# PIVOT: PIVOT-AWARE ON POLICY SELF DISTILLATION FOR MULTI-TURN VLM AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) via Group-Relative Policy Optimization (GRPO) is widely used for multi-turn VLM agent training, yet it suffers from zero-gradient silence on uniform failures and coarse episode-level credit assignment. While On-Policy Distillation (OPD) and On-Policy Self-Distillation (OPSD) mitigates sparse rewards using hindsight information, its underlying mechanisms remain poorly understood. Through controlled counterfactual rollback probes across five multi-turn VLM agent benchmarks, we reveal that performance gains in OPSD/OPD are largely driven by physical state rollback at the *pivot step*, defined as the first unrecoverable action without remaining step budget. However, physical state rollbacks are computationally prohibitive and infeasible in real-world environments. To bridge this gap, we present **P**ivot-Aware **I**nternalized **V**isual **O**n-Policy **T**raining (PIVOT), an RL framework that internalizes pivot localization and state restoration directly into token-level parameter updates, eliminating environment rollbacks during RL training and additional skill hints at test time. PIVOT unifies three functional roles within a single architecture: a failure Analyzer non-invasively localizes the pivot step and diagnoses failure modes from visual trajectory collages and action logs; a detached Teacher re-scores failed tokens under this privileged diagnostic context; and a Student optimizes joint GRPO and confidence-gated OPD objectives. At test time, both Teacher and Analyzer branches are stripped. Evaluated on five multi-turn VLM agent tasks across cognitive grid puzzles, 3D embodied control and navigation, and generative reasoning, PIVOT achieves $0.90$ overall accuracy on Qwen2.5-VL-3B ($+8\%$ over SFT+GRPO baseline and $+5\%$ over previous SOTA) and scales to $0.92$ on Qwen3-VL-2B ($+12\%$ over SFT+GRPO baseline).

## 1 INTRODUCTION

Reinforcement learning (RL) has become a dominant post-training paradigm for vision-language model (VLM) agents (Ouyang et al., 2022; Guo et al., 2025; Yao et al., 2023; Wang et al., 2026b; Shen et al., 2025; Li et al., 2026). The prevailing RLVR framework, Group-Relative Policy Optimization (GRPO) (Shao et al., 2024), samples multiple rollouts per prompt and computes relative advantages by normalizing episode-level returns within each group. However, as analyzed in Sec. 2.1 (Fig. 1(a)) across five multi-turn VLM agent benchmarks, GRPO suffers from two critical failure modes: *zero-gradient silence*, where the policy gradient vanishes as the group returns are identical (Yu et al., 2026), and *coarse episode-level credit assignment*, which ranks entire trajectories by scalar returns without pinpointing which step first renders the task unrecoverable (Lightman et al., 2024; Uesato et al., 2022).

On-Policy Distillation (OPD) (Agarwal et al., 2024; Lu et al., 2026) mitigates sparse rewards via token-level supervision, but fundamentally requires access to white-box teacher logits and matched vocabularies. To bypass these constraints, On-Policy Self-Distillation (OPSD) (Wang et al., 2026a; Zhao et al., 2026; Andrychowicz et al., 2017; Shinn et al., 2023) performs self-supervision by conditioning policy updates on hindsight information. However, the mechanisms driving OPSD/OPD gains in multi-turn VLM agents remain poorly understood. To figure out these underlying mechanisms, we conduct counterfactual probes by systematically rolling back the environment to the *pivot step*, defined as the first unrecoverable action within the remaining step budget, and then resampling trajectory suffixes to evaluate the performance gains.



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.

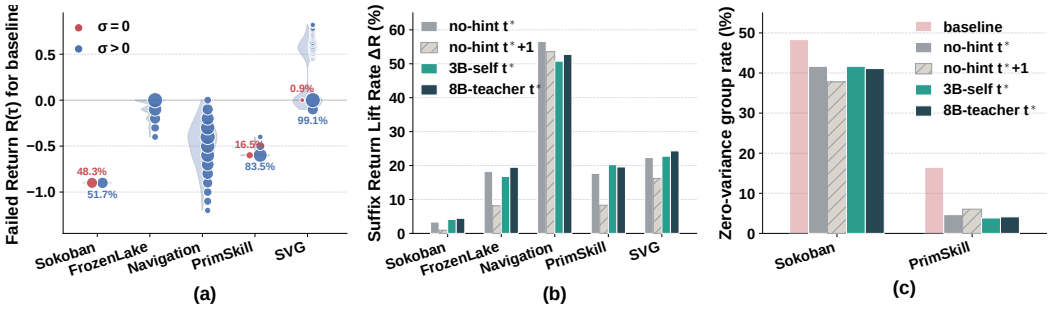



Figure 1: **(a)** Base policy failed-rollout returns $R(\tau)$, distinguishing zero-variance ($\sigma{=}0$, red) from variance-bearing ($\sigma{>}0$, blue) groups (marker size scales with trajectory frequency; percentages denote $\sigma{=}0$ share). **(b)** Suffix return lift $\Delta R$ (%) across four rollback probes at the pivot step: no-hint resets at $t^*$ and $t^*{+}1$, alongside 3B-self and 8B-teacher skill hints at $t^*$. **(c)** Zero-variance group rate (%) across four rollback probes on Sokoban and PrimitiveSkill.

Empirically, as shown in Sec. 2.2 and Figs. 1(b)–(c), our probe reveals three key findings: **(i)** Restoring the environment state to the pivot step without any skill hints provides the primary share of suffix return lift and substantially reduces zero-advantage GRPO groups; **(ii)** Delaying this state reset by even a single step ($t^*{+}1$) causes suffix return gains to sharply diminish; and **(iii)** Incorporating explicit skill hints, whether self-generated or teacher-provided, yields only marginal incremental gains over hint-free state restoration (Fig. 1(b)). These observations yield a vital insight: ***the efficacy of OPD/OPSD in multi-turn VLM agents is largely driven by precise state rollbacks at the pivot step, rather than by additional skill prompting***.

However, state rollback is computationally prohibitive and infeasible in real-world, non-rewindable environments, while inference cannot tolerate additional skill prompt overhead. Therefore, a practical RL framework for multi-turn VLM agents must: **(i)** non-invasively and precisely localize the pivot step directly from trajectory histories, and **(ii)** convert this localized pivot step into fine-grained, token-level policy gradients wihout environment rollbacks during RL training and additional skill hints at test time.

To this end, we present **P**ivot-Aware **I**nternalized **V**isual **O**n-Policy **T**raining (PIVOT), an on-policy distillation framework that internalizes pivot step localization and failure diagnostic hints directly into parameter updates without simulator rollbacks (Fig. 2, Sec. 3). In Stage I, a lightweight SFT phase cold-starts a **failure Analyzer** to non-invasively predict the pivot step, failure mode, and optional guidance skills directly from visual trajectory collages and action logs (Sec. 3.1). In Stage II, a **three-role unified architecture** orchestrates training by consolidating Student, Analyzer, and Teacher roles within shared parameters: the Analyzer extracts a localized visual panel around the pivot step, combining it with the predicted failure mode and guidance skills to construct a privileged context. A detached Teacher then re-scores original failed tokens under this privileged context, while the Student optimizes joint GRPO and Gated OPD objectives (Sec. 3.2). At test time, both Analyzer and Teacher are stripped away, deploying the agent strictly as an unprivileged Student with zero computational and skill prompt overhead. Our main contributions are summarized as follows:

- **Mechanistic Insight into OPSD:** We characterize GRPO failure modes in multi-turn VLM agents and empirically analyze via counterfactual rollback probes that the OPSD/OPD performance gains stem mainly from state rollback at the critical pivot step, rather than skill hints as additional prompts.
- **Non-Invasive Pivot Step Localization:** We propose PIVOT, introducing a trajectory Analyzer that non-invasively localizes pivot steps and provide failure diagnoses directly from complete visual collages and action logs.
- **Unified Three-Role Architecture:** We formulate a single-model paradigm consolidating Student, Analyzer, and Teacher roles, internalizing state rollback during training while preserving zero computational and skill prompt overhead at inference.
- **Superior Performance:** Evaluated on five multi-turn VLM agent benchmarks across cognitive grid puzzles, 3D embodied control and navigation, and generative reasoning, PIVOT achieves 0.90 overall accuracy on Qwen2.5-VL-3B (Bai et al., 2025b) (+8% over the

SFT+GRPO baseline and $+5\%$ over dense-reward SOTA) and scales to $0.92$ on Qwen3-VL-2B (Bai et al., 2025a) ($+12\%$ over the SFT+GRPO baseline).

# 2 Mechanistic Analysis: Physical Restoration Dominates Hindsight Guidance

## 2.1 Limitations of Episode-Level Credit Assignment

**Preliminaries.** We formulate a multi-turn vision-language agent task as a partially observable Markov decision process (POMDP) (Åström, Karl Johan, 1965) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{O}$ the observation space, $\mathcal{A}$ the action space, $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ the transition dynamics, $\mathcal{R}(s_t, a_t)$ the reward function, $\Omega(o_t \mid s_t)$ the observation emission model, and $\gamma \in (0, 1]$ the discount factor. At turn $t$, the agent receives observation $o_t \sim \Omega(\cdot \mid s_t)$, maintains execution history $h_t = (o_0, a_0, \ldots, o_t)$, and samples action $a_t \sim \pi_\theta(\cdot \mid h_t)$. Execution yields a rollout trajectory $\tau = (o_0, a_0, r_0, \ldots, o_{T-1}, a_{T-1}, r_{T-1})$ of length $T$ with cumulative return $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$. Given prompt $q \sim \mathcal{Q}$, post-training RL optimizes expected trajectory return $J(\theta) = \mathbb{E}_{q,\tau}[R(\tau)]$.

The prevailing estimator for RLVR, Group-Relative Policy Optimization (GRPO) (Shao et al., 2024), samples a group of $N$ rollouts $\{\tau^{(n)}\}_{n=1}^{N}$ per prompt and computes standardized group advantages:

$$A_n^{\mathrm{rl}} = \frac{R(\tau^{(n)}) - \mu}{\sigma + \epsilon}, \quad \mu = \frac{1}{N}\sum_{n=1}^{N} R(\tau^{(n)}), \quad \sigma = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\big(R(\tau^{(n)}) - \mu\big)^2}, \tag{1}$$

where $\epsilon > 0$ is a numerical stabilizer. GRPO assigns this scalar $A_n^{\mathrm{rl}}$ uniformly to *every* action token in $\tau^{(n)}$. Consequently, it evaluates trajectories solely by episode-level return, ignoring which specific turn rendered the task unrecoverable.

**Failed Rollouts vs. Zero-Advantage Groups.** Let $R_{\mathrm{succ}}$ denote the domain-specific success return threshold. We define the set of failed rollouts within a sampled group as $N^- = \{\tau^{(n)} \mid R(\tau^{(n)}) < R_{\mathrm{succ}}\}$. While failed rollouts can still receive non-zero advantages when group returns vary, a *zero-advantage group* arises when return variance vanishes ($\sigma$=0 in Eq. eq. 1). In such cases, GRPO suffers from *zero-gradient silence* ($A_n^{\mathrm{rl}} = 0$), rendering policy updates inactive even when execution trajectories exhibit diverse failure behaviors.

**Empirical Observation.** We evaluate untrained `Qwen2.5-VL-3B-Instruct` across five VAGEN environments (Wang et al., 2026b) (Sec. 4.1, $N$=8). As shown in Fig. 1(a), low baseline success on Sokoban ($17.8\%$) and PrimitiveSkill ($8.9\%$) yields frequent zero-variance groups ($\sigma$=0 on $48.3\%$ and $16.5\%$), where identical returns cause all GRPO advantages $A_n^{\mathrm{rl}}$ to vanish despite diverse errors. In FrozenLake and Navigation ($0.0\%$ $\sigma$=0), the $-0.1$ per-step penalty differentiates returns across varying trajectory lengths,forming the $-0.1$ ladder in Figure 1(a), but $\sigma>0$ merely penalizes episode length without isolating the critical failure turn. Similarly, SVG visual scores yield low $\sigma$=0 ($0.9\%$), yet credit assignment remains episode-level rather than step-level.

**Takeaway.** Failed GRPO trajectories suffer from *zero-gradient silence* or *coarse credit*.

## 2.2 Disentangling OPD/OPSD: Physical State Rollback Dominates Skill Hints

Recently, OPD has gained traction by supplying token-level targets via negative KL divergence with a teacher (Agarwal et al., 2024; Lu et al., 2026). On-Policy Self-Distillation (OPSD) relaxes white-box constraints by conditioning policy updates on hindsight information (Wang et al., 2026a; Shinn et al., 2023; Zhao et al., 2026). However, the underlying mechanism driving OPSD in multi-turn VLM agents remains poorly understood. By systematically restoring environment states at critical steps and resampling trajectory suffixes with or without textual guidance, we isolate the true driver of post-failure performance gains.

**Pivot Step $t^*$.** Given a domain feasibility certificate $\mathsf{Feas}(s,k) \in \{0,1\}$ indicating whether state $s$ can reach the goal within $k$ steps (App. B), we define the *pivot step* $t^*$ as the first action that renders the task unfeasible under the remaining budget:

$$t^* = \min\big(\{t \in [0, T-1] : \mathsf{Feas}(s_{t+1}, T-t-1) = 0\} \cup \{T-1\}\big). \tag{2}$$

**Suffix Return Lift Rate.** Let $\tau_{t^*:}$ denote the suffix of $\tau$ from pivot step $t^*$ onward; its discounted return is $R(\tau_{t^*:}) = \sum_{t=t^*}^{T-1} \gamma^{t-t^*} r_t$. After rolling back the environment state to $s_{t^*}$ and resampling a new suffix $\tilde{\tau}_{t^*:} \sim \pi_{\text{base}}$ on faliure trajectories $N^-$, we record a successful recovery if $R(\tilde{\tau}_{t^*:}) > R(\tau_{t^*:})$. We quantify recovery via the *suffix return lift rate* $\Delta R$:

$$\Delta R = \big|\{\tau \in N^- : R(\tilde{\tau}_{t^*:}) > R(\tau_{t^*:})\}\big| \,/\, |N^-|. \tag{3}$$

**Counterfactual Rollback Setup.** On failed rollouts, we roll back the environment state to $s_{t^*}$ and resample a single suffix under four controlled conditions: (1) **no-hint**: resamples purely from the rolled-back state $s_{t^*}$ without skill hints; (2) **no-hint ($t^*$+1)**: rolls back state to $s_{t^*+1}$ to evaluate temporal localization sensitivity; (3) **3B-self**: appends failure mode $m^*$ and self-generated skill hints from $\pi_{\text{base}}$ (mirroring OPSD); (4) **8B-teacher**: appends skill hints generated by a larger teacher model (`Qwen3-VL-8B`, mirroring OPD).

**Empirical Findings.** Fig. 1(b) and (c) summarize the $n$=1 probe (numerical details in App. D): **(i)** Rollback $s_{t^*}$ with **no hints** boosts suffix returns $\Delta R$ across all tasks (3.3%–56.6%) while cutting zero-variance group rates ($48.3\% \rightarrow 41.7\%$ on Sokoban, $16.5\% \rightarrow 3.8\%$ on PrimitiveSkill; Fig. 1(c)),confirming that state rollback delivers the primary performance gains and resolves zero-gradient silence. **(ii)** Resetting just one step later ($t^*$+1) sharply drops $\Delta R$ on four tasks (e.g., Sokoban $3.3\% \rightarrow 0.9\%$, PrimitiveSkill $17.7\% \rightarrow 8.3\%$; Fig. 1(b)), demonstrating that recovery strictly hinges on temporal precision. **(iii)** Skill hints yield marginal additional gain (3B-self differs from no-hint by $\leq 2.6$ points; 8B-teacher advances 3B-self within 0.3–3.5 points).

**Takeaway. (i) State Rollback Drives Recovery:** State rollback at $t^*$ provides the predominant performance gain, serving as the core mechanism underlying OPSD/OPD paradigms. **(ii) Skill Hints Offer Marginal Value:** Skill hints, whether self-generated (3B-self) or teacher-provided (8B-teacher), yield negligible gains beyond pure state rollback. **(iii) Temporal Precision is Critical:** Delaying state rollback by even a single step ($t^*$+1) severely degrades recovery capacity.

### 2.3 From Physical Rollbacks to Parameter-Internalized Updates

These mechanistic findings expose the *Rollback Paradox* in post-training VLM agents: while trajectory recovery stems overwhelmingly from state rollback at the pivot turn ($t^*$), explicit physical rollbacks are computationally prohibitive during online RL and impossible in real-world, non-rewindable environments.

To bridge this gap and convert physical state rollback directly into parameter updates without simulator resets, a practical post-training framework must fulfill three core requirements: **(i) Non-Invasive Pivot Localization:** identifying the critical pivot step ($t^*$) directly from failure trajectory histories, bypassing environment resets; **(ii) Internalized Credit Assignment:** converting the localized pivot turn into fine-grained, token-level policy gradients directly on original failed rollouts without physical rollbacks; and **(iii) Zero Test-Time Overhead:** internalizing hindsight diagnostic capabilities into policy parameters to enable unprivileged deployment without additional skill prompts or auxiliary models at test time.

## 3 PIVOT: Pivot-Aware Internalized Visual On-Policy Training

In this section, we present PIVOT, a post-training framework that consolidates Student, Analyzer, and Teacher roles within a single parameterized VLM policy $\pi_\theta$. Specifically, $\pi_\theta$ non-invasively localizes the pivot step $t^*$ and diagnoses failure via a visual trajectory Analyzer (Sec. 3.1), and acts as its own Privileged Teacher to distill fine-grained visual hindsight into token-level policy updates on original failed rollouts (Sec. 3.2). By internalizing visual state restoration directly into model parameters, PIVOT eliminates environment rollbacks during training while maintaining zero computational and prompt overhead at test time.

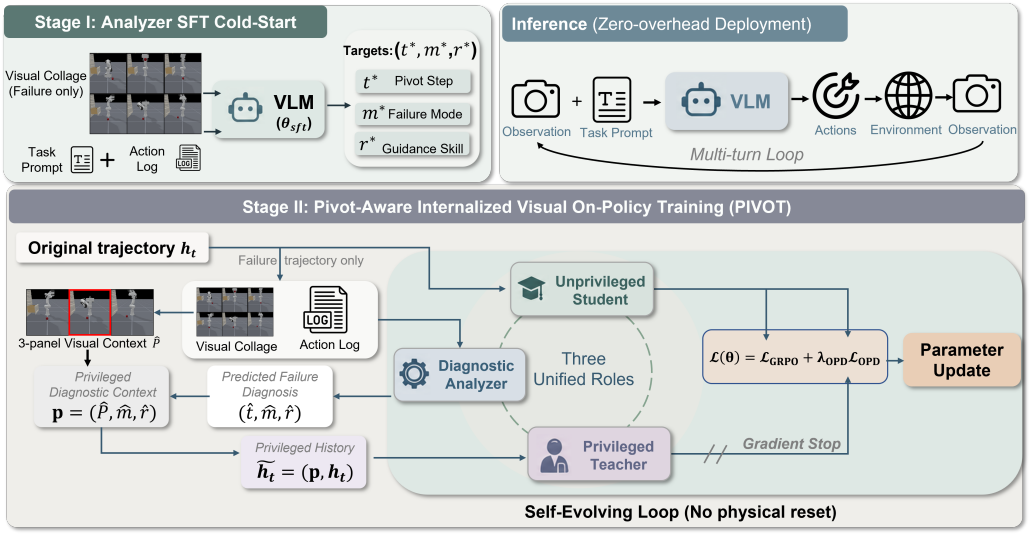



Figure 2: **PIVOT Architecture Overview.** PIVOT unifies Student, Analyzer, and Teacher roles within a single shared policy $\pi_\theta$. **Stage I:** SFT cold-starts the Analyzer to predict diagnostic triplets $(t^*, m^*, r^*)$ from visual trajectory collages $C(\tau)$ and action logs. **Stage II:** For failed rollouts, the Analyzer localizes the pivot step $\hat{t}$ to construct a privileged context $p$, enabling the detached Teacher to re-score failed tokens under $\tilde{h}_t = (p, h_t)$ for joint GRPO and confidence-gated OPD training. **Inference:** Analyzer and Teacher branches are stripped, deploying $\pi_\theta$ strictly as an unprivileged Student with zero computational and prompt overhead.

### 3.1 Stage I: Visual Diagnostic Cold-Start for Pivot Step Localization

To bypass state rollbacks during online RL training, the agent first learns to non-invasively localize the recoverable pivot step $t^*$ directly from VLM agent's trajectory history. As shown in Fig. 2 (Stage I), we initialize this diagnostic capability via the Supervised Fine-Tuning (SFT) phase.

**Diagnostic Target Triplets $(t^*, m^*, r^*)$.** Each failed trajectory $\tau \in N^-$ is annotated with a target diagnostic triplet $z_\tau = (t^*, m^*, r^*)$, where $t^*$ is the ground-truth pivot step from Eq. eq. 2, $m^* \in \mathcal{M}_{\text{fail}}$ denotes a discrete failure mode category (e.g., `deadlock`, `timeout`), and $r^*$ represents an optional guidance skill prompt generated offline via a teacher (App. G).

**Visual Collage Construction & Input.** The Analyzer receives a multi-frame visual collage $C(\tau) = \text{Grid}(o_0, \ldots, o_{T-1})$ augmented with explicit step indices, alongside the full action sequence $a_{0:T-1} = (a_0, \ldots, a_{T-1})$ and task prompt $q$. Crucially, this input relies strictly on observable trajectory histories without requiring access to environment states or internal memories.

**SFT Training.** We optimize $\pi_\theta$ on the diagnostic dataset $\mathcal{D}_{\text{sft}} = \{(q, C(\tau), a_{0:T-1}) \rightarrow (t^*, m^*, r^*)\}$ using standard autoregressive cross-entropy loss (App. F). The resulting checkpoint $\theta_{\text{sft}}$ equips $\pi_\theta$ with visual diagnostic capabilities, providing a robust cold-start for Stage II.

### 3.2 Stage II: Internalizing Localized Pivot Step Hindsight into Policy Updates

In Stage II, PIVOT translates localized pivot step hindsight directly into fine-grained, token-level policy gradients on original failed rollouts without physical environment resets (Fig. 2, Stage II). Each online RL iteration synchronizes on-policy rollout sampling with privileged target scoring, optimizing policy parameters under a unified joint objective.

**Role 1: Diagnostic Analyzer (Pivot Localization).** Given a sampled rollout group $\mathcal{G} = \{\tau^{(1)}, \ldots, \tau^{(N)}\}$, the Diagnostic Analyzer processes each failed rollout $\tau \in N^-$ to predict:

$$(\hat{t}, \hat{m}, \hat{r}) = \pi_\theta\big(q, C(\tau), a_{0:T-1}\big), \tag{4}$$



(a) Sokoban ($t^*$=4, `deadlock`). (b) FrozenLake ($t^*$=6, `timeout`). (c) Navigation ($t^*$=21, `budget`).

(d) PrimSkill ($t^*$=4, `timeout`). (e) SVG ($t^*$=1, `timeout`).

Figure 3: **Hindsight Visual Panels** $P_{t^*} = [o_{t^*-1}, o_{t^*}, o_{t^*+1}]$**.** Localized visual neighborhood extracted around the pivot step $t^*$ across five benchmark tasks (boundary frames zero-padded).

where $\hat{t}$ denotes the predicted pivot step, $\hat{m} \in \mathcal{M}_{\text{fail}}$ the predicted failure category, and $\hat{r}$ the predicted guidance skill. Using the predicted pivot step $\hat{t}$, the Analyzer extracts a 3-panel visual neighborhood crop $P_{\hat{t}} = [o_{\hat{t}-1}, o_{\hat{t}}, o_{\hat{t}+1}]$ centered around $\hat{t}$ (with boundary frames zero-padded), illustating in Fig. 3. The privileged diagnostic context is then constructed as $p = (P_{\hat{t}}, \hat{m}, \hat{r})$.

**Role 2: Privileged Teacher (Hindsight Target Scoring).** The Privileged Teacher re-evaluates the Student's original action tokens under the privileged history $\tilde{h}_t$, formed by prepending diagnostic context $p$ to the standard interaction history $h_t = (o_0, a_0, \ldots, o_t)$:

$$\tilde{h}_t = (p, h_t), \qquad \ell_t^{\text{tea}} = \text{sg}\big[\log \pi_\theta(a_t \mid \tilde{h}_t)\big]. \tag{5}$$

Without executing extra simulation steps or resetting environment states, the Teacher acts as a detached target evaluator: the stop-gradient operator $\text{sg}[\cdot]$ prevents Teacher-side parameter updates. Conditioning on $\tilde{h}_t$ enables the Teacher to provide precise token-level credit by implicitly answering how original actions should be re-weighted under privileged context $p$.

**Role 3: Acting Student (Token-Level Policy Update).** Concurrently, the Unprivileged Student branch evaluates action tokens under standard interaction histories $h_t$, producing unprivileged log-probabilities $\ell_t^{\text{stu}} = \log \pi_\theta(a_t \mid h_t)$. GRPO optimizes policy updates based on group-normalized advantages $A^{\text{rl}}$:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}\Big[m_{\text{act}} \min\big(\rho_t A^{\text{rl}},\ \text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon) A^{\text{rl}}\big)\Big], \tag{6}$$

where $\rho_t = \exp(\ell_t^{\text{stu}} - \ell_t^{\text{old}})$, $\ell_t^{\text{old}} = \log \pi_{\theta_{\text{old}}}(a_t \mid h_t)$, $\varepsilon > 0$ is the clipping threshold, and $m_{\text{act}}$ masks valid action tokens. For failed rollouts ($\mathbb{I}[R(\tau) < R_{\text{succ}}]$), confidence-gated OPD (Agarwal et al., 2024; Lu et al., 2026) (More details in App. F) provides dense token-level targets:

$$\mathcal{L}_{\text{OPD}}(\theta) = \mathbb{E}\Big[\mathbb{I}[R(\tau) < R_{\text{succ}}]\, m_{\text{act}}\, g_t\, (\text{sg}[\ell_t^{\text{tea}}] - \ell_t^{\text{stu}})\Big], \tag{7}$$

where $g_t = \sigma(\beta_{\text{opd}} \delta_t)$ is the confidence gating weight, $\delta_t = \ell_t^{\text{tea}} - \ell_t^{\text{stu}}$ measures the privileged log-probability gap, $\sigma(\cdot)$ is the sigmoid function, and $\beta_{\text{opd}} > 0$ controls the gating temperature.

The overall joint training objective is formulated as:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{GRPO}}(\theta) + \lambda_{\text{opd}} \mathcal{L}_{\text{OPD}}(\theta), \tag{8}$$

where $\lambda_{\text{opd}} > 0$ balances RL exploration and token-level self-distillation. When all rollouts within a group fail ($\sigma = 0$), $\mathcal{L}_{\text{GRPO}}$ vanishes while $\mathcal{L}_{\text{OPD}}$ actively maintains informative policy gradients.

**Inference.** During deployment (Fig. 2, top-right), Analyzer and Teacher branches are stripped away, executing $\pi_\theta$ strictly as an unprivileged Student $a_t \sim \pi_\theta(\cdot \mid h_t)$ with zero extra cost.

## 4 Experiments

### 4.1 Experimental Setup

**Environments.** To evaluate the learning dynamics and visual reasoning capabilities of VLM agents, we adopt the VAGEN (Wang et al., 2026b) evaluation suite, comprising five agentic tasks

across three paradigms: Cognitive Grid Puzzles, Embodied 3D Control, and Generative Reasoning. App. E details the task families, environment specifications, horizon limits, and metrics.

**Experimental Setup & Reference Baselines.** We evaluate PIVOT across two backbone models: `Qwen2.5-VL-3B-Instruct` (Bai et al., 2025b) and `Qwen3-VL-2B-Instruct` (Bai et al., 2025a). We compare against three reference baselines: (i) **Vanilla-GRPO**: Pure RL post-training initialized directly from raw Instruct checkpoints with Analyzer components disabled; (ii) **SFT-GRPO**: GRPO initialized from Stage I Analyzer SFT weights $\theta_{\text{sft}}$ (with OPD disabled during RL), isolating SFT initialization gains from token-level distillation; and (iii) **SDAR** (Lu et al., 2026): A reproduced state-of-the-art on-policy distillation baseline using a frozen `Qwen3-VL-7B-Instruct` teacher and Qwen2.5-VL-3B student (More details in App. I) with the same Gated OPD (Eq. 7), isolating Gated OPD's contribution.

**PIVOT Variants & Ablation Arms.** All PIVOT variants initialize $\pi_\theta$ from Stage I weights $\theta_{\text{sft}}$ and modulate the Teacher's privileged context $p$: (i) **P**: Includes localized visual panel only, $p = (P_{\hat{t}})$; (ii) **M**: Includes failure mode diagnosis only, $p = (\hat{m})$; (iii) **M+P** (*Default*): Combines localized visual panel, failure mode, and predicted pivot step, $p = (P_{\hat{t}}, \hat{m}, \hat{t})$; and (iv) **M+P+R**: Extends M+P by appending predicted guidance skill prompts, $p = (P_{\hat{t}}, \hat{m}, \hat{t}, \hat{r})$. Additionally, Tab. 2 incorporates a negative control **M+P-Random+R**, which replaces predicted pivot steps $\hat{t}$ with uniform random step indices to evaluate temporal localization sensitivity.

**Metrics.** Performance across puzzle and embodied control tasks is evaluated using the average Success Rate (SR) under sparse goal-completion rewards. For SVG Reconstruction, we report a composite visual similarity score averaging DINO and DreamSim embeddings.

**Implementation Details.** ***SFT stage.*** For each environment we sample failed on-policy rollouts from the raw Instruct checkpoint and retain ∼960–1.7k accepted trajectories after loose factual filtering (Tab. 11). Each example pairs an trajectory collage with a compact action log; supervision is JSON $(t^*, m^*, r^*)$, with task solvers pinning the pivot step $t^*$ and failure mode $m^*$ and a frozen teacher optionally writing guidance skill $r^*$ (App. G). We fine-tune the backbone for three epochs per environment to obtain a dedicated Analyzer $\theta_{\text{sft}}$. ***RL stage.*** We set $\theta \leftarrow \theta_{\text{sft}}$ for the shared Student ($\pi_\theta$), Analyzer, and Teacher. We run 250 policy updates with a GRPO group size of $N$=8; the OPD loss ($\lambda_{\text{opd}}$=0.01, gate $\beta_{\text{opd}}$=5) is applied only on failed trajectories. Validation uses 128 episodes per environment. Further SFT data, collage, optimization, reward and hyperparameter details, plus collages and JSON targets, are in Appendices G and H.

### 4.2 Main Results

**Main Results & SOTA Comparison.** As presented in Tab. 1, the full PIVOT framework (M+P+R) consistently outperforms all reference baselines across both backbones. On `Qwen2.5-VL-3B`, M+P+R achieves a 0.90 **overall accuracy**, delivering a $+8\%$ gain over SFT-GRPO ($0.83 \rightarrow 0.90$) and a $+13\%$ lift over pure Vanilla-GRPO. Crucially, PIVOT surpasses both the dense-reward SOTA (GLANCE-Full, 0.86) and the reproduced on-policy distillation baseline SDAR (0.79; App. I), achieving these state-of-the-art results with zero inference-time overhead. The gains are most pronounced in *cognitive grid puzzles* and *embodied control*—where early pivot blunders render trajectory suffixes unrecoverable—highlighted by Sokoban surging from 0.82 to 0.95 and PrimitiveSkill reaching 1.00 (vs. 0.94 under SFT-GRPO).

**Domain Insights & Backbone Scaling.** PIVOT demonstrates strong generalizability and progressive scaling behavior across task domains and model families. On *generative reasoning* (SVG) with Qwen2.5-VL-3B, M+P achieves a peak composite score of 0.84; appending guidance skill prompts (M+P+R) keeps DINO at 0.93 but lowers DreamSim ($0.74 \rightarrow 0.71$). On `Qwen3-VL-2B` (SFT-GRPO at 0.82), enriching the Teacher's privileged context yields continuous performance lifts: M (0.86) → M+P (0.91) → M+P+R (0.92 **overall**, $+12\%$ over SFT-GRPO). Under M+P+R, task-level scores reach 0.90 on FrozenLake, 0.95 on Navigation, and 0.85 on SVG; on Sokoban, M+P attains the peak (0.97).

Table 1: Main results across five VLM agentic benchmarks. Vanilla-GRPO, SFT-GRPO, and M/M+P/M+P+R denote vanilla GRPO, GRPO after Analyzer SFT, and Teacher context with failure mode $\hat{m}$, pivot panel $P_{\hat{t}}$, or added skill text $\hat{r}$; SDAR (Lu et al., 2026) is our reproduced Gated OPD baseline. Bold marks the best result; cyan percentages on M+P+R are relative gains over SFT-GRPO in the same block; light-gray rows follow (Wang et al., 2026b; Li et al., 2026).

| Method | Cognitive Grid Puzzles | | Embodied 3D Control | | | | | | | | Generative Reasoning | | | All |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Sokoban | Frozen Lake | Navigation | | | PrimitiveSkill | | | | | SVG | | | |
| | | | Base | Com. | Avg. | Place | Stack | Draw. | Align | Avg. | DINO | DS | Avg. | |
| *Open-source VLMs* | | | | | | | | | | | | | | |
| Qwen2.5-VL-72B | 0.20 | 0.44 | 0.70 | 0.77 | 0.74 | 1.00 | 0.50 | 0.00 | 1.00 | 0.63 | 0.84 | 0.62 | 0.73 | 0.55 |
| Qwen2.5-VL-7B | 0.14 | 0.14 | 0.33 | 0.38 | 0.35 | 0.00 | 0.00 | 0.00 | 0.75 | 0.19 | 0.84 | 0.27 | 0.56 | 0.28 |
| Qwen2.5-VL-3B (Bai et al., 2025b) | 0.13 | 0.14 | 0.20 | 0.26 | 0.23 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.79 | 0.30 | 0.54 | 0.21 |
| VLM-R1-3B (Shen et al., 2025) | 0.16 | 0.15 | 0.33 | 0.34 | 0.34 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.79 | 0.27 | 0.54 | 0.24 |
| *Proprietary VLMs* | | | | | | | | | | | | | | |
| o4-mini | 0.44 | 0.82 | 0.75 | 0.75 | 0.75 | 1.00 | 0.50 | 0.00 | 0.75 | 0.56 | 0.90 | 0.66 | 0.78 | 0.67 |
| GPT-4o | 0.43 | 0.54 | 0.75 | 0.69 | 0.72 | 0.50 | 0.63 | 0.00 | 0.88 | 0.50 | 0.91 | 0.69 | 0.80 | 0.60 |
| Gemini 2.5 Pro | 0.58 | 0.78 | 0.63 | 0.63 | 0.63 | 0.63 | 0.63 | 0.00 | 0.75 | 0.50 | 0.93 | 0.78 | 0.86 | 0.67 |
| Claude 4.5 Sonnet | 0.31 | 0.80 | 0.67 | 0.67 | 0.67 | 0.63 | 0.50 | 0.00 | 1.00 | 0.53 | 0.95 | 0.81 | 0.88 | 0.64 |
| Claude 3.7 Sonnet | 0.25 | 0.69 | 0.48 | 0.47 | 0.47 | 0.63 | 0.13 | 0.00 | 1.00 | 0.44 | 0.94 | 0.77 | 0.85 | 0.54 |
| *Previous SOTA with World Model Reasoning for Visual State Reconstruction* (Qwen2.5-VL-3B-Instruct) | | | | | | | | | | | | | | |
| VAGEN-Full (Wang et al., 2026b) | 0.79 | 0.72 | 0.80 | 0.81 | 0.81 | 1.00 | 0.88 | 1.00 | 1.00 | 0.97 | 0.90 | 0.66 | 0.78 | 0.81 |
| GLANCE-Full (Li et al., 2026) | 0.85 | 0.78 | 0.86 | 0.88 | 0.87 | 1.00 | 0.88 | 1.00 | 1.00 | 0.97 | 0.92 | 0.70 | 0.81 | 0.86 |
| *Reproduced OPD baseline* (Qwen2.5-VL-3B-Instruct) | | | | | | | | | | | | | | |
| SDAR (Lu et al., 2026) | 0.52 | 0.77 | 0.88 | 0.84 | 0.86 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.91 | 0.68 | 0.80 | 0.79 |
| PIVOT (Qwen2.5-VL-3B-Instruct) | | | | | | | | | | | | | | |
| Vanilla-GRPO | 0.78 | 0.71 | 0.87 | 0.75 | 0.81 | 0.92 | 0.86 | 0.89 | 0.93 | 0.90 | 0.90 | 0.66 | 0.78 | 0.80 |
| SFT-GRPO | 0.82 | 0.75 | 0.93 | 0.77 | 0.85 | 0.96 | 0.90 | 0.93 | 0.97 | 0.94 | 0.92 | 0.68 | 0.80 | 0.83 |
| M | 0.86 | 0.83 | 0.94 | 0.78 | 0.86 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **0.93** | 0.73 | 0.83 | 0.88 |
| M+P | 0.90 | 0.77 | 0.95 | **0.83** | 0.89 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **0.93** | **0.74** | **0.84** | 0.88 |
| M+P+R | **0.95** +16% | **0.84** +12% | **0.97** +4% | **0.83** +8% | **0.90** +6% | **1.00** +4% | **1.00** +11% | **1.00** +8% | **1.00** +3% | **1.00** +6% | **0.93** +1% | 0.71 +4% | 0.82 +3% | **0.90** +8% |
| PIVOT (Qwen3-VL-2B-Instruct) | | | | | | | | | | | | | | |
| Vanilla-GRPO | 0.77 | 0.89 | 0.97 | 0.73 | 0.85 | 0.77 | 0.77 | 0.77 | 0.77 | 0.77 | 0.90 | 0.68 | 0.79 | 0.81 |
| SFT-GRPO | 0.80 | 0.87 | 0.92 | 0.74 | 0.83 | 0.83 | 0.80 | 0.81 | 0.84 | 0.82 | 0.91 | 0.67 | 0.79 | 0.82 |
| M | 0.82 | 0.81 | 0.93 | 0.75 | 0.84 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | 0.92 | 0.70 | 0.81 | 0.86 |
| M+P | **0.97** | 0.89 | 0.97 | 0.80 | 0.89 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **0.93** | 0.71 | 0.82 | 0.91 |
| M+P+R | 0.92 +15% | **0.90** +3% | **0.98** +7% | **0.91** +23% | **0.95** +14% | **1.00** +20% | **1.00** +25% | **1.00** +23% | **1.00** +19% | **1.00** +22% | **0.93** +2% | **0.76** +13% | **0.85** +8% | **0.92** +12% |

## 4.3 Ablation Study

**Impact of Pivot Step Localization.** Tab. 2 dissects diagnostic contributions. Compared to Vanilla-GRPO (0.78 Sokoban / 0.81 Navigation), single channels $P$ and $M$ both raise accuracy to $\sim$ 0.86, their combination ($M+P$) reaches 0.90/0.89, and the full stack ($M+P+R$) peaks at 0.95/0.90. Crucially, substituting the localized step $t^*$ with a random frame ($M+P$-Random$+R$) degrades performance below Vanilla-GRPO (0.75/0.80). This negative control confirms that arbitrary visual context introduces harmful noise, proving precise pivot localization indispensable.

| Variant | Sok. | Nav. |
|---|---|---|
| Vanilla-GRPO | 0.78 | 0.81 |
| P | 0.86 | 0.85 |
| M | 0.86 | 0.86 |
| M+P | 0.90 | 0.89 |
| M+P+R | **0.95** | **0.90** |
| M+P-Random+R | 0.75 | 0.80 |

Table 2: Pivot-step ablation.

**Optimal Diagnostic Context.** Tab. 1 shows that progressively enriching privileged context from $M$ to $M+P+R$ delivers strong multimodal synergy, driving overall performance to 0.90 on Qwen2.5-VL-3B (+8% over SFT-GRPO) and 0.92 on Qwen3-VL-2B (+12%). The full $M+P+R$ stack peaks on FrozenLake, Navigation, and PrimitiveSkill on both backbones, and on Sokoban for 3B (0.95); on Qwen3-VL-2B, $M+P$ attains the Sokoban peak (0.97). SVG is mixed: on 3B the predicted guidance skill $\hat{r}$ slightly degrades the composite (0.84→0.82), whereas on 2B $M+P+R$ is best (0.85). Overall, pairing visual state restoration with failure modes and guidance skills forms the primary foundation for effective credit assignment.

## 4.4 Further Analysis

**Self-Evolving Pivot Localization.** Fig. 4 illustrates the co-evolution of the Analyzer's diagnostic accuracy ($\hat{t} = t^*$) alongside policy optimization on FrozenLake, as this environment shortest-path solver provides ground-truth pivot labels $t^*$ on failed rollouts. Under M+P+R, 15-step moving accuracy starts near 0.56, rises after step 50 to 0.90 by step 100, and later holds near 0.95. This trajectory confirms the Analyzer is genuinely self-evolving, forming a loop where sharper pivot localization yields finer credit.

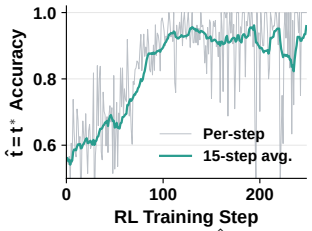



Figure 4: FrozenLake $\hat{t}$ accuracy.

**Training Efficiency.** Fig. 5 compares the average per-step runtime during RL optimization on Sokoban. Despite incorporating a multi-role diagnostic branch, PIVOT achieves a lower per-update latency (95 s) than standard Vanilla-GRPO (102 s). This advantage stems from PIVOT rapidly suppressing unviable actions and converging toward shorter, successful trajectories, which significantly reduces rollout sampling time as it's the primary computational bottleneck in VLM RL. In contrast, delegating hindsight diagnosis to an external frozen `Qwen3-VL-8B` teacher introduces cross-model inference overhead, inflating the step time to 373 s (3.9×). At test time, stripping all diagnostic branches leaves an unprivileged student with zero parameter or inference overhead.

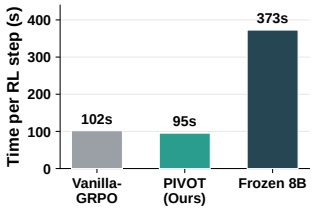



Figure 5: Time per RL step (s).

## 5 Related Work

**RL and hindsight distillation for VLM agents.** VLMs serve as goal-driven agents across web navigation, device control, and embodied decision-making tasks (Koh et al., 2024; Shridhar et al., 2021). Beyond test-time prompting with frozen backbones (Yao et al., 2023; Shinn et al., 2023), task adaptability is enhanced via fine-tuning and online RL (Bai et al., 2025b; Shen et al., 2025; Ouyang et al., 2022). However, standard outcome RL and process supervision (Schulman et al., 2017; Shao et al., 2024; Guo et al., 2025; Li et al., 2026; Lightman et al., 2024; Wang et al., 2024) rely on trial-and-error or dense shaping, providing coarse feedback that fails to isolate unrecoverable decision points. Retrospective feedback from hindsight, reflection, and memory systems (Andrychowicz et al., 2017; Madaan et al., 2023; Zhao et al., 2024; Wang et al., 2023) yields post-hoc signals but incurs severe prompt-length and latency overhead at inference. While on-policy distillation internalizes feedback into parameters via self-generated rollouts (Ross et al., 2011; Agarwal et al., 2024; Lu et al., 2026; Wang et al., 2026a), agentic self-teachers remain structurally decoupled or prompt-dependent (Wu et al., 2026). In contrast, PIVOT localizes pivot turns from visual histories and distills token-level hindsight directly into policy parameters without dense shaping, extra prompts, or environment rollbacks.

**Policy entropy collapse and multi-turn credit assignment.** Outcome-based RL frequently suffers from policy entropy collapse, prematurely shrinking pass@$k$ exploration diversity and trapping policies in local optima (Yue et al., 2026; Cui et al., 2025; Yu et al., 2026). This degeneration is acute in multi-turn agent interaction (Wang et al., 2026b; Li et al., 2026), where standard policy gradients over-reinforce frequent success paths while suppressing alternative exploration branches (Schulman et al., 2017; Cui et al., 2025). In sequential settings, an uncorrected early blunder distorts subsequent state observations, compounding downstream execution errors and pruning viable subtrees (Andrychowicz et al., 2017; Sinha et al., 2026). Although recent diversity-preserving techniques attempt to mitigate policy sharpening, they remain bound to trajectory-level returns or group-level advantages (Sinha et al., 2026; Shao et al., 2024; Yu et al., 2026; Kazemnejad et al., 2025). Consequently, they sustain only surface-level diversity without identifying the exact pivot turn where task failure originated (Uesato et al., 2022; Lightman et al., 2024). PIVOT bridges this credit-assignment gap by isolating critical pivot steps via visual panels and distilling confidence-gated token-level gradients onto failed rollouts, maintaining informative optimization even within zero-variance failure groups where conventional RL advantages vanish.

## 6 Conclusion

In this work, we address the credit assignment bottleneck of outcome-based RL in multi-turn VLM agents. Our analysis shows that post-failure recovery is driven by physical state restoration at pivot steps ($t^*$), rather than textual skill guidance. To avoid prohibitive simulator rollbacks, we present PIVOT, which internalizes visual state restoration into a unified Student-Analyzer-Teacher policy. By localizing $t^*$ and distilling gated hindsight onto failed rollouts, PIVOT removes physical resets at training and prompt overhead at deployment, outperforming SOTAs and prior distillation methods.

## AI USE STATEMENT

In this work, we used generative AI tools for research execution including implementing analysis scripts, checking table/figure consistency, and debugging the paper build. Additionally, we used generative AI tools to aid or polish writing, for retrieval and discovery of related work, and to draft sections of the paper. We have reviewed all AI-assisted work: every added citation was manually checked against the original paper or arXiv record, and LLM-assisted code, figures, and wording were verified by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

## REFERENCES


Åström, Karl Johan. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, pp. 21246–21263, 2024.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. volume 30, 2017.

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Yi Ding and Ruqi Zhang. Does on-policy distillation really distill? from noisy teacher to self-improvement. *arXiv preprint arXiv:2608.31046*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Ayano Hiranaka, Minjune Hwang, Sharon Lee, Chen Wang, Li Fei-Fei, Jiajun Wu, and Ruohan Zhang. Primitive skill-based robot learning from human evaluative feedback. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7817–7824. IEEE, 2023.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

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.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Haoxi Li, Qinglin Hou, Jianfei Ma, Jinxiang Lai, Tao Han, Sikai Bai, Jingcai Guo, Jie ZHANG, and Song Guo. What you think is what you see: Driving exploration in VLM agents via visual-linguistic curiosity. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. volume 35, pp. 27730–27744, 2022.

Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images and text. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16175–16186. IEEE, 2025.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Sebastian Schrader. Sokoban. Gymnasium environment, 2018. `https://github.com/mpSchrader/gym-sokoban`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

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. *arXiv preprint arXiv:2504.07615*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Abhijeet Sinha, Sundari Elango, and Dianbo Liu. Expected return causes outcome-level mode collapse in reinforcement learning and how to fix it with inverse probability scaling. *arXiv preprint arXiv:2601.21669*, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, et al. Maniskill3: Gpu parallelized robot simulation and rendering for generalizable embodied ai. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. *Advances in Neural Information Processing Systems*, 38, 2026.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents. *arXiv preprint arXiv:2604.10674*, 2026a.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, et al. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. *Advances in Neural Information Processing Systems*, 38:172871–172933, 2026b.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, et al. SEED: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *Advances in Neural Information Processing Systems*, 38:57654–57689, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

# PIVOT: PIVOT-AWARE ON POLICY SELF DISTILLATION FOR MULTI-TURN VLM AGENTS
## *–Supplementary Material (Text)–*

## APPENDIX CONTENTS

A Notation . . . 14
B Pivot Certificates . . . 14
C GRPO Grouping and Zero-Gradient Groups . . . 15
D Counterfactual Recovery . . . 16
E Environments and Rewards . . . 16
F Training Objectives . . . 18
G SFT Data and Optimization . . . 18
H Analyzer Collages, Prompts, and SFT Targets . . . 19
I SDAR Reproduction . . . 32
J Training Curves . . . 33
K Discussion and Limitations . . . 33

## A NOTATION

Tab. 3 lists symbols used in Sec. 2–3.2 and their meanings.

## B PIVOT CERTIFICATES

Offline Analyzer SFT and counterfactual probes pin a diagnostic pair $(t^*, m^*)$ on each failed rollout. Tab. 4 lists the certificate used per VAGEN task; all modes lie in the closed vocabulary $\mathcal{M}_{\text{fail}}$ referenced in Sec. 3.1.

**Pivot step $t^*$.** When a feasibility test $\mathsf{Feas}(s, k) \in \{0, 1\}$ is defined (Sokoban, FrozenLake, Navigation proxy, PrimitiveSkill stage budget), we use the main-text pivot step Eq. eq. 2: the first turn after which the goal is unreachable within the remaining $k_t = T - t - 1$ actions, defaulting to $T-1$ if the trajectory never becomes infeasible. Navigation may pin an *earlier* $t^*$ when a terminal event fires first (invalid-action suffix, blocked move, or stagnation window); only if none fire do we fall back to the travel-time proxy in Eq. eq. 2. SVG does not use spatial Feas: each step submits a full canvas, so we set $t^*$ to the index of the last scored submission (typically $T-1$ with $T_{\max}=2$).

**Failure mode $m^*$ on grid puzzles.** For Sokoban and FrozenLake, after $t^*$ from Eq. eq. 2 we label timeout versus irreversible deadlock with two reachability checks at the post-action state $s_{t^*+1}$:

$$m^* = \begin{cases} \texttt{timeout}, & \mathsf{Feas}(s_{t^*+1}, k_{t^*}) = 0 \ \wedge \ \mathsf{Feas}(s_{t^*+1}, \infty) = 1, \\ \texttt{deadlock}, & \mathsf{Feas}(s_{t^*+1}, \infty) = 0, \end{cases} \tag{9}$$

where $k_{t^*} = T - t^* - 1$ and the $\mathsf{Feas}(\cdot, \infty)$ probe uses unbounded search on grid worlds.

**Failure modes on other benchmarks.** Eq. eq. 9 does *not* apply outside the two grid puzzles. Navigation assigns the earliest matching event among `format`, `blocked`, and `stagnation`; if none occur, `budget` marks insufficient remaining moves under a Euclidean travel-time lower bound. PrimitiveSkill compares unfinished manipulation stages with the remaining turn budget (`timeout`) and may further tag format, target, or ordering mistakes at $t^*$. SVG reads parsing validity and DINO/DreamSim progression on the final canvas (`format`, `regression`, `stagnation`, `mismatch`). Tab. 4 summarizes each environment in one place.

Table 3: Notation (Sec. 2–3.2).

| Symbol | Meaning |
|---|---|
| $\mathcal{M}$ | Partially observable Markov decision process. |
| $\mathcal{S}, \mathcal{O}, \mathcal{A}$ | Latent-state, observation, and action spaces. |
| $\mathcal{P}, \Omega$ | State-transition and observation kernels. |
| $\mathcal{R}, \gamma$ | Per-step reward function and discount factor. |
| $q, \mathcal{Q}$ | Task prompt and task-prompt distribution. |
| $s_t, o_t, a_t, r_t$ | Latent state, observation, action, and reward at turn $t$. |
| $h_t$ | Observable interaction history $(o_0, a_0, \ldots, o_t)$ (Sec. 2.1). |
| $\tilde{h}_t$ | Privileged Teacher history with diagnostic context prepended, $\tilde{h}_t = (p, h_t)$ (Eq. eq. 5). |
| $\pi_\theta$ | Shared autoregressive VLM: Student $a_t \sim \pi_\theta(\cdot \mid h_t)$; Analyzer outputs $(\hat{t}, \hat{m}, \hat{r})$ from $x_\tau$ (Eq. eq. 4); Teacher supplies $\ell^{\text{tea}}$ under $\tilde{h}_t$. |
| $\theta$ | Trainable weights shared by all roles of $\pi_\theta$. |
| $J(\theta)$ | Outcome-based objective $\mathbb{E}_{q\sim\mathcal{Q},\, \tau\sim\pi_\theta}[R(\tau)]$. |
| $\tau, T, T_{\max}$ | Trajectory, realized trajectory length, and maximum horizon. |
| $R(\tau)$ | Discounted trajectory return $\sum_{t=0}^{T-1} \gamma^t r_t$. |
| $R_{\text{succ}}$ | Success-return threshold; rollouts with $R(\tau) < R_{\text{succ}}$ are failures for OPD (Eq. eq. 7). |
| $N^-$ | Failed-rollout set $\{\tau : R(\tau) < R_{\text{succ}}\}$; $\lvert N^-\rvert$ is its cardinality. |
| $\mathcal{G}_q$ | On-policy GRPO group $\{\tau_q^{(1)}, \ldots, \tau_q^{(N)}\}$ for prompt $q$. |
| $N$ | Number of on-policy trajectories sampled per prompt. |
| $\mu, \sigma, \epsilon$ | Group return mean, standard deviation, and stabilizer in Eq. eq. 1 (distinct from clip $\varepsilon$). |
| $A_n^{\text{rl}}$ | Group-relative advantage of rollout $n$; per-prompt form $A_{q,n}^{\text{rl}}$. |
| $k_t$ | Remaining action budget after step $t$ in Eq. eq. 2: $T - t - 1$. |
| $\mathsf{Feas}(s, k)$ | Indicator that state $s$ is goal-reachable within $k$ actions. |
| $t^*$ | Pivot step (Eq. eq. 2): first infeasible action, or $T-1$ if none. |
| $\tau_{t^*:}$ | Original trajectory suffix from pivot step $t^*$. |
| $\tilde{\tau}_{t^*:}$ | Resampled suffix after state restoration at $t^*$. |
| $\Delta R$ | Suffix-return lift rate in Eq. eq. 3 (distinct from the OPD log-prob gap $\delta_t$). |
| $\mathcal{M}_{\text{fail}}$ | Closed vocabulary of discrete failure modes. |
| $m^*, r^*$ | Ground-truth failure mode and optional guidance skill in SFT labels. |
| $\hat{t}, \hat{m}, \hat{r}$ | Analyzer-predicted pivot, failure mode, and guidance skill. |
| $z_\tau$ | SFT diagnosis triplet $(t^*, m^*, r^*)$; $r^*$ may be empty. |
| $C(\tau)$ | Full-trajectory visual collage $C(\tau) = \text{Grid}(o_0, \ldots, o_{T-1})$ with step labels (Sec. 3.1). |
| $a_{0:T-1}^{\text{txt}}$ | Compact textual action transcript fed to the Analyzer (serializes the action sequence $a_{0:T-1}$ in Eqs. eq. 4 and eq. 10). |
| $x_\tau$ | Serialized Analyzer input $(q, C(\tau), a_{0:T-1}^{\text{txt}})$. |
| $P_{\hat{t}}$ | Local visual panel $[o_{\hat{t}-1}, o_{\hat{t}}, o_{\hat{t}+1}]$ (boundary frames zero-padded; Sec. 3.2). |
| $p$ | Teacher privileged context; ablation arms use $(P_{\hat{t}})$, $(P_{\hat{t}}, \hat{m})$, $(P_{\hat{t}}, \hat{m}, \hat{t})$, or $(P_{\hat{t}}, \hat{m}, \hat{t}, \hat{r})$ (Sec. 4.1). |
| $\pi_{\text{base}}$ | Fixed untrained instruct checkpoint for offline rollouts and the Sec. 2.2 probe. |
| $\pi_{\theta_{\text{old}}}$ | Frozen behavior policy for on-policy collection; synced from $\theta$ after each RL update. |
| $\nu_\tau$ | Loose factual-filter accept flag (App. G). |
| $\mathcal{D}_{\text{sft}}$ | Accepted SFT set after loose filtering. |
| $m_{\text{act}}$ | Valid-action token mask (distinct from failure mode $m^*$). |
| $\ell^{\text{stu}}, \ell^{\text{old}}, \ell^{\text{tea}}$ | Student, frozen-behavior, and Teacher token log-probabilities. |
| $\rho$ | Token-level importance ratio $\exp(\ell^{\text{stu}} - \ell^{\text{old}})$. |
| $\varepsilon$ | GRPO PPO-style clip radius in Eq. eq. 6. |
| $\delta_t, g_t$ | Privileged log-prob gap $\delta_t = \ell_t^{\text{tea}} - \ell_t^{\text{stu}}$ and gate $g_t = \sigma(\beta_{\text{opd}}\delta_t)$ (Eq. eq. 7; token-level form in Eq. eq. 11). |
| $\beta_{\text{opd}}, \lambda_{\text{opd}}$ | OPD gate sharpness and joint-loss weight. |

## C GRPO Grouping and Zero-Gradient Groups

We form GRPO groups as contiguous blocks of $N{=}8$ rollouts that share the same prompt key (`task_id`, `goal_idx`, `eval_set`). Blocks with fewer than eight rollouts are discarded, leaving 720 of 1680 Navigation rollouts and seven PrimitiveSkill rollouts unused. Tab. 5 reports two group-level rates: **All-fail** is the fraction of groups with no successful rollout; $\sigma{=}0$ is the fraction whose

Table 4: Offline pivot certificate $(t^*, m^*)$ by environment (Eq. eq. 2 when applicable; Eq. eq. 9 for Sokoban/FrozenLake only).

| **Environment** | **Pivot** $t^*$ | **Failure modes** $m^*$ |
|---|---|---|
| Sokoban | Eq. eq. 2; Feas = box-pushing plan $\leq k$ | `timeout`, `deadlock` |
| FrozenLake | Eq. eq. 2; Feas = hole-free path $\leq k$ | `timeout`, `deadlock` |
| Navigation | earliest terminal event; else Eq. eq. 2 (travel-time proxy) | `format`, `blocked`, `stagnation`, `budget` |
| PrimitiveSkill | Eq. eq. 2; unfinished stages vs. remaining turns | `timeout`, `format`, `wrong_target`, `order_error` |
| SVG | index of last scored canvas | `format`, `regression`, `stagnation`, `mismatch` |

Table 5: Untrained 3B GRPO groups ($N$=8).

| Environment | Traj. | Grps. | SR (%) | All-fail (%) | $\sigma$=0 (%) |
|---|---|---|---|---|---|
| Sokoban | 1440 | 180 | 17.8 | 48.3 | 48.3 |
| FrozenLake | 1728 | 216 | 20.3 | 39.4 | 0.0 |
| Navigation | 960 | 120 | 43.0 | 5.8 | 0.0 |
| PrimitiveSkill | 2912 | 364 | 8.9 | 46.2 | 16.5 |
| SVG | 1728 | 216 | 0.1 | 99.5 | 0.9 |
| Pooled | 8768 | 1096 | — | 51.3 | 13.6 |

Table 6: Counterfactual rollback arms.

| Arm | Injected at restore |
|---|---|
| no-hint | none |
| 3B-self | $m^*$ pinned; $\hat{r}$ from untrained 3B |
| 8B-teach. | $m^*$ pinned; $\hat{r}$ from `Qwen3-VL-8B` |

eight returns match at $10^{-8}$ rounding. These two rates need not agree when failures receive different partial credit, as on FrozenLake and SVG.

## D Counterfactual Recovery

**Counterfactual recovery protocol.** Tab. 6 summarizes the rollback arms used in Sec. 2.2. Every arm starts from the same failed untrained `Qwen2.5-VL-3B-Instruct` rollouts, restores the environment at the solver-pinned pivot step $t^*$ (Eq. eq. 2), and resamples the suffix with $\pi_{\text{base}}$. A retry counts as a lift when the new suffix return strictly exceeds the original suffix return from $t^*$. By default we restore at $t_{\text{restore}}$=$t^*$ and roll forward to the horizon; the $t^*$+1 control instead restores one step later as a negative control, and lift collapses on all five environments. Hinted arms inject solver-pinned $m^*$ plus an arm-specific guidance string into the actor prompt at restore; the no-hint arm skips injection entirely. Tab. 7 gives the main-text $n$=1 results at $t^*$, and Tab. 8 reports the matching no-hint run at $t^*$+1.

## E Environments and Rewards

**Benchmarks.** We evaluate five tasks from VAGEN (Wang et al., 2026b) (Tab. 9), grouped into the three families of Tab. 1.

**Sokoban (Schrader, 2018).** In this classic puzzle, the agent must push all boxes onto target locations. The visual state is a 6×6 grid ($T_{\max}$=9), and the action space is discrete (up, down, left, right). Validation reports success rate.

**FrozenLake (Towers et al., 2026).** The agent navigates a 4×4 grid to reach a goal while avoiding holes ($T_{\max}$=9). The visual state and discrete action space match Sokoban; we disable the slippery setting for determinism. Validation reports success rate.

Table 7: Suffix lift $\Delta R$ at $t^*$ ($n$=1; % with $R(\tilde{\tau}_{t^*:}) > R(\tau_{t^*:})$).

| Environment | Fail | no-hint | 3B-self | 8B-teach. |
|---|---|---|---|---|
| Sokoban | 1165 | 3.3 | 4.1 | 4.4 |
| FrozenLake | 1378 | 18.2 | 16.8 | 19.5 |
| Navigation | 969 | 56.6 | 50.8 | 52.7 |
| PrimitiveSkill | 2898 | 17.7 | 20.3 | 19.6 |
| SVG | 1727 | 22.4 | 22.8 | 24.3 |

Table 8: $n$=1 no-hint lift at $t^*$ vs. $t^*$+1 (%).

| Environment | $t^*$ | $t^*$+1 |
|---|---|---|
| Sokoban | 3.3 | 0.9 |
| FrozenLake | 18.2 | 8.1 |
| Navigation | 56.6 | 53.6 |
| PrimitiveSkill | 17.7 | 8.3 |
| SVG | 22.4 | 16.2 |

Table 9: VAGEN tasks: $T_{\max}$ and validation metrics.

| **Task** | $T_{\max}$ | **Metric** |
|---|---|---|
| *Cognitive Grid* | | |
| Sokoban [6, 6], 1 box | 9 | success rate |
| FrozenLake 4×4 | 9 | success rate |
| *Embodied 3D* | | |
| Navigation (Base / Com.) | 25 | avg. SR |
| PrimitiveSkill (4 skills) | 6 | mean SR |
| *Generative* | | |
| SVG reconstruction | 2 | mean DINO & DreamSim |

Table 10: Episode return $R(\tau)$ (GRPO).

| **Env** | **Reward structure** |
|---|---|
| Sokoban | $-0.1$/step; $+11$ goal; fail $R = -0.9$ |
| FrozenLake | $-0.1$/step; sparse goal |
| Navigation | $-0.1$/step; $+10$ on target |
| PrimitiveSkill | $-0.1$/step; $+2$/stage; $+10$ success |
| SVG | last canvas: $\frac{1}{2}$(DINO + DS) |

Table 12: Stage II RL hyperparameters (3B & 2B).

| Item | Value |
|---|---|
| Shared $\pi_\theta$ | `Qwen2.5-VL-3B` / `Qwen3-VL-2B` |
| Analyzer / Teacher | `policy_vllm`; panel $P_t$ |
| GRPO $N$, clip $\varepsilon$ | 8; 0.2 |
| $\gamma$; actor LR | 0.95; $1\times10^{-6}$ |
| KL coeff. | `low_var_kl`, 0.01 |
| $\lambda_{\text{opd}}$ / $\beta_{\text{opd}}$ | 0.01 / 5 |
| OPD mask | failed traj., all valid tokens |
| Invalid-action pen. | 0.1 |
| Temps (train / val) | 1.0 / 0.4 |
| History; max response | 2; 512 |
| Batch / val $n$ | 16 / 128 (Nav. 8) |
| Updates; GPUs | 250 (Sok. 350); 8 |

Table 11: Analyzer SFT splits (90/10; 3B packs, same recipe on 2B).

| Environment | Acc. | Train | Val |
|---|---|---|---|
| Sokoban | 1170 | 1053 | 117 |
| FrozenLake | 1375 | 1237 | 138 |
| Navigation | 959 | 863 | 96 |
| PrimitiveSkill | 1677 | 1509 | 168 |
| SVG | 1719 | 1547 | 172 |

**Navigation (Kolve et al., 2017; Yang et al., 2025).** A 3D embodied task ($T_{\max}$=25): the agent follows instructions to find an object from a first-person view with discrete actions (e.g., `moveahead`). We average success rate on the Base and Common-sense splits.

**PrimitiveSkill (Tao et al., 2025; Hiranaka et al., 2023).** The agent controls a Panda arm to perform tabletop manipulation ($T_{\max}$=6). Actions use a hybrid space (e.g., `pick(`$x, y, z$`)`): the policy must ground objects in a third-person 3D scene to coordinates. Validation reports mean success rate over four skills.

**SVG reconstruction (Rodriguez et al., 2025).** The agent generates SVG code that replicates a target image ($T_{\max}$=2). The action space is open-ended text: each step submits a full canvas. Validation averages DINO and DreamSim on the final submission.

**Rewards and task specs.** Tables 9–12 collect per-task horizons and validation metrics, the episode-return structure used in GRPO (logged as `final_reward` with discount $\gamma$=0.95), Analyzer SFT split sizes, and Stage II training settings. Invalid actions receive an additional 0.1 penalty on top of the domain reward. For OPD masking we mark a rollout as failed when its logged return falls below the success threshold $R_{\text{succ}}$=1.0 whenever the environment does not expose a separate success bit.

# F TRAINING OBJECTIVES

The main text defines group advantages (Eq. eq. 1), the GRPO surrogate (Eq. eq. 6), gated OPD (Eq. eq. 7), and their sum (Eq. eq. 8). Below we give the Stage I Analyzer loss explicitly and expand Stage II objectives to token indices, including the KL regularizer omitted from the main display.

**Analyzer SFT.** Stage I (Sec. 3.1) trains the shared checkpoint to read a failed rollout without environment access: input $x_\tau = (q, C(\tau), a^{\mathrm{txt}}_{0:T-1})$ matches Eq. eq. 4, and the label is the solver-supervised triplet $z_\tau = (t^*, m^*, r^*)$. Pivot step $t^*$ and mode $m^*$ are pinned offline as in App. B; optional guidance text $r^*$ is produced by a frozen teacher on records that pass the loose factual filter (App. G). Let $z_{\tau,1:L_\tau}$ denote the tokenized JSON string for that object. We minimize length-normalized autoregressive negative log-likelihood,

$$\mathcal{L}_{\mathrm{sft}}(\theta) = -\mathbb{E}_{(x_\tau, z_\tau)\sim\mathcal{D}_{\mathrm{sft}}}\left[\frac{1}{L_\tau}\sum_{\ell=1}^{L_\tau}\log \pi_\theta\big(z_{\tau,\ell} \mid x_\tau, z_{\tau,<\ell}\big)\right]. \tag{10}$$

The resulting weights $\theta_{\mathrm{sft}}$ initialize Stage II and supply the Analyzer used at training time; at inference only the unprivileged Student is deployed (Sec. 3.1).

**Confidence-gated OPD.** Sec. 3.2 uses the same $\pi_\theta$ as a detached Teacher: privileged log-probs $\ell^{\mathrm{tea}}_{q,n,t,\ell}$ condition on $\tilde{h}_t = (p, h_t)$ from Eq. eq. 5, while $\ell^{\mathrm{stu}}_{q,n,t,\ell}$ uses the unprivileged history $h_t$ only. Rollout actions are not resampled; both branches re-score the tokens already taken on the failed trajectory. For each failed rollout we define the hindsight gap and gate

$$\Delta_{q,n,t,\ell} = \mathrm{sg}\big[\ell^{\mathrm{tea}}_{q,n,t,\ell} - \ell^{\mathrm{stu}}_{q,n,t,\ell}\big], \qquad g_{q,n,t,\ell} = \sigma(\beta_{\mathrm{opd}}\Delta_{q,n,t,\ell}), \tag{11}$$

matching the main-text $\delta_t$ and $g_t$ in Eq. eq. 7 ($\lambda_{\mathrm{opd}}$ and $\beta_{\mathrm{opd}}$ in Tab. 12). The failed-trajectory loss aggregates over all valid action tokens on $R(\tau_q^{(n)}) < R_{\mathrm{succ}}$:

$$\mathcal{L}_{\mathrm{opd}}(\theta) = \mathbb{E}_{q,n,t,\ell}\left[\mathbb{I}\Big[R(\tau_q^{(n)}) < R_{\mathrm{succ}}\Big]\, m_{\mathrm{act},q,n,t,\ell}\, g_{q,n,t,\ell}\left(\mathrm{sg}\big[\ell^{\mathrm{tea}}_{q,n,t,\ell}\big] - \ell^{\mathrm{stu}}_{q,n,t,\ell}\right)\right]. \tag{12}$$

Stop-gradient on Teacher logits and on $\Delta$ leaves the update on Student parameters:

$$\nabla_\theta\mathcal{L}_{\mathrm{opd}} = -\mathbb{E}_{q,n,t,\ell}\left[\mathbb{I}\Big[R(\tau_q^{(n)}) < R_{\mathrm{succ}}\Big]\, m_{\mathrm{act},q,n,t,\ell}\, g_{q,n,t,\ell}\, \nabla_\theta\ell^{\mathrm{stu}}_{q,n,t,\ell}\right]. \tag{13}$$

When a GRPO group has $\sigma_q$=0, $A^{\mathrm{rl}}_{q,n}$ vanishes for every token but this term can still train the Student on failed rollouts (Sec. 3.2).

**SFT and RL optimization.** Analyzer SFT optimizes Eq. eq. 10 with AdamW at learning rate $2\times10^{-6}$ (batch size 8, maximum sequence length 8192, three epochs per environment). Each environment yields its own $\theta_{\mathrm{sft}}$ checkpoint, which initializes Stage II for that task only. All Stage II knobs not listed here (GRPO group size, OPD weights, KL coefficient, batching, and validation counts) match Tab. 12.

# G SFT DATA AND OPTIMIZATION

**Loose factual filter for Analyzer SFT.** Sec. 3.1 trains on accepted failures only; each example must pass a loose factual filter $\nu_\tau$ after the solver pins $(t^*, m^*)$ (App. B). We require parseable Analyzer JSON, $m^* \in \mathcal{M}_{\mathrm{fail}}$, and consistency between the solver replay and the logged trajectory. When a guidance field $r^*$ is present, it must be non-empty and refer to the action at $t^*$. Task-specific checks further drop bad labels: Sokoban `deadlock` unless the replay matches the pinned state; FrozenLake skills that contradict hole-fall or budget markers; Navigation skills that mention an unreached goal. Any failed check removes the record; we do not apply lift-based replay filtering when building $\mathcal{D}_{\mathrm{sft}}$.

**Analyzer SFT data.** Data come from failed rollouts of the raw instruct checkpoint at sampling temperature 1.0 (one dataset per environment). The environment solver supplies $(t^*, m^*)$; a frozen `Qwen3-VL-8B` model writes optional $r^*$ on records that survive $\nu_\tau$. Tab. 11 lists the resulting 90/10 train/validation counts for `Qwen2.5-VL-3B` packs (`Qwen3-VL-2B` follows the same pipeline). Example collages, role prompts, and JSON targets appear in App. H; AdamW and Stage II settings are in Tab. 12.

# H ANALYZER COLLAGES, PROMPTS, AND SFT TARGETS

This section documents training-time prompts and Analyzer supervision visuals omitted from the main text for space. Prompt templates appear first, followed by multi-page collage figures.

**Role prompts by environment.** For each VAGEN task we include three prompt families: the unprivileged Student (task instruction plus short RGB history), the Analyzer (full collage $C(\tau)$ and action transcript $a^{\text{txt}}_{0:T-1}$, JSON output), and the privileged Teacher under M+P or M+P+R (panel $P_{\hat{t}}$ plus mode and optional skill text; see Fig. 3). Vision placeholders use <image>; closed failure-mode lists follow App. B. Ground-truth SFT targets $(t^*, m^*, r^*)$ are built as in App. G.

## Sokoban

Sokoban — Student ($\pi_\theta$, RL rollout)

```
You are an expert agent operating in the Sokoban environment. Your goal is to push all the
boxes onto the target spots. Once all boxes are on the targets, you win!
# Rules You can only push boxes. You can't pull them, so plan ahead to avoid getting
stuck. You can't walk through or push boxes into walls. To avoid traps, do not push boxes
into corners or against walls where they can't be moved again.
# Visual Elements in the Image: Character: A small, green alien-like figure with two
antennae and black eyes. It represents you. Box: A yellow crate marked with an orange
"X" across its front. It is the box you need to push. Target: A black tile outlined in
red, with a small red diamond shape in the center. It marks the destination where a box
should be pushed.
# Current Step Your current observation is shown in the image: <image> Your admissible
actions are ["up", "down", "left", "right"].
Now it's your turn to make a move (choose ONE action only for the current step). You
should first reason step-by-step about the current situation | observe the positions
of boxes and targets, plan a path to push a box toward a target, and avoid traps like
corners or walls. This reasoning process MUST be enclosed within <redacted_thinking>
</redacted_thinking> tags. Once you've finished your reasoning, you should choose an
admissible action for current step and present it within <action> </action> tags.
```

Sokoban — Analyzer (failed rollout, SFT/online)

```
Screenshot collage (<image>): 3x3 grid in row-major order, each panel labeled by original
0-based step index (0..8). Match the panel label with that step in the trajectory text
below.
Analyze this failed agent episode from the screenshot collage and the compact action log.
Return ONLY valid JSON.
You need to complete three fields: 1. pivot_step: the 0-based index of the first step
after which the remaining step budget cannot solve the puzzle. 2. failure_mode: "timeout"
if the puzzle is still solvable with more steps, or "deadlock" if the box is irreversibly
stuck. 3. failure_reason: extract the failed trajectory into avoidance rules (the core
mistake and warning signs), then one short imperative sentence the policy can act on at
pivot_step, not a retrospective explanation. Put both parts in this one string.
Important constraints: - Use the screenshot collage together with the compact action log;
do not ignore the images. - Step indices are 0-based integers. - pivot_step MUST be an
integer in [0, 8] matching a labeled collage panel. - Do not copy an example index;pick
the true fatal step for THIS trajectory. - Prefer a valid box-push that made the remaining
budget insufficient. Invalid / no-op actions (action=invalid or moved=false) are usually
not the pivot. - failure_mode must be exactly timeout or deadlock. - Return only these
top-level fields: pivot_step, failure_mode, failure_reason.
Return format: { "pivot_step": <integer>, "failure_mode": "timeout", "failure_reason":
"Avoid repeating the core mistake. At the pivot, take the productive action instead of
that compact-log action." }
Episode context: - Task description: Push the box onto the target without trapping
it against a wall or corner. - episode_success: failure - Number of steps: 9 - Valid
pivot_step range: [0, 8] - Compact action log: step 0: action=left moved=true step 1:
action=right moved=true step 2: action=right moved=false step 3: action=left moved=true
step 4: action=left moved=true step 5: action=up moved=false step 6: action=right
moved=true step 7: action=left moved=true step 8: action=down moved=true
```

Sokoban — Teacher (M+P+R{}, privileged scoring)

```
You are an expert agent operating in the Sokoban environment. Your goal is to push all the
boxes onto the target spots. Once all boxes are on the targets, you win!
# Rules You can only push boxes. You can't pull them, so plan ahead to avoid getting
stuck. You can't walk through or push boxes into walls. To avoid traps, do not push boxes
into corners or against walls where they can't be moved again.
# Visual Elements in the Image: Character: A small, green alien-like figure with two
antennae and black eyes. It represents you. Box: A yellow crate marked with an orange
```

```
"X" across its front.  It is the box you need to push.  Target:  A black tile outlined in
red, with a small red diamond shape in the center.  It marks the destination where a box
should be pushed.
# Current Step Your current observation is shown in the image:  <image>
Hindsight 3-panel around the predicted failure pivot (not available when selecting the
current action):  <image>
Your admissible actions are ["up", "down", "left", "right"].
**Episode-Level Skill** Refer to this episode-level skill when deciding what action to
take in the current episode:  [Failure mode:  deadlock Refer to this episode-level skill
when deciding what action to take:  Avoid pushing the box into a corner where it becomes
trapped.  At step 4, do not push up again;instead, move right to create space.]
Now it's your turn to make a move (choose ONE action only for the current step).  You
should first reason step-by-step about the current situation | observe the positions
of boxes and targets, plan a path to push a box toward a target, and avoid traps like
corners or walls.  This reasoning process MUST be enclosed within <redacted_thinking>
</redacted_thinking> tags.  Once you've finished your reasoning, you should choose an
admissible action for current step and present it within <action> </action> tags.
```

## FrozenLake

FrozenLake — Student ($\pi_\theta$, RL rollout)

```
You are an expert agent operating in the FrozenLake environment.  Reach the goal (G)
without falling into holes (O).
# Rules Move carefully on the ice.  Holes end the episode.  The goal tile wins.
# Current Step Your current observation is shown in the image:  <image> Your admissible
actions are ["up", "down", "left", "right"].
Now it's your turn to make a move (choose ONE action only for the current step).  You
should first reason step-by-step about the current situation.  This reasoning process MUST
be enclosed within <redacted_thinking> </redacted_thinking> tags.  Once you've finished
your reasoning, you should choose an admissible action for current step and present it
within <action> </action> tags.
```

FrozenLake — Analyzer (failed rollout, SFT/online)

```
Screenshot collage (<image>):  2x2 grid in row-major order, each panel labeled by original
0-based step index (0..2).  Match the panel label with that step in the trajectory text
below.
Analyze this failed agent episode from the screenshot collage and the compact action log.
Return ONLY valid JSON.
You need to complete three fields:  1.  pivot_step:  the 0-based index of the first step
after which the remaining step budget cannot reach the goal.  2.  failure_mode:  "timeout"
if the agent is still on safe ice and more steps could reach the goal, or "deadlock" if the
agent fell into a hole (irreversible).  3.  failure_reason:  extract the failed trajectory
into avoidance rules (the core mistake and warning signs), then one short imperative
sentence the policy can act on at pivot_step, not a retrospective explanation.  Put both
parts in this one string.
Important constraints:  - Use the screenshot collage together with the compact action log;
do not ignore the images.  - Step indices are 0-based integers.  - pivot_step MUST be an
integer in [0, 2] matching a labeled collage panel.  - Do not copy an example index;pick
the true fatal step for THIS trajectory.  - Prefer the step that walked into a hole, or a
valid move that made the remaining budget insufficient.  Invalid / empty / no-op actions
(action=invalid or moved=false) are usually not the pivot, unless the episode ends on that
empty action.  - failure_mode must be exactly timeout or deadlock.  - Return only these
top-level fields:  pivot_step, failure_mode, failure_reason.
Return format:  { "pivot_step":  <integer>, "failure_mode":  "timeout", "failure_reason":
"Avoid repeating the core mistake.  At the pivot, take the productive action instead of
that compact-log action." }
Episode context:  - Task description:  Navigate from the start to the goal on the frozen
lake without falling into holes.  - episode_success:  failure - Number of steps:  3 -
Valid pivot_step range:  [0, 2] - Compact action log:  step 0:  action=left moved=true
remaining_turns=8 step 1:  action=right moved=true remaining_turns=7 step 2:  action=right
moved=true remaining_turns=6
```

FrozenLake — Teacher (M+P+R{}, privileged scoring)

```
You are an expert agent operating in the FrozenLake environment.  Reach the goal (G)
without falling into holes (O).
# Rules Move carefully on the ice.  Holes end the episode.  The goal tile wins.
# Current Step Your current observation is shown in the image:  <image>
Hindsight 3-panel around the predicted failure pivot (not available when selecting the
current action):  <image>
Your admissible actions are ["up", "down", "left", "right"].
**Episode-Level Skill** Refer to this episode-level skill when deciding what action to take
in the current episode:  [Failure mode:  deadlock Refer to this episode-level skill when
```

```
deciding what action to take: Avoid stepping onto hole tiles when a safe path exists. At
step 1, move right along safe ice instead of up into the hole.]
Now it's your turn to make a move (choose ONE action only for the current step). You
should first reason step-by-step about the current situation. This reasoning process MUST
be enclosed within <redacted_thinking> </redacted_thinking> tags. Once you've finished
your reasoning, you should choose an admissible action for current step and present it
within <action> </action> tags.
```

## Navigation

Navigation — Student ($\pi_\theta$, RL rollout)

```
You are an embodied agent navigating an AI2-THOR indoor environment from egocentric RGB.
# Instruction navigate to the Bread in the room and be as close as possible to it
# Actions (choose ONE) - moveahead / moveback / moveright / moveleft: translate 0.5m -
rotateright / rotateleft: yaw 90 degrees - lookup / lookdown: pitch 30 degrees
Prior steps: 3. Recent actions (2): step 2: moveahead;step 3: rotateright Now at step
4. Current egocentric view: <image>
Reason in <redacted_thinking> </redacted_thinking>, then put ONE action in <action>
</action>.
```

Navigation — Analyzer (failed rollout, SFT/online)

```
Screenshot collage (<image>): 5x5 grid in row-major order, each panel labeled by original
0-based step index (0..24). Match the panel label with that step in the trajectory text
below.
Analyze this failed agent episode from the screenshot collage and the compact action log.
Return ONLY valid JSON.
You need to complete three fields: 1. pivot_step: the 0-based index of the earliest
structural mistake | a terminal blocked translation, the start of a no-progress loop, a
trailing invalid suffix, or else the first step whose remaining translations cannot cover
the Euclidean distance. 2. failure_mode: exactly one of "budget", "format", "blocked",
"stagnation". budget = remaining translations cannot cover remaining distance. format
= empty/invalid action exhausted slack. blocked = a move was rejected by a wall/door.
stagnation = recent moves did not reduce distance. 3. failure_reason: extract the failed
trajectory into avoidance rules (the core mistake and warning signs), then one short
imperative sentence the policy can act on at pivot_step, not a retrospective explanation.
Put both parts in this one string.
Important constraints: - Use the screenshot collage together with the compact action log;
do not ignore the images. - Step indices are 0-based integers. - pivot_step MUST be an
integer in [0, 24] matching a labeled collage panel. - Do not copy an example index;pick
the true fatal step for THIS trajectory. - Prefer the earliest real navigation action that
matches that mode. Rotate/look are valid and may be a stagnation pivot. Empty/invalid
is the format pivot when it is a trailing suffix. - failure_mode must be exactly one of:
budget, format, blocked, stagnation. - Return only these top-level fields: pivot_step,
failure_mode, failure_reason.
Return format: { "pivot_step": <integer>, "failure_mode": "budget", "failure_reason":
"Avoid repeating the core mistake. At the pivot, take the productive action instead of
that compact-log action." }
Episode context: - Task description: navigate to the Bread in the room and be as
close as possible to it - episode_success: failure - Number of steps: 25 - Valid
pivot_step range: [0, 24] - Compact action log: step 0: action=moveright moved=false
last_action_success=false action_is_valid=true dist=2.87 remaining_turns=24 step 1:
action=moveright moved=false last_action_success=false action_is_valid=false dist=2.87
remaining_turns=23 step 2: action=moveahead moved=true last_action_success=true
action_is_valid=true dist=2.39 remaining_turns=22 step 3: action=moveleft moved=true
last_action_success=true action_is_valid=true dist=2.56 remaining_turns=21 step 4:
action=rotateright moved=false last_action_success=false action_is_valid=false dist=2.56
remaining_turns=20 step 5: action=moveahead moved=false last_action_success=false
action_is_valid=false dist=2.39 remaining_turns=19 step 6: action=moveright moved=true
last_action_success=true action_is_valid=true dist=2.87 remaining_turns=18 step 7:
action=moveright moved=true last_action_success=true action_is_valid=true dist=3.36
remaining_turns=17 step 8: action=moveleft moved=true last_action_success=true
action_is_valid=true dist=2.87 remaining_turns=16 step 9: action=rotateleft moved=false
last_action_success=true action_is_valid=true dist=2.87 remaining_turns=15 step 10:
action=moveleft moved=false last_action_success=false action_is_valid=false dist=3.02
remaining_turns=14 step 11: action=moveahead moved=true last_action_success=true
action_is_valid=true dist=2.56 remaining_turns=13 step 12: action=invalid moved=false
last_action_success=false action_is_valid=true remaining_turns=12 step 13: action=invalid
moved=false last_action_success=false action_is_valid=true remaining_turns=11 step
14: action=invalid moved=false last_action_success=false action_is_valid=true
remaining_turns=10 step 15: action=moveleft moved=true last_action_success=true
action_is_valid=true dist=2.82 remaining_turns=9 step 16: action=invalid moved=false
last_action_success=false action_is_valid=false remaining_turns=8 step 17: action=invalid
moved=false last_action_success=false action_is_valid=false remaining_turns=7 step 18:
action=moveleft moved=false last_action_success=false action_is_valid=true dist=2.82
```

```
remaining_turns=6 step 19: action=moveleft moved=false last_action_success=false
action_is_valid=true dist=2.82 remaining_turns=5 step 20: action=invalid moved=false
last_action_success=false action_is_valid=false remaining_turns=4 step 21: action=moveahead
moved=false last_action_success=false action_is_valid=true dist=2.82 remaining_turns=3 step
22: action=rotateright moved=false last_action_success=true action_is_valid=true dist=2.82
remaining_turns=2 step 23: action=rotateright moved=false last_action_success=true
action_is_valid=true dist=2.82 remaining_turns=1 step 24: action=moveahead moved=false
last_action_success=false action_is_valid=true dist=2.82 remaining_turns=0
```

Navigation — Teacher (M+P+R{}, privileged scoring)

```
You are an embodied agent navigating an AI2-THOR indoor environment from egocentric RGB.
# Instruction navigate to the Bread in the room and be as close as possible to it
# Actions (choose ONE) - moveahead / moveback / moveright / moveleft: translate 0.5m -
rotateright / rotateleft: yaw 90 degrees - lookup / lookdown: pitch 30 degrees
Prior steps: 3. Recent actions (2): step 2: moveahead;step 3: rotateright Now at step
4. Current egocentric view: <image>
Reason in <redacted_thinking> </redacted_thinking>, then put ONE action in <action>
</action>.
**Episode-Level Skill** Refer to this episode-level skill when deciding what action to take
in the current episode: [Failure mode: blocked Refer to this episode-level skill when
deciding what action to take: Avoid repeated moveahead into a closed door. At step 22,
rotate or sidestep before advancing again.]
Hindsight 3-panel around the predicted failure pivot (not available when selecting the
current action): <image>
```

## PrimitiveSkill

PrimitiveSkill — Student ($\pi_\theta$, RL rollout)

```
You are an AI assistant controlling a Franka Emika robot arm. Your goal is to understand
human instructions and translate them into a sequence of executable actions for the robot,
based on visual input and the instruction.
Action Space Guide You can command the robot using the following actions:
1. pick(x, y, z) # To grasp an object located at position(x,y,z) in the robot's workspace.
2. place(x, y, z) # To place the object currently held by the robot's gripper at the
target position (x,y,z). 3. push(x1, y1, z1, x2, y2, z2) # To push an object from
position (x1,y1,z1) to (x2,y2,z2).
Hints: 1. The coordinates (x, y, z) are in millimeters and are all integers. 2. Please
ensure that the coordinates are within the workspace limits. 3. The position is the
center of the object, when you place, please consider the volume of the object. It's
always fine to set z much higher when placing an item. 4. We will provide the object
positions to you, but you need to match them to the object in the image by yourself.
You're facing toward the negative x-axis, and the negative y-axis is to your left, the
positive y-axis is to your right, and the positive z-axis is up.
Examples: round1: image1 Human Instruction: Put red cube on
green cube and yellow cube on left target Object positions:
[(62,-55,20),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)] Reasoning: I can see
from the picture that the red cube is on my left and green cube is on my right and near
me. Since I'm looking toward the negative x axis, and negative y-axis is to my left,
(62,-55,20) would be the position of the red cube, (75,33,20) would be the position of
the green cube and (-44,100,20) is the position of the yellow cube. Also the (100,-43,0)
would be the position of the left target, and (100,43,0) would be the porition of the right
target. I need to pick up red cube first and place it on the green cube, when placing,
I should set z much higher. Anwer: pick(62,-55,20)|place(75,33,50) round2: image2 Human
Instruction: Put red cube on green cube and yellow cube on left target Object positions:
[(75,33,50),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)] Reasoning: Now the red cube is
on the green cube, so I need to pick up the yellow cube and place it on the left target.
Anwer: pick(-44,100,20)|place(100,-43,50)
You can take up to 2 action(s) at a time, separated by |. You should first give
your thought process, and then your answer. Your response should be in the format
of: <redacted_thinking>...</redacted_thinking><answer>...</answer> e.g.
<redacted_thinking>I need to pick the red_cube_pos at (10,20,30) and place it on
the green_block_pos at (50,60,40).</redacted_thinking><answer>pick(10,20,30)|
place(50,60,70)</answer>
Prior steps: 3. Recent actions (2): step 2: pick(62,-55,20)|place(75,33,50);step 3:
pick(-44,100,20) Now at step 4.
[Current Observation]: <image> Human Instruction: Please align the cubes in the y-axis,
which means the x-coordinates of both cubes should be 0 (+-10mm) x_workspace_limit:
[-100, 100] y_workspace_limit: [-100, 100] z_workspace_limit: [0, 120] Object positions:
[(62,-55,20),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)] Other information: (none)
Decide your next action(s).
```

PrimitiveSkill — Analyzer (failed rollout, SFT/online)

```
Screenshot collage (<image>): 3x2 grid in row-major order, each panel labeled by original
0-based step index (0..5). Match the panel label with that step in the trajectory text
below.
Analyze this failed agent episode from the screenshot collage and the compact action log.
Return ONLY valid JSON.
You need to complete three fields: 1. pivot_step: the 0-based index of the first step
after which the remaining turns cannot finish the remaining task stages. 2. failure_mode:
exactly one of "timeout", "format", "wrong_target", "order_error". timeout = remaining
stages exceed remaining turns. format = no legal pick/place/push. wrong_target = a legal
action missed the object or target. order_error = place/push before a required pick. 3.
failure_reason: extract the failed trajectory into avoidance rules (the core mistake and
warning signs), then one short imperative sentence the policy can act on at pivot_step, not
a retrospective explanation. Put both parts in this one string.
Important constraints: - Use the screenshot collage together with the compact action log;
do not ignore the images. - Step indices are 0-based integers. - pivot_step MUST be an
integer in [0, 5] matching a labeled collage panel. - Do not copy an example index;pick
the true fatal step for THIS trajectory. - Prefer a real pick/place/push that made the
remaining stage budget insufficient. Empty/invalid is usually not the pivot unless that
empty action exhausted the last slack. - failure_mode must be exactly one of: timeout,
format, wrong_target, order_error. - Return only these top-level fields: pivot_step,
failure_mode, failure_reason.
Return format: { "pivot_step": <integer>, "failure_mode": "timeout", "failure_reason":
"Avoid repeating the core mistake. At the pivot, take the productive action instead of
that compact-log action." }
Episode context: - Task description: Please align the cubes in the y-axis, which
means the x-coordinates of both cubes should be 0 (+-10mm) - episode_success: failure
- Number of steps: 6 - Valid pivot_step range: [0, 5] - Compact action log: step
0: action=empty legal=false remaining_stages=2 remaining_turns=5 step 1: action=empty
legal=false remaining_stages=2 remaining_turns=4 step 2: action=empty legal=false
remaining_stages=2 remaining_turns=3 step 3: action=empty legal=false remaining_stages=2
remaining_turns=2 step 4: action=empty legal=false remaining_stages=2 remaining_turns=1
step 5: action=pick(107,64,20)|place(0,-185,120) legal=true remaining_stages=1
remaining_turns=0
```

PrimitiveSkill — Teacher (M+P+R{}, privileged scoring)

```
You are an AI assistant controlling a Franka Emika robot arm. Your goal is to understand
human instructions and translate them into a sequence of executable actions for the robot,
based on visual input and the instruction.
Action Space Guide You can command the robot using the following actions:
1. pick(x, y, z) # To grasp an object located at position(x,y,z) in the robot's workspace.
2. place(x, y, z) # To place the object currently held by the robot's gripper at the
target position (x,y,z). 3. push(x1, y1, z1, x2, y2, z2) # To push an object from
position (x1,y1,z1) to (x2,y2,z2).
Hints: 1. The coordinates (x, y, z) are in millimeters and are all integers. 2. Please
ensure that the coordinates are within the workspace limits. 3. The position is the
center of the object, when you place, please consider the volume of the object. It's
always fine to set z much higher when placing an item. 4. We will provide the object
positions to you, but you need to match them to the object in the image by yourself.
You're facing toward the negative x-axis, and the negative y-axis is to your left, the
positive y-axis is to your right, and the positive z-axis is up.
Examples: round1: image1 Human Instruction: Put red cube on
green cube and yellow cube on left target Object positions:
[(62,-55,20),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)] Reasoning: I can see
from the picture that the red cube is on my left and green cube is on my right and near
me. Since I'm looking toward the negative x axis, and negative y-axis is to my left,
(62,-55,20) would be the position of the red cube, (75,33,20) would be the position of
the green cube and (-44,100,20) is the position of the yellow cube. Also the (100,-43,0)
would be the position of the left target, and (100,43,0) would be the porition of the right
target. I need to pick up red cube first and place it on the green cube, when placing,
I should set z much higher. Anwer: pick(62,-55,20)|place(75,33,50) round2: image2 Human
Instruction: Put red cube on green cube and yellow cube on left target Object positions:
[(75,33,50),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)] Reasoning: Now the red cube is
on the green cube, so I need to pick up the yellow cube and place it on the left target.
Anwer: pick(-44,100,20)|place(100,-43,50)
You can take up to 2 action(s) at a time, separated by |. You should first give
your thought process, and then your answer. Your response should be in the format
of: <redacted_thinking>...</redacted_thinking><answer>...</answer> e.g.
<redacted_thinking>I need to pick the red_cube_pos at (10,20,30) and place it on
the green_block_pos at (50,60,40).</redacted_thinking><answer>pick(10,20,30)|
place(50,60,70)</answer>
Prior steps: 3. Recent actions (2): step 2: pick(62,-55,20)|place(75,33,50);step 3:
pick(-44,100,20) Now at step 4.
[Current Observation]: <image> Human Instruction: Please align the cubes in the y-axis,
which means the x-coordinates of both cubes should be 0 (+-10mm) x_workspace_limit:
```

```
[-100, 100] y_workspace_limit: [-100, 100] z_workspace_limit: [0, 120] Object positions:
[(62,-55,20),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)] Other information: (none)
**Episode-Level Skill** Refer to this episode-level skill when deciding what action to take
in the current episode: [Failure mode: wrong_target Refer to this episode-level skill when
deciding what action to take: Avoid placing on the wrong cube. At step 4, pick the yellow
cube and place on the left target.]
Hindsight 3-panel around the predicted failure pivot (not available when selecting the
current action): <image>
Decide your next action(s).
```

## SVG Reconstruction

SVG Reconstruction — Student ($\pi_\theta$, RL rollout)

```
You are a precise SVG code generator.
SVG Quick Guide Goal: Transform the provided image into precise SVG code that replicates
the image.
Process: 1. First analyze the image carefully, identifying distinct visual elements 2.
Identify colors, dimensions, positions, and relationships between elements 3. Generate
accurate SVG code that reproduces the image;you can use <path> for better shape
Rewards: - Overall visual similarity: +5.0 - Structural accuracy: +10.0
Example: <redacted_thinking>I can see the image contains a red circle and a
blue rectangle. The circle is positioned at the top-left, while the rectangle
is at the bottom-right.</redacted_thinking> <answer><svg viewBox="0 0 100 100"
xmlns="http://www.w3.org/2000/svg"> <circle cx="25" cy="25" r="15" fill="red" /> <rect
x="60" y="60" width="30" height="20" fill="blue" /> </svg></answer>
You can take up to 1 action(s) at a time, separated by ,. You should first give
your thought process, and then your answer. Your response should be in the
format of: <redacted_thinking>...</redacted_thinking><answer>...</answer>
e.g. <redacted_thinking>I can see the image contains a red circle and a blue
rectangle. The circle is positioned at the top-left, while the rectangle is
at the bottom-right.</redacted_thinking> <answer><svg viewBox="0 0 100 100"
xmlns="http://www.w3.org/2000/svg"> <circle cx="25" cy="25" r="15" fill="red" /> <rect
x="60" y="60" width="30" height="20" fill="blue" /> </svg></answer>
[Initial Observation]: Your target image is shown here: <image> Please carefully observe
the image, and generate SVG code that reproduces it as accurately as possible. Decide on
your SVG code.
```

SVG Reconstruction — Analyzer (failed rollout, SFT/online)

```
Screenshot collage (<image>): 2x2 grid in row-major order. The panel labeled target
is the ground-truth icon;panels labeled step k are the rasterized SVG submitted at that
0-based step. Match step k with the compact action log.
Analyze this failed agent episode from the screenshot collage and the compact action log.
Return ONLY valid JSON.
You need to complete three fields: 1. pivot_step: the 0-based index of the terminating
submission (the drawing that was scored, or the empty action that ended the episode).
2. failure_mode: "timeout" if the task is still solvable but the remaining budget is
not enough, or "deadlock" if the failure is irreversible. Prefer timeout;pin the last
submission (empty SVG stays action=empty in the log). 3. failure_reason: extract the
failed trajectory into avoidance rules (the core mistake and warning signs), then one short
imperative sentence the policy can act on at pivot_step, not a retrospective explanation.
Put both parts in this one string.
Important constraints: - Use the screenshot collage together with the compact action log;
do not ignore the images. - Step indices are 0-based integers. - pivot_step MUST be an
integer in [0, 1] matching a labeled collage panel. - Do not copy an example index;pick
the true fatal step for THIS trajectory. - Empty submissions are still a timeout pivot;do
not skip them. For a valid SVG, prefer the last submission. - failure_mode must be exactly
timeout or deadlock. - Return only these top-level fields: pivot_step, failure_mode,
failure_reason.
Return format: { "pivot_step": <integer>, "failure_mode": "timeout", "failure_reason":
"Avoid repeating the core mistake. At the pivot, take the productive action instead of
that compact-log action." }
Episode context: - Task description: Reconstruct the target image as SVG code. -
episode_success: failure - Number of steps: 2 - Valid pivot_step range: [0, 1] - Compact
action log: step 0: action=svg valid_svg=true remaining_turns=1 average_score=0.50 step 1:
action=svg valid_svg=true remaining_turns=0 average_score=0.50
```

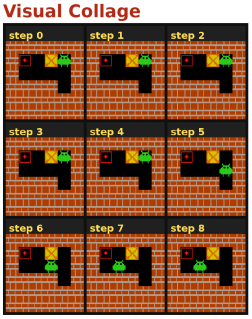
SVG Reconstruction — Teacher (M+P+R{}, privileged scoring)

```
You are a precise SVG code generator.
SVG Quick Guide Goal: Transform the provided image into precise SVG code that replicates
the image.
```

```
Process:  1.  First analyze the image carefully, identifying distinct visual elements 2.
Identify colors, dimensions, positions, and relationships between elements 3.  Generate
accurate SVG code that reproduces the image;you can use <path> for better shape
Rewards:  - Overall visual similarity:  +5.0 - Structural accuracy:  +10.0
Example:  <redacted_thinking>I can see the image contains a red circle and a
blue rectangle.  The circle is positioned at the top-left, while the rectangle
is at the bottom-right.</redacted_thinking> <answer><svg viewBox="0 0 100 100"
xmlns="http://www.w3.org/2000/svg"> <circle cx="25" cy="25" r="15" fill="red" /> <rect
x="60" y="60" width="30" height="20" fill="blue" /> </svg></answer>
You can take up to 1 action(s) at a time, separated by ,.  You should first give
your thought process, and then your answer.  Your response should be in the
format of:  <redacted_thinking>...</redacted_thinking><answer>...</answer>
e.g.  <redacted_thinking>I can see the image contains a red circle and a blue
rectangle.  The circle is positioned at the top-left, while the rectangle is
at the bottom-right.</redacted_thinking> <answer><svg viewBox="0 0 100 100"
xmlns="http://www.w3.org/2000/svg"> <circle cx="25" cy="25" r="15" fill="red" /> <rect
x="60" y="60" width="30" height="20" fill="blue" /> </svg></answer>
[Initial Observation]:  Your target image is shown here:  <image> Please carefully observe
the image, and generate SVG code that reproduces it as accurately as possible.
**Episode-Level Skill** Refer to this episode-level skill when deciding what action to take
in the current episode:  [Failure mode:  timeout Refer to this episode-level skill when
deciding what action to take:  Avoid omitting the main icon strokes.  At step 1, redraw the
sun disk and rays before submitting.]
Hindsight 3-panel around the predicted failure pivot (not available when selecting the
current action):  <image>
Decide on your SVG code.
```

**Analyzer SFT collages.** Figure 8 summarizes SFT-stage input–output examples across environments (continued on subsequent pages as noted). Each figure is environment-specific: columns 1–2 are the visual trajectory collage and compact action log; column 3 is the JSON supervision target (Navigation logs truncate with "…" around the pivot step).

**Visual Collage**



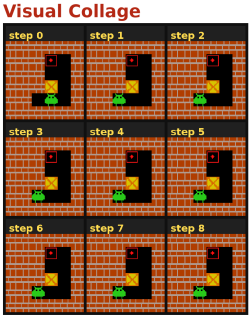

**Compact action log**

```
step 0: action=right moved=false
step 1: action=up moved=false
step 2: action=invalid
moved=false
step 3: action=invalid
moved=false
step 4: action=down moved=true
step 5: action=left moved=true
step 6: action=left moved=true
step 7: action=invalid
moved=false
step 8: action=down moved=false
```

**SFT JSON DATA**

```
{
 "pivot_step": 5,
 "failure_mode":
 "timeout",
 "failure_reason":
 "At step 5, action=left moved=true
 placed the box against a wall,
 wasting one step of budget."
}
```

**Visual Collage**



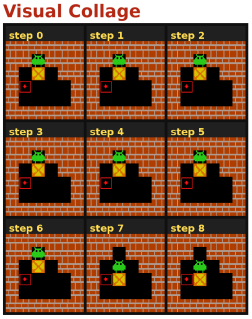

**Compact action log**

```
step 0: action=left moved=true
step 1: action=down moved=false
step 2: action=down moved=false
step 3: action=down moved=false
step 4: action=up moved=false
step 5: action=left moved=false
step 6: action=invalid
moved=false
step 7: action=up moved=false
step 8: action=right moved=true
```

**SFT JSON DATA**

```
{
 "pivot_step": 6,
 "failure_mode":
 "timeout",
 "failure_reason":
 "At step 6, the agent performed an
 invalid action that wasted a step
 budget without moving the box,
 leaving insufficient steps to
 complete the task."
}
```

**Visual Collage**



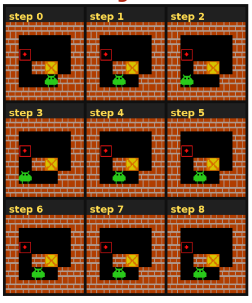

**Compact action log**

```
step 0: action=right moved=false
step 1: action=right moved=false
step 2: action=right moved=false
step 3: action=up moved=false
step 4: action=up moved=false
step 5: action=right moved=false
step 6: action=down moved=true
step 7: action=invalid
moved=false
step 8: action=right moved=true
```

**SFT JSON DATA**

```
{
 "pivot_step": 5,
 "failure_mode":
 "timeout",
 "failure_reason":
 "At step 5, action=right did not move
 the box, wasting a step budget while
 the agent was still far from the
 target."
}
```

**Visual Collage**




**Compact action log**

```
step 0: action=left moved=true
step 1: action=left moved=true
step 2: action=down moved=false
step 3: action=right moved=true
step 4: action=down moved=false
step 5: action=down moved=false
step 6: action=invalid
moved=false
step 7: action=invalid
moved=false
step 8: action=down moved=false
```

**SFT JSON DATA**

```
{
 "pivot_step": 2,
 "failure_mode":
 "timeout",
 "failure_reason":
 "At step 2, action=down did not move
 the box, wasting a step and leaving
 the agent with insufficient budget to
 complete the task."
}
```

Figure 6: Sokoban. SFT-stage input–output examples for this environment. Columns 1–2 are the visual trajectory collage and compact action log; column 3 is the JSON supervision target.

**Visual Collage**




**Compact action log**

```
step 0: action=down moved=true
remaining_turns=8
```

**SFT JSON DATA**

```
{
 "pivot_step": 0,
 "failure_mode":
 "deadlock",
 "failure_reason":
 "Avoid moving down from the starting
 position when surrounded by holes; it
 leads to immediate fall. At step 0,
 do not move down."
}
```

**Visual Collage**



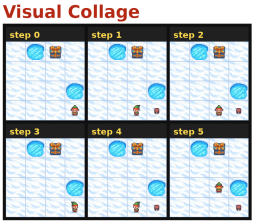

**Compact action log**

```
step 0: action=right moved=true
remaining_turns=8
step 1: action=invalid
moved=false remaining_turns=7
step 2: action=left moved=true
remaining_turns=6
step 3: action=down moved=true
remaining_turns=5
```

**SFT JSON DATA**

```
{
 "pivot_step": 3,
 "failure_mode":
 "deadlock",
 "failure_reason":
 "Avoid moving down when already
 adjacent to the goal; instead, move
 right to claim the goal. At step 3,
 do not move down."
}
```

**Visual Collage**



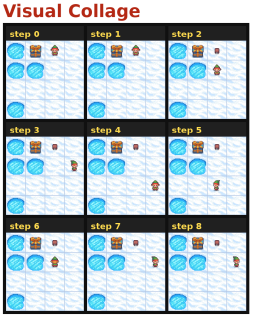

**Compact action log**

```
step 0: action=left moved=true
remaining_turns=8
step 1: action=down moved=false
remaining_turns=7
step 2: action=right moved=true
remaining_turns=6
step 3: action=left moved=true
remaining_turns=5
step 4: action=up moved=true
remaining_turns=4
step 5: action=right moved=true
remaining_turns=3
```

**SFT JSON DATA**

```
{
 "pivot_step": 5,
 "failure_mode":
 "deadlock",
 "failure_reason":
 "Avoid moving right when adjacent to
 a hole and the goal is reachable by
 other means. At step 5, do not move
 right; move up instead."
}
```

**Visual Collage**




**Compact action log**

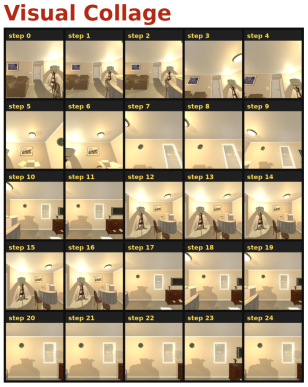
```
step 0: action=invalid
moved=false remaining_turns=8
step 1: action=down moved=true
remaining_turns=7
step 2: action=right moved=true
remaining_turns=6
step 3: action=down moved=true
remaining_turns=5
step 4: action=left moved=true
remaining_turns=4
step 5: action=up moved=true
remaining_turns=3
step 6: action=right moved=true
remaining_turns=2
step 7: action=right moved=false
remaining_turns=1
step 8: action=invalid
moved=false remaining_turns=0
```

**SFT JSON DATA**

```
{
 "pivot_step": 6,
 "failure_mode":
 "timeout",
 "failure_reason":
 "Avoid moving right when the agent is
 already aligned with the goal and has
 limited turns; instead, move toward
 the goal or wait to preserve turns
 for critical moves."
}
```

Figure 7: FrozenLake. SFT-stage input–output examples for this environment. Columns 1–2 are the visual trajectory collage and compact action log; column 3 is the JSON supervision target.

### Visual Collage




### Compact action log

```
step 0: action=moveback
moved=true
last_action_success=true
action_is_valid=true dist=3.10
remaining_turns=24
...
step 20: action=invalid
moved=false
last_action_success=false
action_is_valid=true
remaining_turns=4
step 21: action=invalid
moved=false
last_action_success=false
action_is_valid=true
remaining_turns=3
step 22: action=moveright
moved=true
last_action_success=true
action_is_valid=true dist=2.70
remaining_turns=2 <- pivot step
step 23: action=moveleft
moved=true
last_action_success=true
action_is_valid=true dist=2.32
remaining_turns=1
step 24: action=invalid
moved=false
last_action_success=false
action_is_valid=true
remaining_turns=0
```

### SFT JSON DATA

```
{
 "pivot_step": 22,
 "failure_mode":
 "budget",
 "failure_reason":
 "Avoid moving right when already
 facing away from the FloorLamp; it
 only wastes budget. At step 22, pivot
 left instead of moving right."
}
```

### Visual Collage




### Compact action log

```
step 0: action=invalid
moved=false
last_action_success=false
action_is_valid=true
remaining_turns=24
...
step 21: action=moveahead
moved=true
last_action_success=true
action_is_valid=true dist=2.24
remaining_turns=3
step 22: action=moveahead
moved=false
last_action_success=false
action_is_valid=true dist=2.24
remaining_turns=2
step 23: action=rotateleft
moved=false
last_action_success=true
action_is_valid=true dist=2.24
remaining_turns=1 <- pivot step
step 24: action=moveright
moved=false
last_action_success=false
action_is_valid=true dist=2.24
remaining_turns=0
```

### SFT JSON DATA

```
{
 "pivot_step": 23,
 "failure_mode":
 "budget",
 "failure_reason":
 "Avoid unnecessary rotations that
 don't progress toward the target. At
 step 23, move closer to the storage
 container instead of rotating left."
}
```

**Visual Collage**




**Compact action log**

```
step 0: action=moveleft
moved=true
last_action_success=true
action_is_valid=true dist=3.05
remaining_turns=24
...
step 20: action=lookup
moved=false
last_action_success=false
action_is_valid=true dist=3.01
remaining_turns=4
step 21: action=moveahead
moved=true
last_action_success=true
action_is_valid=true dist=2.99
remaining_turns=3
step 22: action=rotateright
moved=false
last_action_success=true
action_is_valid=true dist=2.99
remaining_turns=2 <- pivot step
step 23: action=moveright
moved=true
last_action_success=true
action_is_valid=true dist=3.01
remaining_turns=1
step 24: action=moveahead
moved=true
last_action_success=true
action_is_valid=true dist=2.52
remaining_turns=0
```

**SFT JSON DATA**

```
{
 "pivot_step": 22,
 "failure_mode":
 "stagnation",
 "failure_reason":
 "Avoid rotating right when already
 facing the GarbageCan; it only delays
 progress. At step 22, move forward
 instead of rotateright."
}
```

**Visual Collage**




**Compact action log**

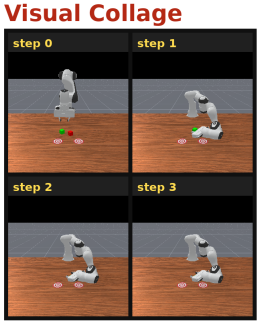
```
step 0: action=rotateright
moved=false
last_action_success=true
action_is_valid=true dist=2.75
remaining_turns=24
...
step 21: action=moveright
moved=false
last_action_success=false
action_is_valid=false dist=2.29
remaining_turns=3
step 22: action=moveahead
moved=false
last_action_success=false
action_is_valid=true dist=2.29
remaining_turns=2
step 23: action=moveahead
moved=false
last_action_success=false
action_is_valid=true dist=2.29
remaining_turns=1 <- pivot step
step 24: action=moveright
moved=false
last_action_success=false
action_is_valid=true dist=2.29
remaining_turns=0
```

**SFT JSON DATA**

```
{
 "pivot_step": 23,
 "failure_mode":
 "blocked",
 "failure_reason":
 "Repeatedly tried moveahead in
 blocked corridors without pivoting or
 backing up. At step 23, pivot or step
 back instead of repeating moveahead."
}
```

Figure 6: Navigation. SFT-stage input–output examples for this environment. Columns 1–2 are the visual trajectory collage and compact action log; column 3 is the JSON supervision target.

**Visual Collage**



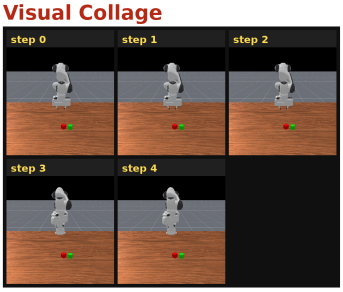

**Compact action log**

```
step 0:
action=pick(-73,-69,20)|push(-36
,23,20,10,0,10)
legal=true remaining_stages=3
remaining_turns=5
step 1:
action=pick(80,-100,0)|place(100
,-150,50)
legal=true remaining_stages=3
remaining_turns=4
step 2:
action=pick(80,-100,0)|place(100
,-100,50)
legal=true remaining_stages=3
remaining_turns=3
step 3:
action=pick(80,-100,20)|pick(80,
100,20)
legal=true remaining_stages=3
remaining_turns=2
```

**SFT JSON DATA**

```
{
 "pivot_step": 3,
 "failure_mode":
 "timeout",
 "failure_reason":
 "Avoid picking up the wrong cube
 again. At step 3, do not repeat
 pick(80,-100,20)|pick(80,100,20)."
}
```

**Visual Collage**



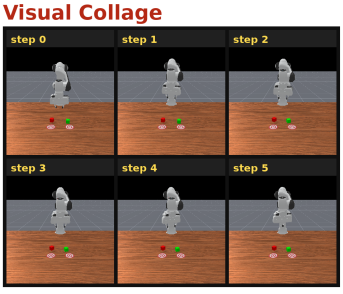

**Compact action log**

```
step 0:
action=push(109,105,20,100,34,20
)
legal=true remaining_stages=2
remaining_turns=5
step 1:
action=push(154,33,20,159,33,20)
legal=true remaining_stages=2
remaining_turns=4
step 2:
action=place(146,108,20)|place(1
46,39,50)
legal=true remaining_stages=2
remaining_turns=3
step 3:
action=push(146,33,20,100,33,160
)|place(146,33,20)
legal=true remaining_stages=2
remaining_turns=2
step 4:
action=push(154,108,20,164,102,2
0)|place(146,33,20)
legal=true remaining_stages=2
remaining_turns=1
```

**SFT JSON DATA**

```
{
 "pivot_step": 4,
 "failure_mode":
 "order_error",
 "failure_reason":
 "Avoid pushing the red cube right
 while placing the green cube at
 (146,33,20) — that action misaligns
 the cubes. At step 4, do not repeat
 push(154,108,20,164,102,20)|place(146
 ,33,20);
 instead, push the red cube left to
 align both cubes at x=0."
}
```

**Visual Collage**




**Compact action log**

```
step 0:
action=pick(80,-100,20)|place(80
,-43,50)
legal=true remaining_stages=3
remaining_turns=5
step 1:
action=place(0,115,30)|place(0,-
115,30)
legal=true remaining_stages=3
remaining_turns=4
step 2:
action=pick(80,-100,0)|place(60,
-95,50)
legal=true remaining_stages=3
remaining_turns=3
step 3:
action=place(80,-100,20)|place(8
0,100,20)
legal=true remaining_stages=3
remaining_turns=2
step 4: action=pick(75,-43,50)
legal=true remaining_stages=3
remaining_turns=1
step 5:
action=pick(80,-100,0)|place(80,
0,0)
legal=true remaining_stages=3
remaining_turns=0
```

**SFT JSON DATA**

```
{
 "pivot_step": 3,
 "failure_mode":
 "timeout",
 "failure_reason":
 "Avoid placing both cubes
 simultaneously at step 3. At step 3,
 do not repeat
 place(80,-100,20)|place(80,100,20);
 instead, place only the red cube at
 its target."
}
```

Figure 7: PrimitiveSkill. SFT-stage input–output examples for this environment. Columns 1–2 are the visual trajectory collage and compact action log; column 3 is the JSON supervision target.

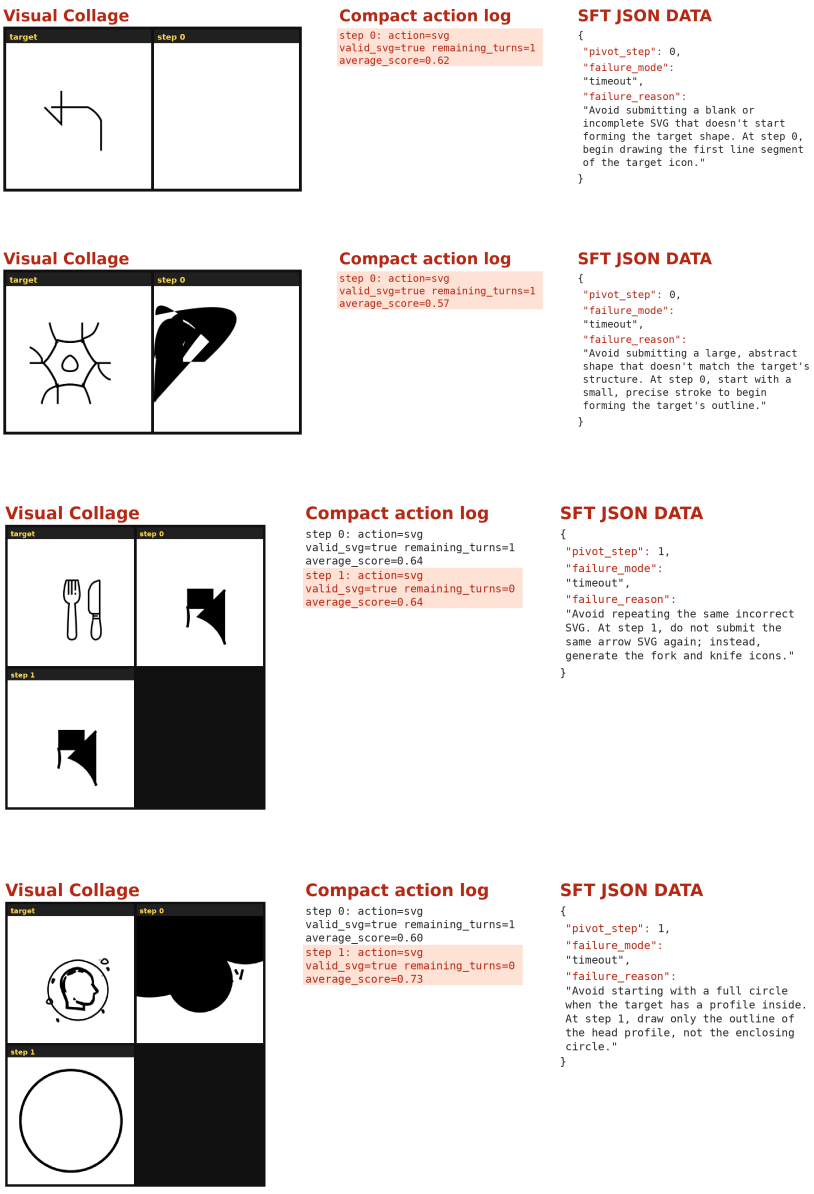



Figure 8: SVG. SFT-stage input–output examples for this environment. Columns 1–2 are the visual trajectory collage and compact action log; column 3 is the JSON supervision target.

Table 13: SDAR settings (3B student, frozen 7B teacher).

| Item | Value |
|---|---|
| Student $\pi_\theta$ / teacher $\pi_\phi$ | `Qwen2.5-VL-3B` / `7B-Instruct` (frozen) |
| Skill | `seed_visual`, `episode_only`; all frames, no collage |
| Distillation | all valid tokens; $\lambda_{\text{sdar}}$=0.01, $\beta_{\text{sdar}}$=5 |
| RL (shared with PIVOT) | $N$=8, LR $1\times10^{-6}$, KL 0.01, invalid pen. 0.1 |
| Training | batch 16/val 128; 250 updates (Sok. 350); history 2 |
| Pivot Analyzer / $P_{\hat{t}}$ | none |

# I SDAR REPRODUCTION

We reproduce SDAR (Lu et al., 2026) on the same five VAGEN tasks, rewards, and validation budget as PIVOT. Unlike PIVOT, SDAR skips Stage I Analyzer SFT and never localizes a pivot step: a frozen larger VLM writes one episode-level visual skill and supplies gated reverse-KL targets on all valid action tokens. The student remains `Qwen2.5-VL-3B-Instruct` initialized from the raw Instruct checkpoint; because the Qwen2.5 family has no 8B variant, the teacher is `Qwen2.5-VL-7B-Instruct`. Tab. 1 reports scores; the following records roles and implementation flags for audit.

**Roles.** Only the 3B student $\pi_\theta$ is trained; a frozen 7B teacher $\pi_\phi$ is queried through an OpenAI-compatible vision API (`analysis_backend=openai`). There is no Analyzer head, failure-mode vocabulary, or pivot panel. On each on-policy trajectory, $\pi_\phi$ sees every RGB frame as its own image (`use_visual_collage=False`) together with the action transcript, generates a single free-form episode skill $r_\phi$ (`skill_mode=episode_only`, prompt `seed_visual`), and rescores the student's sampled tokens under $\tilde{h}_t^\phi = (h_t, r_\phi)$. The 7B model is dropped at inference, leaving the same zero-overhead student rollout as PIVOT.

**Objective.** Let $\ell_t^{\text{stu}} = \log \pi_\theta(a_t \mid h_t)$ and $\ell_t^\phi = \log \pi_\phi(a_t \mid \tilde{h}_t^\phi)$, with gated reverse KL

$$\Delta_t^\phi = \text{sg}\big[\ell_t^\phi - \ell_t^{\text{stu}}\big], \qquad g_t^\phi = \sigma\big(\beta_{\text{sdar}}\Delta_t^\phi\big), \tag{14}$$

$$\mathcal{L}_{\text{SDAR}}(\theta) = \mathbb{E}\Big[m_{\text{act}}\, g_t^\phi \,\big(\text{sg}[\ell_t^\phi] - \ell_t^{\text{stu}}\big)\Big]. \tag{15}$$

The joint loss is the same GRPO-plus-distillation form as Eq. eq. 8,

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{GRPO}}(\theta) + \lambda_{\text{sdar}}\mathcal{L}_{\text{SDAR}}(\theta), \tag{16}$$

with $\phi$ under stop-gradient. Unlike PIVOT, Eq. eq. 15 is applied to all valid action tokens rather than failed trajectories only, and the teacher context is a global episode skill rather than a localized visual panel $P_{\hat{t}}$.

**Protocol.** We match the VAGEN data, horizons, sparse rewards, GRPO grouping, and validation budget of Sec. 4.1 and App. E: Sokoban (room $[6, 6]$, 1 box, solution length $[1, 5]$, $T_{\max}$=9), FrozenLake $4\times4$ with no slip ($T_{\max}$=9), Navigation ($T_{\max}$=25), PrimitiveSkill ($T_{\max}$=6), and SVG ($T_{\max}$=2). SDAR runs only on Qwen2.5-VL-3B (no Qwen3-VL-2B row in Tab. 1).

## J TRAINING CURVES

Figs. 9 and 10 show Stage II validation learning curves for Vanilla-GRPO, M, M+P, and M+P+R on Qwen2.5-VL-3B and Qwen3-VL-2B; dots mark running-best validation points aligned with Tab. 1.

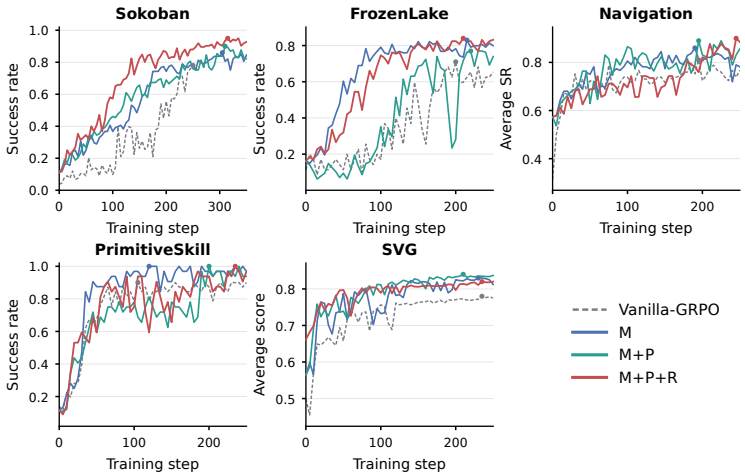



Figure 9: Training curves for Vanilla-GRPO, M, M+P, and M+P+R on Qwen2.5-VL-3B. Dots mark the running-best validation point.

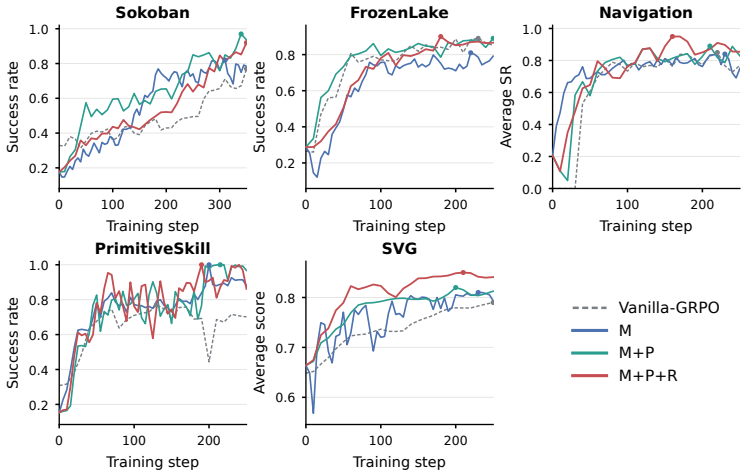



Figure 10: Training curves for Vanilla-GRPO, M, M+P, and M+P+R on Qwen3-VL-2B. Dots mark the running-best point.

## K DISCUSSION AND LIMITATIONS

While PIVOT demonstrates strong performance and efficiency across multi-turn VLM agent tasks, several scope boundaries and promising directions for future work remain:

- **Pivot Supervision Source & Discovery:** Stage I SFT currently relies on offline target labels generated via feasibility heuristics. Extending PIVOT to complex environments lacking explicit solvers presents an exciting opportunity for unsupervised or self-diagnostic pivot discovery. Inspired by recent insights on token-level uncertainty and self-adaptation (Ding & Zhang, 2026), the pivot step $t^*$ could be determined intrinsically using token-level entropy metrics (e.g., locating critical state transitions via minimum-entropy positions), or empirically via rollout sampling by identifying the step where the suffix return lift growth is minimized.

- **Task Domain Applicability:** The current definition of the pivot step $t^*$ relies on remaining-budget goal reachability, which naturally aligns with structured visual environments. Adapting it to open-ended dialogue or continuous control may use dynamic, probabilistic, or entropy-guided reachability.
- **Model Scale and Physical Realization:** Our evaluation focuses on compact VLM backbones (Qwen2.5-VL-3B and Qwen3-VL-2B) within simulated benchmarks. Scaling this internalized distillation recipe to larger parameters and non-rewindable physical robotic systems represents an important direction for future exploration.