhang, Zhiwei Liu, Jieyu Zhang, Juntao Tan, Manli Shu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Caiming Xiong, et al. Latte: Learning to think with vision specialists. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11203–11240, 2025a.

Zixian Ma, Jianguo Zhang, Zhiwei Liu, Jieyu Zhang, Juntao Tan, Manli Shu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Caiming Xiong, et al. Taco: Learning multi-modal models to reason and act with synthetic chains-of-thought-and-action. In *ICLR Workshop on Reasoning and Planning for Large Language Models*, 2025b.

Youssef Mroueh, Nicolas Dupuis, Brian Belgodere, Apoorva Nitsure, Mattia Rigotti, Kristjan Greenewald, Jiri Navratil, Jerret Ross, and Jesus Rios. Revisiting group relative policy optimization: Insights into on-policy and off-policy training. *arXiv preprint arXiv:2505.22257*, 2025. URL https://arxiv.org/abs/2505.22257.

OpenAI. Gpt-4 technical report. *arXiv*, abs/2303.08774, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. Gpt-5 system card. Technical report, OpenAI, 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf. Accessed: 2026-03-01.

Ji Qi, Ming Ding, Weihan Wang, Yushi Bai, Qingsong Lv, Wenyi Hong, Bin Xu, Lei Hou, Juanzi Li, Yuxiao Dong, et al. Cogcom: A visual language model with chain-of-manipulations reasoning. *arXiv preprint arXiv:2402.04236*, 2024.

Changle Qu, Sunhao Dai, Hengyi Cai, Jun Xu, Shuaiqiang Wang, and Dawei Yin. Matchtir: Fine-grained supervision for tool-integrated reasoning via bipartite matching. *arXiv preprint arXiv:2601.10712*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, February 2026. Accessed: 2026-08-17.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model, 2025. *arXiv preprint arXiv:2504.07615*, 2025.

Mingyang Song, Haoyu Sun, Jiawei Gu, Linjie Li, Luxin Xu, Ranjay Krishna, and Yu Cheng. Adareasoner: Dynamic tool orchestration for iterative visual reasoning. *arXiv preprint arXiv:2601.18631*, 2026.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2025a.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, et al. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025b.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Austin Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal llms, 2024a.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9568–9578, 2024b.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Jiacong Wang, Zijian Kang, Haochen Wang, Haiyong Jiang, Jiawen Li, Bohong Wu, Ya Wang, Jiao Ran, Xiao Liang, Chao Feng, et al. Vgr: Visual grounded reasoning. *arXiv preprint arXiv:2506.11991*, 2025.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

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

xAI. Grok-1.5 vision preview and realworldqa benchmark. `https://x.ai/blog/grok-1.5v`, 2024. RealWorldQA dataset released as part of the Grok-1.5V announcement.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

Guowei Xu, Peng Jin, Ziang Wu, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. Llava-cot: Let vision language models reason step-by-step. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2087–2098, 2025.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL `https://arxiv.org/abs/2407.10671`.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, et al. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *arXiv preprint arXiv:2503.10615*, 2025.

En Yu, Kangheng Lin, Liang Zhao, Jisheng Yin, Yana Wei, Yuang Peng, Haoran Wei, Jianjian Sun, Chunrui Han, Zheng Ge, et al. Perception-r1: Pioneering perception policy with reinforcement learning. *arXiv preprint arXiv:2504.07954*, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities, 2023.

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

Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *arXiv preprint arXiv:2503.12937*, 2025a.

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

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling. *arXiv preprint arXiv:2507.07998*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025a.

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. *arXiv preprint arXiv:2505.14362*, 2025b.

Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. R1-zero's" aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint arXiv:2503.05132*, 2025a.

Zetong Zhou, Dongping Chen, Zixian Ma, Zhihan Hu, Mingyang Fu, Sinan Wang, Yao Wan, Zhou Zhao, and Ranjay Krishna. Reinforced visual perception with tools. *arXiv preprint arXiv:2509.01656*, 2025b.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025a.

Muzhi Zhu, Hao Zhong, Canyu Zhao, Zongze Du, Zheng Huang, Mingyu Liu, Hao Chen, Cheng Zou, Jingdong Chen, Ming Yang, et al. Active-o3: Empowering multimodal large language models with active perception via grpo. *arXiv preprint arXiv:2505.21457*, 2025b.

A AI USE STATEMENT

We used generative AI tools to paraphrase, improve writing style, summarize, proof-check, assist with code generation, and monitor experiments across design choices. All technical claims and experimental results were verified by the authors.

B DETAILS OF TOOL AWARE ADAPTIVE SAMPLING

Notation. Let a trajectory be defined as $x = \big(S_1, \ldots, S_n\big)$, where $S_i = (\tau_i, a_i, o_i)$ are thought, action (tool call), and observation respectively. Let $\mathcal{D}$ denote the training set of traces. For any scalar quantity $v(x)$ computed per trace, we define a robust min–max normalization to $[0, 1]$ by

$$\mathrm{norm}(v(x)) \;=\; \mathrm{clip}_{[0,1]}\left(\frac{v(x) - q_5(v)}{q_{95}(v) - q_5(v) + \varepsilon}\right),$$

where $q_\alpha(v)$ is the α-th percentile of $\{v(x) : x \in \mathcal{D}\}$, $\varepsilon > 0$ is a small constant, and $\mathrm{clip}_{[0,1]}(\cdot)$ clamps values to $[0, 1]$. We set $\varepsilon = 0.05$

(1) Number of tool calls $m(x)$.

$$m_{\mathrm{raw}}(x) \;=\; \sum_{i=1}^{n} \mathbf{1}\big[a_i \neq \varnothing\big], \quad \hat{m}(x) \;=\; \mathrm{norm}\big(m_{\mathrm{raw}}(x)\big).$$

Here $\mathbf{1}[\cdot]$ is the indicator function. LATTE traces (Ma et al., 2025a) explicitly log actions per step.

(2) Number of distinct tools $u(x)$. Let $\mathcal{A}(x) = \{\, a_i : a_i \neq \varnothing \,\}$ be the multiset of tools invoked.

$$u_{\mathrm{raw}}(x) \;=\; \big|\mathrm{unique}(\mathcal{A}(x))\big|, \quad \hat{u}(x) \;=\; \mathrm{norm}\big(u_{\mathrm{raw}}(x)\big).$$

Distinct tool counts capture heterogeneity of specialist use in LATTE.

(3) Trace length (turns) $L(x)$.

$$L_{\mathrm{raw}}(x) \;=\; n, \qquad \hat{L}(x) \;=\; \mathrm{norm}\big(L_{\mathrm{raw}}(x)\big).$$

It is to be noted that the trace length might be different from the number of tool calls as a particular step might not invoke a tool.

(4) Reasoning hardness $R(x)$. We combine textual and structural proxies: long thoughts, math usage, and multi-image reasoning.

Let $\mathrm{tok}(t_i)$ be the token count of thought t_i; let $\mathbf{1}_{\mathrm{math}}(x) = \mathbf{1}\big[\exists i : a_i \in \{\mathrm{CALCULATE}, \mathrm{SOLVEMATHEQUATION}\}\big]$; Let $\mathrm{img}(x)$ be the number of distinct images referenced across steps. Define

$$R_{\mathrm{raw}}(x) = \alpha_1 \frac{\sum_{i=1}^{n} \mathrm{tok}(t_i)}{n} + \alpha_2 \mathbf{1}_{\mathrm{math}}(x) + \alpha_3 \mathrm{img}(x),$$

$$\hat{R}(x) = \mathrm{norm}\big(R_{\mathrm{raw}}(x)\big),$$

with weights $\alpha_1, \alpha_2, \alpha_3 > 0$ (e.g., α_1=1, α_2=2, α_3=1). LATTE traces include math/logic tools and multi-image compositions; longer, tool-involving thoughts typically indicate harder reasoning.

(5) Knowledge reliance $K(x)$. Count reliance on external knowledge/tools:

$$K_{\mathrm{raw}}(x) = \frac{\sum_{i=1}^{n} \mathbf{1}\big[a_i \in \{\mathrm{QKB}, \mathrm{QLM}\}\big]}{n},$$

$$\hat{K}(x) = \mathrm{norm}\big(K_{\mathrm{raw}}(x)\big).$$

This captures traces that explicitly query KB/LM, as LATTE supports tools that fetch external information. Here, QKB is QUERYKNOWLEDGEBASE and QLM is QUERYLANGUAGEMODEL.

(6) Likelihood hardness $\ell(x)$. Let $\ell(x)$ be the likelihood loss for trace x (e.g., sum of token-level NLL over thoughts and action tokens) on the model to be trained. We normalize as the following:

$$\hat{\ell}(x) = \mathrm{norm}\big(\ell(x)\big).$$

The hyper-parameters for computing the sampling distribution for each stage is as follows:

- Stage 1: $w_s = [0.305, 0.169, 0.136, 0.136, 0.085, 0.169]$ $\mu_s = 0.2, \sigma_s = 0.1$,
- Stage 2: $w_s = [0.250, 0.188, 0.156, 0.250, 0.063, 0.094]$ $\mu_s = 0.4, \sigma_s = 0.15$,
- Stage 3: $w_s = [0.221, 0.221, 0.176, 0.265, 0.029, 0.088]$ $\mu_s = 0.6, \sigma_s = 0.2$,
- Stage 4: $w_s = [0.200, 0.200, 0.200, 0.257, 0.057, 0.086]$ $\mu_s = 0.75, \sigma_s = 0.25$

In Fig. 6, we show the fraction of tools used in the different stages for the LATTE-Trace 98K dataset, when the above-mentioned sampling is used.

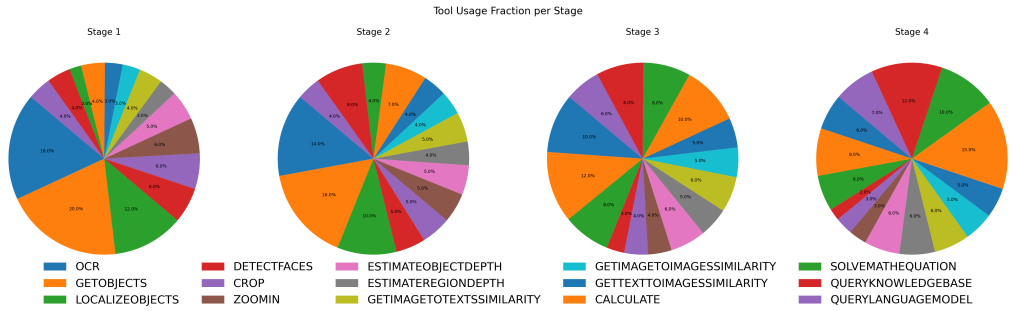


Figure 6: Pie-chart showing the fraction of tools in the LATTE-Trace 98K dataset.

C Implementation Details

For the SFT and action prediction stage of fine-tuning, as well as the prompt format, we mainly follow the setup of LATTE (Ma et al., 2025a). For Mantis-CLIP and Mantis-SigLIP based models, per-device batch size is set as 1 and distributed across 8 A100 GPUs. We use gradient accumulation

steps of 16. We use Adam optimizer with weight decay of 0 and warmup ratio of 0.03. We use a cosine-based learning rate scheduler. Maximum sequence length is set as 8192. The low-rank adaptor is disabled and data formats of bf16 and tf32 are used. Training is carried out for 1 epoch. This configuration is repeated for other models as well. We use the dataset formats as established in LATTE (Ma et al., 2025a)

For the LLava-OneVision model, we use a learning rate of 2e-6 for the vision tower. The projector uses 2 multilayer perceptrons with GeLU activation. The last two layers are then used for updating.

For the action prediction head, when using cross-entropy loss, we use the number of possible actions to be $A = 16$ including the `TERMINATE` action. For the horizon T we set it as 50. When using the regression-based model, we embed the tool names using the e5 base model (Wang et al., 2022) into an embedding of dimension $E = 768$. For the action prediction loss, we use a default loss weight of $1e - 4$.

For the action prediction, the intended supervision is attached only to designated decision positions, namely the hidden state immediately preceding an action emission. Let $M^{\text{dec}}_{b,\ell} \in \{0, 1\}$ identify these positions and $M^{\text{hor}}_{b,\ell,t}$ indicate valid forecast targets. The effective mask is

$$M^{\text{plan}}_{b,\ell,t} = M^{\text{dec}}_{b,\ell} M^{\text{hor}}_{b,\ell,t}. \tag{15}$$

For a decision position, the target consists of the remaining ground-truth actions followed by one explicit TERMINATE label. Positions after the first TERMINATE are padding and receive mask value zero. Thus TERMINATE is learned as an action, while repeated padding is ignored.

For the RL fine-tuning stage, we use the same hyper-parameters as SFT. For the GRPO stage, we sample $G = 8$ rollouts per prompt with synchronous on-policy generation (limited to 512 generated tokens and a maximum prompt length of 1024 tokens), and optimize with a per-device prompt batch size of 1 and 16 gradient-accumulation steps for one epoch over the training set, using a learning rate of $1\text{e}{-6}$, clipping range $\epsilon = 0.1$, and KL coefficient $\beta = 0.01$.

For the consistency reward, we consider $\hat{a}^{act}_{i,t}$ as the output of the action prediction head just before the action tool is called using token prediction. For the discrepancy reward, we use the RBF kernel where the bandwidth $\sigma = \sqrt{0.5 f(d^2)}$. Here, $f(\cdot)$ is the median function and d is the pairwise distance between the data points. Next, we proceed to describe baselines and ablations described in the main paper.

- *NAVIGATE w/ Decoupled Planner*: In this setup, we do not have stage II, that is the stage for fine-tuning action prediction head. Rather, we train a separate multimodal model that predicts the action sequence, given the input image and action sequence. Once the actions i.e. tools are known ahead of time, it is then added as contextual information to the model doing next token prediction. The format used is: `CONTEXT: Use the following tools: <Tool 1>, <Tool 2>.....`
- *w/o Cons. Reward*: In this setup, we do not use the consistency reward $R^{cons}_{i,t}$ in the reinforcement fine-tuning stage.
- *w/o Disc. Reward*: In this setup, we do not use the discrepancy rewards $\hat{R}^{MMD}_{i,t}$, $\hat{R}^{var}_{i,t}$ in the reinforcement fine-tuning stage
- *Stepwise BGE*: In this setup, we consider an additional reward based on the ground truth text and model predictions. For the ground truth and prediction per reasoning step, we consider the BGE (Xiao et al., 2023) embedding. The reward is considered as $0.5 * (1 + \cos(r_{gt}, r_{pred}))$, where r_{gt} and r_{pred} are ground truth embeddings and predicted embeddings respectively.
- *Act Pred w/ CE*: In this setup, we use Eq. (5) instead of Eq. (7) in the main paper. It uses cross-entropy loss instead of the embedding regression loss.
- *Static Score Weight*: Instead of using different weights w_s across the curriculum stages, a single weight w_s of Stage 1 is used, which is the same across different stages.
- *Static Sampler Stats*: Instead of using different stats μ_s, σ_s across the curriculum stages, a single stat μ_s, σ_s of Stage 1 is used, which is the same across different stages.

We also introduce additional metrics in Fig. 4(a). Let the predicted action sequence be $P = [a_1, a_2, \ldots, a_m]$, the ground-truth sequence be $G = [g_1, g_2, \ldots, g_n]$, and at each step t the model produces a ranked list of candidate actions $R_t = [\, r_{t,1}, r_{t,2}, \ldots \,]$ (e.g., obtained by sorting per-step logits). For step t, define the top-k hit indicator

$$\mathrm{hit}_k(t) \;=\; \mathbf{1}[\, g_t \in \{r_{t,1}, r_{t,2}, \ldots, r_{t,k}\} \,]. \tag{16}$$

Let $\mathcal{D}$ be the evaluation set, and evaluate over the ground-truth length $|G|$. The Action@k is

$$\mathrm{Action@}k \;=\; \frac{\displaystyle\sum_{(x,G)\in\mathcal{D}} \sum_{t=1}^{|G|} \mathrm{hit}_k(t)}{\displaystyle\sum_{(x,G)\in\mathcal{D}} |G|}. \tag{17}$$

We evaluate over the ground-truth length $|G|$. If the policy outputs fewer steps, the missing steps count as misses. If it outputs more, you can ignore extras or stop when the model outputs an explicit `TERMINATE` action.

For the Seq-F1, we use the **Longest Common Subsequence (LCS)** between P and G:

$$m = \mathrm{LCS_len}(P, G), \quad \mathrm{Precision(Pr.)} = \tfrac{m}{|P|}, \quad \mathrm{Recall(Re.)} = \tfrac{m}{|G|}, \quad \mathrm{F1} = \tfrac{2Pr.Re.}{Pr.+Re.}. \tag{18}$$

These metrics consider actions appearing **in the right order**, while tolerating insertions/deletions.

In Figure 4 (c) of the main paper, we consider different aggregations of the reward. This is described below:

$$\boxed{\hat{R}_{i,t} = \hat{R}^{form}_{i,t} \times (\hat{R}^{acc}_{i,t} + \hat{R}^{cons}_{i,t} + \hat{R}^{MMD}_{i,t} + \hat{R}^{var}_{i,t})} \quad \texttt{Default Gated} \tag{19}$$

$$\boxed{\hat{R}_{i,t} = \hat{R}^{form}_{i,t} + \hat{R}^{acc}_{i,t} + \hat{R}^{cons}_{i,t} + \hat{R}^{MMD}_{i,t} + \hat{R}^{var}_{i,t}} \quad \texttt{Arithmetic} \tag{20}$$

$$\boxed{\hat{R}_{i,t} = (\hat{R}^{form}_{i,t} \times \hat{R}^{acc}_{i,t} \times \hat{R}^{cons}_{i,t} \times \hat{R}^{MMD}_{i,t} \times \hat{R}^{var}_{i,t})^{1/5}} \quad \texttt{Geometric} \tag{21}$$

$$\boxed{\hat{R}_{i,t} = 5/(1/\hat{R}^{form}_{i,t} + 1/\hat{R}^{acc}_{i,t} + 1/\hat{R}^{cons}_{i,t} + 1/\hat{R}^{MMD}_{i,t} + 1/\hat{R}^{var}_{i,t})} \quad \texttt{Harmonic} \tag{22}$$

It is important to note that handling degenerate cases of a reward $R = 0$ is done through the max operation such that $\hat{R} = \max(R, \epsilon_r)$. In this case, $\epsilon_r = 0.01$.

D Additional Experiments

In Fig. 7, we consider different ablations over the NAVIGATE baseline. We proceed to describe the different ablations as follows:

- *w/o no. tool calls:* This is the case when the variable $\hat{m}$ is removed from the normalized features in computing the difficulty score of Eq. (2) in the main paper.
- *w/o no. distinct tools:* This is the case when the variable $\hat{u}$ is removed from the normalized features in computing the difficulty score of Eq. (2) in the main paper.
- *w/o no. of turns in trace:* This is the case when the variable $\hat{L}$ is removed from the normalized features in computing the difficulty score of Eq. (2) in the main paper.
- *w/o reasoning hardness:* This is the case when the variable $\hat{R}$ is removed from the normalized features in computing the difficulty score of Eq. (2) in the main paper.
- *w/o knowledge reliance:* This is the case when the variable $\hat{K}$ is removed from the normalized features in computing the difficulty score of Eq. (2) in the main paper.

- *w/o likelihood hardness:* This is the case when the variable $\hat{l}$ is removed from the normalized features in computing the difficulty score of Eq. (2) in the main paper.
- *w/ Action Head Type 1:* This is the case when we use a two-layer action-prediction head. In this case, the size of the middle layer is the arithmetic mean of hidden dimension and the action embedding dimension different from that defined in Eq. (4) of the main paper.
- *w/ Action Head Type 2:* This is the case when we use a three-layer action-prediction head. In this case, the size of the first middle layer is at one-third the difference between the hidden layer and action embedding dimension. The size of the second middle layer is at two-third the difference between the hidden layer and action embedding dimension. This is different from that defined in Eq. (4) of the main paper.
- *w/ Action Prediction Loss Wt. 1e-3:* In this case, we use a weight of $1e-3$ for the action prediction loss defined in Eq. (7) of the main paper different from the default weight of $1e-4$.
- *w/ Action Prediction Loss Wt. 1e-2:* In this case, we use a weight of $1e-2$ for the action prediction loss defined in Eq. (7) of the main paper different from the default weight of $1e-4$.
- *w/ Sampler Type 1:* We consider the following alternative sampler for the curriculum learning

$$p_t(x) \propto \tilde{p}(x)\, \exp\left(-\frac{|D_s(x)-\mu_s|}{b_s}\right). \tag{23}$$

  The hyper-parameters for this sampler for the different stages are as follows:
  - Stage 1: $w_s = [0.305, 0.169, 0.136, 0.136, 0.085, 0.169]$ $\mu_s = 0.20, b_s = 0.170$,
  - Stage 2: $w_s = [0.250, 0.188, 0.156, 0.250, 0.063, 0.094]$ $\mu_s = 0.40, b_s = 0.255$,
  - Stage 3: $w_s = [0.221, 0.221, 0.176, 0.265, 0.029, 0.088]$ $\mu_s = 0.60, b_s = 0.340$,
  - Stage 4: $w_s = [0.200, 0.200, 0.200, 0.257, 0.057, 0.086]$ $\mu_s = 0.75, b_s = 0.425$
- *w/ Sampler Type 2:* We consider the following alternative sampler for the curriculum learning.

$$p_t(x) \propto \tilde{p}(x)\, \frac{1}{1+\left(\frac{D_s(x)-\mu_s}{\gamma_s}\right)^2}. \tag{24}$$

  The hyper-parameters for this sampler for the different stages are as follows:
  - Stage 1: $w_s = [0.305, 0.169, 0.136, 0.136, 0.085, 0.169]$ $\mu_s = 0.2, \gamma_s = 0.118$,
  - Stage 2: $w_s = [0.250, 0.188, 0.156, 0.250, 0.063, 0.094]$ $\mu_s = 0.4, \gamma_s = 0.177$,
  - Stage 3: $w_s = [0.221, 0.221, 0.176, 0.265, 0.029, 0.088]$ $\mu_s = 0.6, \gamma_s = 0.235$,
  - Stage 4: $w_s = [0.200, 0.200, 0.200, 0.257, 0.057, 0.086]$ $\mu_s = 0.75, \gamma_s = 0.294$
- *w/ Covariance Discrepancy Type 1:* The log-Euclidean discrepancy measures the distance between covariance matrices in the log-SPD space, where taking the matrix logarithm maps covariances to a flat, Euclidean-like geometry. This makes the metric more stable and sensitive to meaningful structural changes, especially when covariance spectra vary widely. Normalizing by the log-norms ensures the score remains within 0 and 1, preserving comparability across training steps.

$$\Omega_{\mathrm{LE}}(\Sigma_{i,t}, \Sigma_{i,t-1}) = \frac{\|\log(\Sigma_{i,t}) - \log(\Sigma_{i,t-1})\|_F}{\|\log(\Sigma_{i,t})\|_F + \|\log(\Sigma_{i,t-1})\|_F}, \qquad \Omega_{\mathrm{LE}} \in [0,1]. \tag{25}$$

- *w/ Covariance Discrepancy Type 2:* The affine-invariant Riemannian discrepancy computes the geodesic distance between two covariance matrices on the SPD manifold, providing a principled, transformation-invariant comparison. By evaluating the log-map of the congruence transform $\Sigma_{i,t}^{-\frac{1}{2}}\, \Sigma_{i,t-1}\, \Sigma_{i,t}^{-\frac{1}{2}}$, it captures intrinsic geometric differences independent of scaling or affine changes. The normalized form keeps the resulting score within $[0,1]$, making it suitable for curriculum weighting or adaptive sampling.

$$\Omega_{\mathrm{AIRM}}(\Sigma_{i,t}, \Sigma_{i,t-1}) = \frac{\left\|\log\left(\Sigma_{i,t}^{-\frac{1}{2}} \Sigma_{i,t-1} \Sigma_{i,t}^{-\frac{1}{2}}\right)\right\|_F}{1+\left\|\log\left(\Sigma_{i,t}^{-\frac{1}{2}} \Sigma_{i,t-1} \Sigma_{i,t}^{-\frac{1}{2}}\right)\right\|_F}, \qquad \Omega_{\mathrm{AIRM}} \in [0,1]. \tag{26}$$

- *w/ 2x Default RBF Kernel Bandwidth:* In this case, we consider twice the default bandwidth used in the RBF kernel of MMD in Eq. (10) of the main paper.
- *w/ 0.5x Default RBF Kernel Bandwidth:* In this case, we consider half the default bandwidth used in the RBF kernel of MMD in Eq. (10) of the main paper.

It is to be noted that for both Covariance Discrepancy Type 1 and Type 2, the covariance matrix is conditioned such that $\Sigma \leftarrow \Sigma + \epsilon_s I$. Here $\epsilon_s = 0.01$. From the results, we see that all ablations produce drops in performance compared to that of the baseline. Interestingly, using a larger action prediction head produces drop in performance. This is because it perturbs the logit geometry and the optimization dynamics rather than adding useful capacity. A wider/deeper (often untied) head typically inflates logit norms, lowering the effective temperature so the softmax becomes over-confident; this saturates cross-entropy gradients, starves the backbone of corrective signal, and worsens calibration and generalization.

Furthermore, newer covariance-discrepancy objectives although more principled, can under-perform a simple baseline because they amplify estimation error and impose wrong bias for your data regime. Full or geometry-aware covariance distances are high-variance when batch sizes are modest and features are high-dimensional; their gradients depend on poorly estimated second-order statistics, leading to unstable updates.

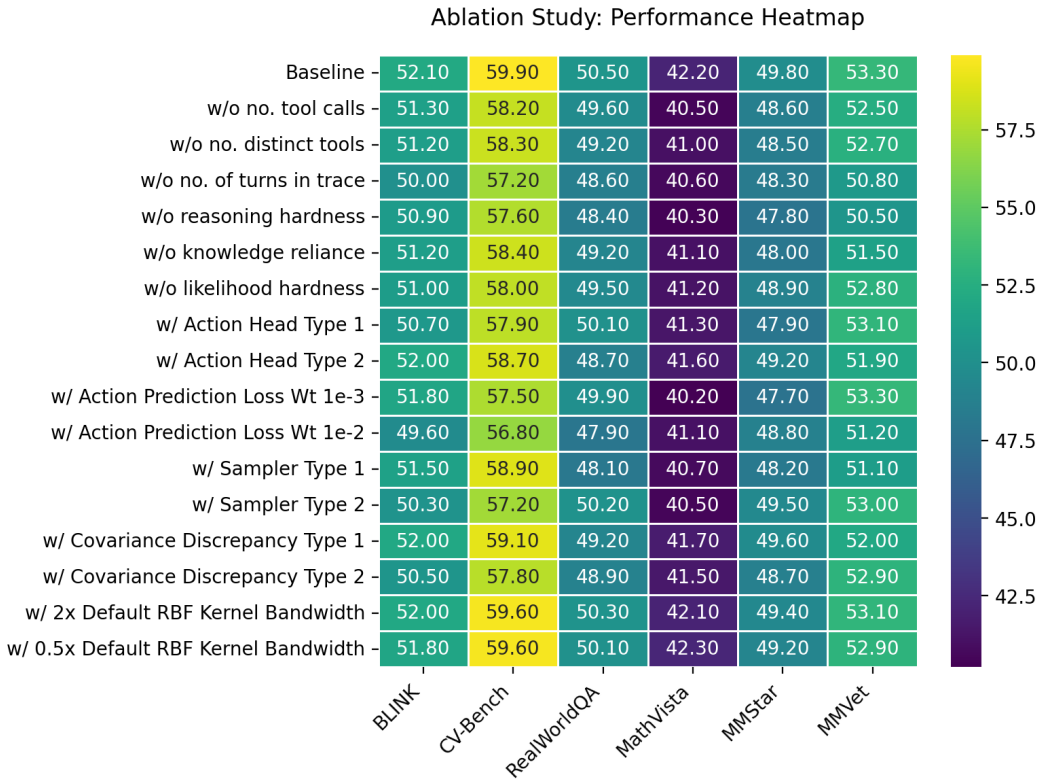


Figure 7: Accuracy variation for different ablations of NAVIGATE.

We also study several other design choices and understand the effect of those design choices in the performance of our model. These include:

- *Design Choice 1:* In this case, we consider the variation in the action planner horizon from the default value of $T = 50$. We consider other choices of $T = 16, 32, 64$ as $1a$, $1b$ and $1c$.
- *Design Choice 2:* In this case, we consider the variation in the embedding model for action regression. The default is the e5 model (Wang et al., 2022) while in this design choice we consider the BGE-base (Xiao et al., 2023) embedding model.

- *Design Choice 3:* In this case, we consider an alternative fine-tuning of the planner head and the next token prediction head. The default setup considers a joint training of the planner head and the next token prediction head.
- *Design Choice 4:* The base distribution $\tilde{p}(x)$ is different from the uniform distribution. In this case, we consider $\tilde{p}(x)$ as inversely proportional to frequency over tools.
- *Design Choice 5:* In this case, we consider multi-token prediction as an alternative to the action prediction head. We consider variations of predicting 2, 3, 4, 5 tokens ahead and call them $5a$, $5b$, $5c$, $5d$, respectively.

The results are shown in Table 3. All the proposed design choices under-perform compared to the default setup. The most interesting result is about the multi-token prediction. Using multi-token prediction as an alternative to action prediction produces a large drop in performance compared to the base NAVIGATE framework. However, as more future tokens are predicted, the performance tends to slightly increase.

Table 3: Accuracy Results of different design choices when applied on our proposed NAVIGATE framework.

Model	Perception	Perception + Reasoning
NAVIGATE	54.2	48.4
NAVIGATE w/ 1a	53.2	47.1
NAVIGATE w/ 1b	53.0	47.5
NAVIGATE w/ 1c	53.1	48.2
NAVIGATE w/ 2	54.0	48.3
NAVIGATE w/ 3	53.6	48.3
NAVIGATE w/ 4	54.1	47.4
NAVIGATE w/ 5a	50.0	45.4
NAVIGATE w/ 5b	50.3	46.0
NAVIGATE w/ 5c	50.8	46.5
NAVIGATE w/ 5d	50.9	47.8

Table 4 compares NAVIGATE trained with different curriculum granularities across three Perception tasks (BLINK, CV-Bench, RealWorldQA) and three Perception+Reasoning tasks (MathVista, MM-Star, MMVet). The default four-stage curriculum yields the highest averages: Perception Avg 54.2 (vs. 53.6 with 2-stage and 53.7 with 3-stage), Perception+Reasoning Avg 48.4 (vs. 47.6 and 47.9), and the Overall Avg 51.3 (vs. 50.6 and 50.8). Improvements are small but consistent across most benchmarks—for example, MMVet 53.3 (four-stage) vs. 52.3/52.8, MathVista 42.2 vs. 41.7/41.8, CV-Bench 59.9 vs. 59.0/59.3, and RealWorldQA 50.5 vs. 50.0/49.9. The four-step schedule likely helps because its finer-grained pacing narrows difficulty jumps between phases, allowing the model to stabilize optimization and consolidate perception skills before progressively integrating multi-step reasoning. In contrast, compressing the curriculum into two or three stages widens the distribution gap between successive phases, underexposes the model to mid-difficulty examples, and can cause optimization shocks (e.g., noisier gradients or partial forgetting), which collectively translate into the observed, slightly lower averages.

Table 4: Effect of using different sizes of curriculum. The default is using a four stage curriculum.

Method	**Perception**				**Perception + Reasoning**				**Overall**
	BLINK	CV-Bench	RealWorldQA	Avg	MathVista	MMStar	MMVet	Avg	Avg
NAVIGATE	52.1	59.9	50.5	54.2	42.2	49.8	53.3	48.4	51.3
NAVIGATE w/ 2 stage curriculum	51.8	59.0	50.0	53.6	41.7	49.0	52.3	47.6	50.6
NAVIGATE w/ 3 stage curriculum	52.0	59.3	49.9	53.7	41.8	49.2	52.8	47.9	50.8

Figures 8 (a) and (b) illustrate how overall accuracy improves as the fine-tuning percentage increases across four RLVR methods—GRPO, DAPO (Yu et al., 2025b), GSPO (Zheng et al., 2025a), and ARPO (Dong et al., 2025)—on both MMVet and RealWorldQA. In both datasets, accuracy consistently rises with more fine-tuning, but the methods differ in effectiveness: ARPO shows the strongest performance throughout, achieving the highest accuracy at every fine-tuning level, while GRPO

yields the lowest accuracy curves. DAPO and GSPO provide intermediate performance, with GSPO generally outperforming DAPO as fine-tuning progresses. Although all methods converge more closely near 100% fine-tuning, ARPO maintains a clear margin on both benchmarks, demonstrating its robustness across diverse multimodal reasoning settings.

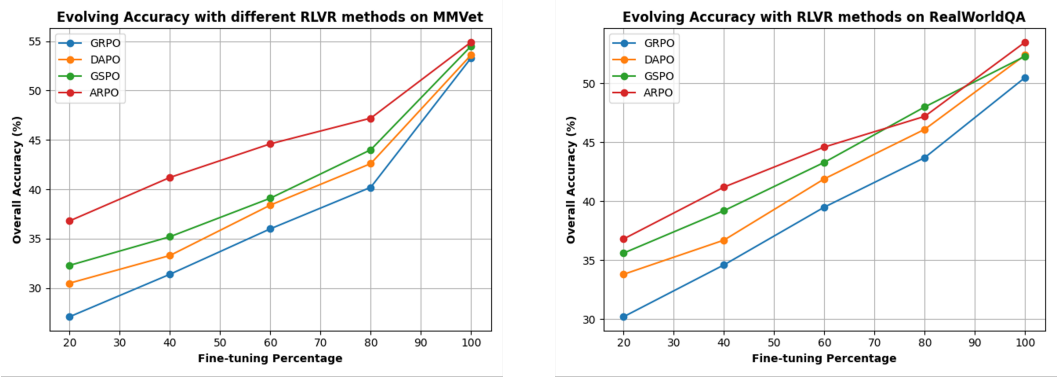


Figure 8: Accuracy variation with training percentage for different RL variants when applied to NAVIGATE.

Figure 9 shows how the NAVIGATE framework's reward evolves during training when paired with four different RLVR methods—ARPO, GRPO, GSPO, and DAPO—and demonstrates that all variants exhibit a steady upward trend as training progresses. Among them, NAVIGATE with ARPO consistently achieves the highest reward throughout most of training, followed closely by GSPO, while GRPO and DAPO trail with lower reward curves. Although all methods converge towards similar values near full training completion, the margin between ARPO and the other approaches remains visible across much of the trajectory, indicating that ARPO provides stronger and more stable reward optimization when integrated with NAVIGATE.



Figure 9: Evolution of rewards for different RL algorithms on LATTE Trace 98K .

Figure 10 (a) and (b) illustrate how progressively adding more stages of the curriculum affects NAVIGATE's accuracy on (a) perception-only tasks and (b) combined perception and reasoning tasks. In both settings, performance improves monotonically as additional curriculum stages are

included: training solely on Stage 1 yields the lowest accuracy, while incorporating Stage 2 provides a noticeable boost. Adding Stage 3 further enhances results, and the full four-stage curriculum achieves the highest accuracy overall. Although absolute accuracy values differ slightly between the two benchmarks, the trend is consistent—finer curriculum granularity leads to substantially better model performance, underscoring the importance of staged learning for both perception and multimodal reasoning.

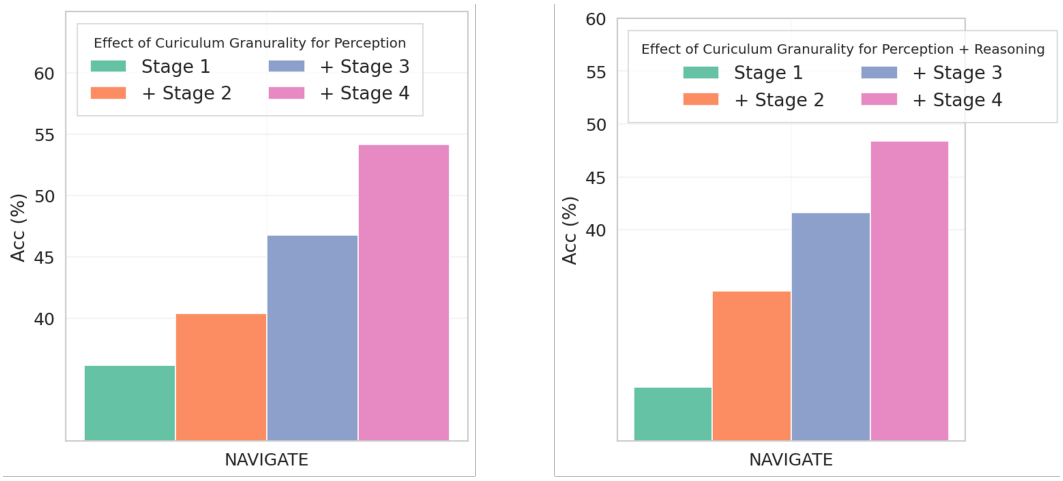


Figure 10: Effect of training only on part of the curriculum for (a) perception and (b) perception + reasoning datasets.

Figure 11 (a) presents two complementary visualizations examining how different ablations and methods influence action-space behavior in multimodal reasoning. In panel (a), the heatmap reports the standard deviation of action predictions across multiple steps in the decoding procedure for various NAVIGATE ablations, showing that removing stages generally increases variability, whereas the full NAVIGATE model exhibits more stable and consistent action predictions across tasks.

To understand how the model's behavior evolves, we examine the distribution of its predicted actions at various stages of training. For each query, we draw ten trajectories from different checkpoints. These trajectories are embedded using the e5 encoder (Wang et al., 2022) and mapped into a low-dimensional space using t-SNE (Van der Maaten & Hinton, 2008), as illustrated in panel (b). Results reveal that NAVIGATE (full) produces a more structured and moderately dispersed distribution, while ablations such as removing Stage I or II lead to noticeably broader and noisier exploration patterns. Methods like AdaReasoner and LATTE show distinct but comparatively less balanced distributions. Together, the visualizations highlight that NAVIGATE's staged curriculum contributes to both more stable action trajectories, more controlled action-space exploration during reasoning, reflecting a broader search over the action space.

The Table 5 summarizes computational overhead of training Qwen2.5-VL-3B on LATTE-Trace-98K dataset using 8 A100 GPUs, highlighting clear increase in both training time and peak RAM usage across three stages of the pipeline. Stage I is lightweight since it is just curriculum generation, completing in 22.3 minutes with a modest 14.2 GB RAM. Stage II introduces substantially greater complexity, raising training time to 95.6 minutes and memory usage to 72.4 GB. Stage III is the most computationally demanding since it is Reinforcement Learning with Verifiable Rewards, requiring 125.4 minutes of training and peaking at 78.9 GB of RAM. Overall, the results illustrate a progressively heavier compute load as the training procedure advances through later stages.

Table 6 reports average inference overhead of Qwen2.5-VL-3B across three reasoning frameworks on MMVet using a A100 GPU, showing that NAVIGATE is the fastest at 0.22 minutes per turn with the shortest thinking trace of 10.1 tokens. LATTE incurs a slightly higher per-turn time of 0.25 minutes and produces a moderately longer trace of 14.2 tokens, while AdaReasoner is the most expensive, requiring 0.30 minutes per turn and generating the longest thinking trace at 16.5 tokens. Overall, the

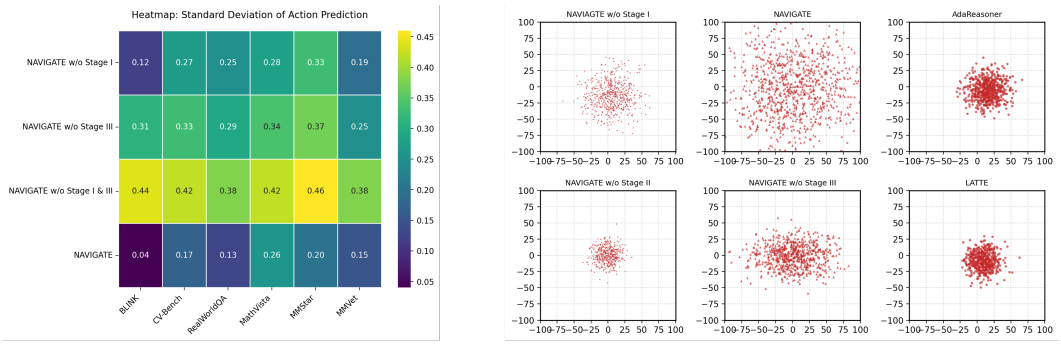


Figure 11: (a) Visualization showing how the action prediction varies across the multi-step reasoning trace for different ablations of NAVIGATE. (b) Distribution of action space exploration for different methods.

Table 5: Computational overhead of using Qwen2.5-VL-3B when trained with LATTE-Trace-98K (Ma et al., 2025a). The training is done using 8 A100 GPUs.

Model	Training Time (mins)	Peak RAM (GB)
Stage I	22.3	14.2
Stage II	95.6	72.4
Stage III	125.4	78.9

results indicate that NAVIGATE achieves both the lowest latency and most concise reasoning traces among the compared methods.

Table 6: Average computational overhead during inference when using Qwen2.5-VL-3B across different frameworks. The measurement is carried out on A100 GPUs on MMVet.

Model	Inference time per turn (mins)	Thinking trace length (tokens)
LATTE	0.25	14.2
AdaReasoner	0.30	16.5
NAVIGATE	0.22	10.1

In Table 7, we also measure length stability of the thinking trace. For the length stability, we compute the standard deviation of the length of the thinking trace across different turns of a data sample. This is then averaged over the whole dataset. From the results, we see that LATTE and AdaReasoner tend to produce more unstable thinking trace lengths. On the other hand, the variation in length is much less for our proposed NAVIGATE framework. However, removing the covariance discrepancy reward $\hat{R}_{i,t}^{var}$, increases the standard deviation to 0.25, which suggests that the covariance discrepancy reward is important for maintaining length stability.

To isolate this factor, we add a matched control initialized from the same backbone and trained using exactly the same LATTE-Trace-98K samples as shown in Table 8.

- **LATTE SFT**: the existing supervised trace-imitation baseline;
- **LATTE SFT $\rightarrow$ vanilla GRPO**: the same SFT initialization, rollout budget, optimizer, training steps, and format/accuracy rewards as NAVIGATE, but without the adaptive curriculum, planner head, consistency reward, MMD reward, or variance reward; and
- **NAVIGATE**: the complete method.

The result suggests two more precise conclusions. As shown, vanilla GRPO improves over LATTE SFT, meaning that part of the original gain must be attributed to interactive end-task optimization.

Table 7: Measuring thinking length stability during inference when using Qwen2.5-VL-3B across different frameworks. The measurement is carried out on A100 GPUs on MMVet.

Model	Average Standard Deviation Thinking Trace length (tokens)	Mean Thinking Trace length (tokens)
LATTE	0.46	14.2
AdaReasoner	0.41	16.5
NAVIGATE	0.09	10.1
NAVIGATE w/o $\hat{R}_{i,t}^{var}$	0.25	12.2

Table 8: Matched control on LATTE-Trace-98K.

Method	P	P+R	Overall
LATTE SFT	47.5	43.0	45.2
LATTE SFT $\rightarrow$ vanilla GRPO	51.8	46.0	48.9
NAVIGATE w/o Stage I	53.1	47.4	50.2
NAVIGATE w/o Stage II	53.4	47.6	50.5
NAVIGATE w/o Stage III	52.8	47.0	49.9
NAVIGATE	**54.2**	**48.4**	**51.3**

Second, the residual difference of approximately 2.4 overall points between matched vanilla GRPO and NAVIGATE would isolate the combined effect of the adaptive curriculum, semantic planner, and flexible reward design.

We also show results in Table 9, where the training source data is varied from 98K to 293K. As expected from the results, increasing the scale of the dataset improves the quantitative performance for perception tasks but decreases for perception + reasoning tasks.

We then add consistency, MMD, and variance rewards individually as shown in Table 10 and obtain improved performance compared to the baseline.

We add an ablation in Table 11 over planning horizon $T \in \{1, 3, 5, 16, 32, 50, 64\}$, where $T = 1$ represents next-action prediction without multi-step look-ahead and $T = 50$ is the default configuration. All variants use the same backbone, training data, reward construction, rollout budget, and optimization steps.

The results suggest that the main benefit is obtained when moving from one-step prediction to a short multi-action plan. Increasing T beyond the typical executed trajectory length provides smaller gains and eventually saturates. This is expected because T is a supervision horizon rather than a requirement that the model execute exactly T tool calls.

We propose a held-out-tool evaluation in which one tool family is excluded from training and introduced only at inference through its name, natural-language description, argument schema, and frozen execution endpoint. The overall performance combining perception and perception+reasoning are shown in Table 12

Here, oracle-tool accuracy supplies the correct tool output to the model and therefore separates tool-routing failure from downstream reasoning failure. The results indicate that providing only the tool name yields limited selection and task accuracy, whereas adding a natural-language description and argument schema substantially improves both metrics. The remaining gap relative to the known-tool setting suggests that semantic action embeddings facilitate limited transfer, but do not fully eliminate the need for prior exposure to the tool interface. The comparatively stable oracle-tool accuracy further indicates that most of the transfer gap arises from selecting and invoking the unseen tool, rather than from reasoning over the resulting tool output.

We also show additional visualization in Fig. 12, 13, 14, 15, 16 and 17. In most cases, our proposed framework invokes the correct tools and produces more correct answers than LATTE (Ma et al., 2025a) and AdaReasoner (Song et al., 2026).

Table 9: Trace-source transfer experiment.

Trace source	Training method	P	P+R	Overall
LATTE-Trace-293K	Vanilla SFT	52.5	41.5	47.0
LATTE-Trace-293K	Vanilla SFT → GRPO	54.2	43.2	48.7
LATTE-Trace-293K	NAVIGATE	**56.1**	**47.2**	**51.6**
LATTE-Trace-98K	Vanilla SFT → GRPO	51.8	46.0	48.9
LATTE-Trace-98K	NAVIGATE	**54.2**	**48.4**	**51.3**

Table 10: Component-wise reward ablation over a common format-plus-accuracy baseline.

Reward configuration	Perception	Perception+Reasoning	Overall
Format + accuracy only	52.8	47.0	49.9
Format + accuracy + consistency	53.4	47.7	50.6
Format + accuracy + MMD	53.5	47.8	50.7
Format + accuracy + variance	53.2	47.5	50.4
Full NAVIGATE	**54.2**	**48.4**	**51.3**

Table 11: Effect of the planner horizon. P and P+R denote average accuracy on perception and perception-plus-reasoning benchmarks. Executed calls denotes the average number of tool calls per example.

Horizon T	P	P+R	Overall	Executed calls
1	52.9	47.1	50.0	2.42
3	53.4	47.7	50.6	2.68
5	53.8	48.0	50.9	2.79
16	53.2	47.1	50.2	2.81
32	53.0	47.5	50.3	2.84
50	**54.2**	**48.4**	**51.3**	**2.86**
64	53.1	48.2	50.7	2.87

Table 12: Held-out-tool transfer evaluation.

Held-out tool setting	Tool selection	Task accuracy	Oracle-tool accuracy
New OCR tool, name only	18.6	31.4	50.8
New OCR tool, name + description	43.2	41.7	51.0
New OCR tool, description + schema	**56.8**	**46.3**	51.1
Known OCR tool	71.4	50.5	51.3
New calculator API, name only	24.1	34.8	49.9
New calculator API, name + description	48.7	43.6	50.2
New calculator API, description + schema	**61.2**	**47.1**	50.4
Known calculator API	74.5	49.8	50.6

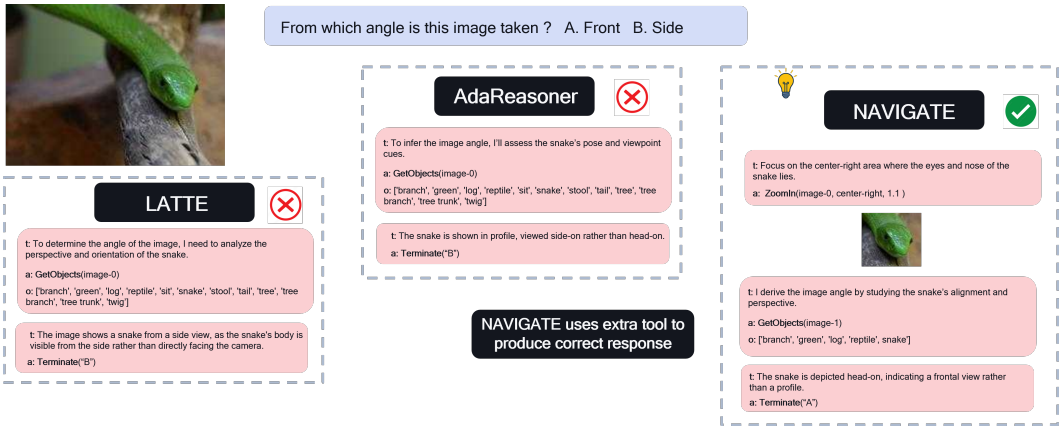


Figure 12: We demonstrate how our proposed framework can improve over existing works.

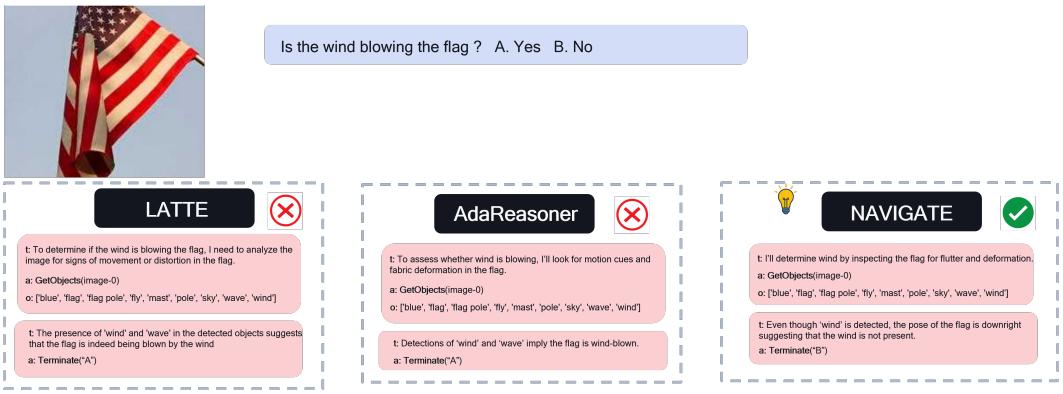


Figure 13: We demonstrate how our proposed framework can improve over existing works.

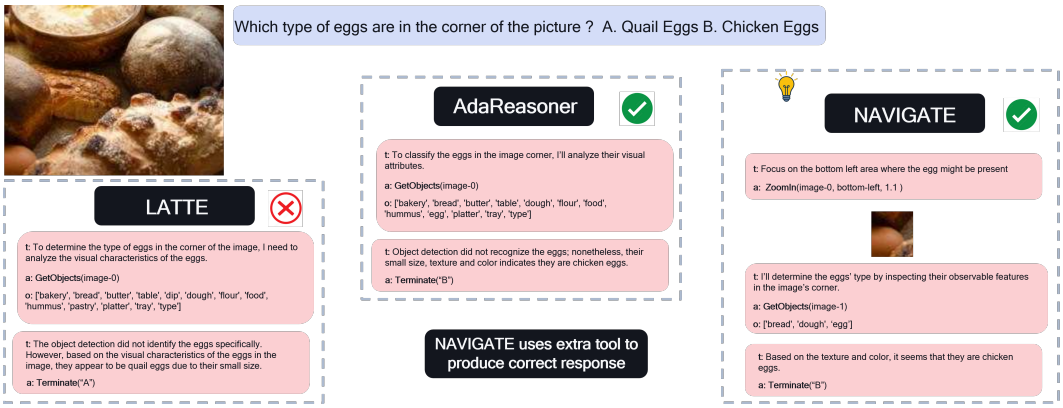


Figure 14: We demonstrate how our proposed framework can improve over existing works.

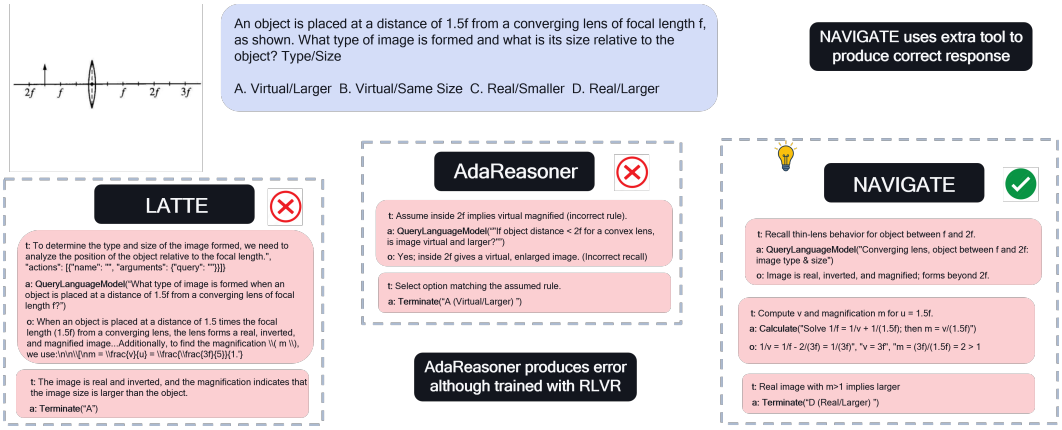


Figure 15: We demonstrate how our proposed framework can improve over existing works.

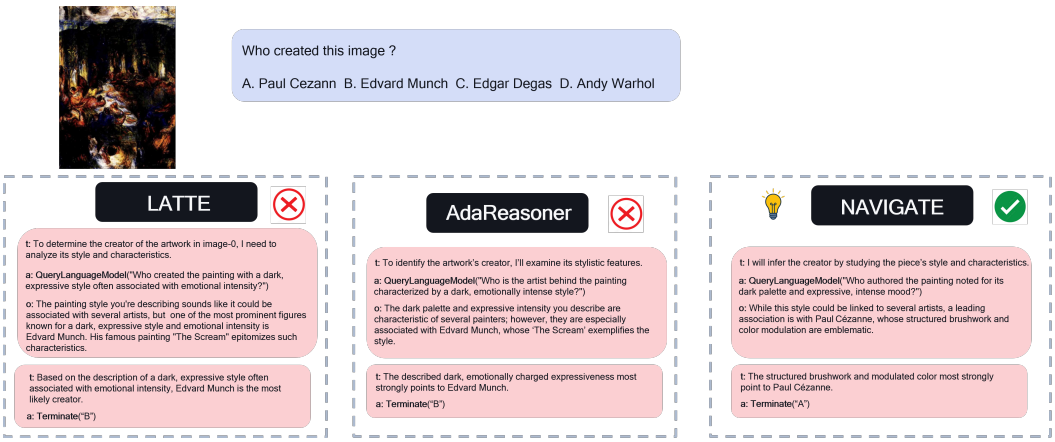


Figure 16: We demonstrate how our proposed framework can improve over existing works.

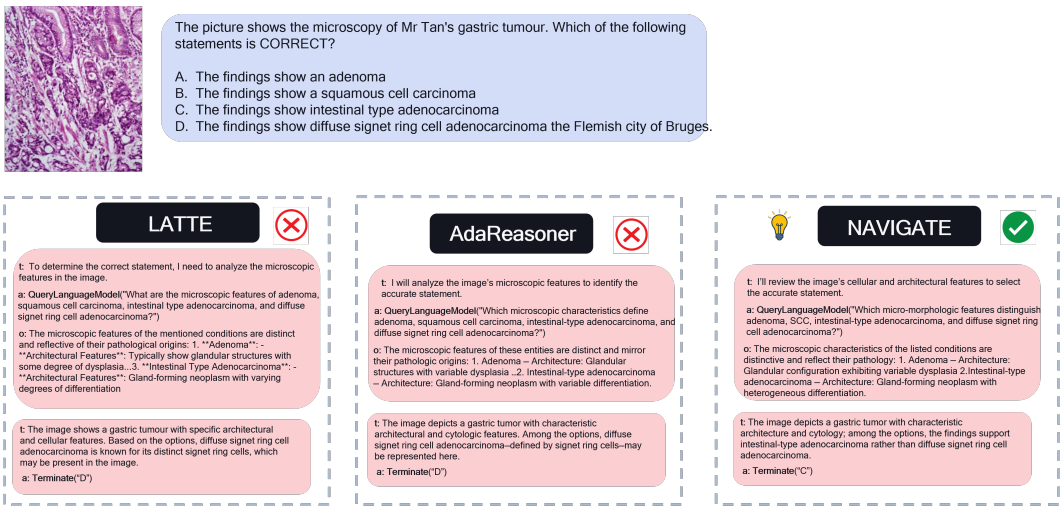


Figure 17: We demonstrate how our proposed framework can improve over existing works.