SOLE-R1: Video-Language Reasoning as the Sole Reward for On-Robot Reinforcement Learning

Philip Schroeder[1, 2] **Thomas Weng**[2] **Karl Schmeckpeper**[2]

Eric Rosen[2] **Stephen Hart**[2] **Ondrej Biza**[2]

[1]MIT [2]RAI Institute

Abstract

Vision-language models (VLMs) have shown impressive capabilities across diverse tasks, motivating efforts to leverage these models to supervise robot learning. However, when used as evaluators in reinforcement learning (RL), today's strongest models often fail under partial observability and distribution shift, enabling policies to exploit perceptual errors rather than solve the task. To address this limitation, we introduce **SOLE-R1** (Self-Observing LEarner), a video-language reasoning model explicitly designed to serve as the *sole* reward signal for online RL. Given only raw video observations and a natural-language goal, SOLE-R1 performs per-timestep spatiotemporal chain-of-thought (CoT) reasoning and produces dense estimates of task progress that can be used directly as rewards. To train SOLE-R1, we develop a large-scale video trajectory and reasoning synthesis pipeline that generates temporally grounded CoT traces aligned with continuous progress supervision. This data is combined with foundational spatial and multi-frame temporal reasoning, and used to train the model with a hybrid framework that couples supervised fine-tuning with RL from verifiable rewards. Across four different simulation environments and a real-robot setting, SOLE-R1 enables zero-shot online RL from random initialization: robots learn previously unseen manipulation tasks without ground-truth rewards, success indicators, demonstrations, or task-specific tuning. SOLE-R1 succeeds on 24 unseen tasks and substantially outperforms strong vision-language rewarders, including Robometer, RoboReward, ReWiND, GPT-5, and Gemini-3-Pro, while exhibiting markedly greater robustness to reward hacking. We release all models, data, code, and demos at the anonymous page: `https://philip-mit.github.io/sole-r1/`

1 Introduction

Large language models (LLMs) and vision-language models (VLMs) have demonstrated impressive capabilities across a wide range of reasoning and perception tasks [18, 57]. These capabilities have motivated growing efforts to leverage foundation models as sources of supervision for robot learning, replacing or reducing the need for task-specific reward engineering and human annotation. Ideally, a robot could acquire new skills entirely from scratch by interacting with the world, while receiving guidance derived solely from pretrained foundation models.

In practice, however, this idealized paradigm remains out of reach. When used as reward functions or evaluators for reinforcement learning (RL), current state-of-the-art models, such as GPT-5 and Gemini-3-Pro, exhibit systematic failures in grounded visual reasoning. Despite exhibiting impressive visual captioning and question-answering abilities, they lack robustness to partial observability and distribution shift [23, 54]. As a result, when robot policies are trained using rewards derived from these models, the policies quickly discover behaviors that exploit failures in perception or reasoning, eliciting high predicted reward, or perceived success, without achieving true success.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

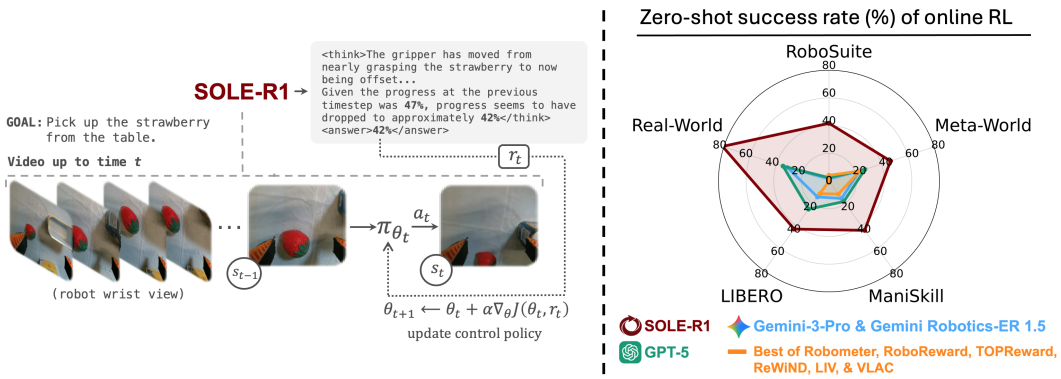


Figure 1: SOLE-R1 is a video-language reasoning model designed to guide online RL with per-timestep chain-of-thought reasoning and progress prediction. In large-scale experiments across 40 tasks, SOLE-R1 outperforms strong baseline models with zero-shot online RL.

To address this limitation, we introduce SOLE-R1, a foundation model with video-language reasoning explicitly designed to guide online RL. SOLE-R1 (**S**elf-**O**bserving **LE**arner) generates per-timestep chain-of-thought (CoT) reasoning directly from raw video observations, yielding a dense estimate of task progress relative to goals specified in natural language. To generate the training data for SOLE-R1, we develop a video trajectory and reasoning synthesis pipeline which produces CoT traces that reason about changes over time, aligned with continuous progress signals. We generate over one million CoT reasoning examples from more than 40,000 real-world and simulated videos. We also carefully curate a diverse collection of general spatial and multi-frame temporal reasoning data to serve as a foundational layer of our training mixture. Together, this training induces video-native reasoning that explicitly integrates both spatial and temporal structure (Figure 2). We propose a two-stage hybrid recipe for training SOLE-R1: (1) supervised fine-tuning (SFT) to develop high-quality spatiotemporal CoT reasoning, (2) RL with verifiable rewards (RLVR) to further develop accurate progress prediction, boosted by the strong CoT reasoning developed during SFT.

We demonstrate that SOLE-R1 reasoning can successfully serve as the *SOLE* signal for learning unseen tasks through online RL. In this setting, the robot begins with a random policy and learns entirely through interaction, guided only by SOLE-R1's predicted rewards - without access to any ground-truth rewards, task-specific tuning, demonstrations, or other off-policy trajectories. In total, SOLE-R1 achieves zero-shot success on 24 unseen manipulation tasks across 4 simulator environments and a real-robot setting, while strong baseline reasoning models (e.g., GPT-5, Gemini-3, and other VLM rewarders) succeed on fewer than 10 tasks. SOLE-R1 generalizes to unseen task families (pick-and-place, articulated object manipulation, button/lever/knob interactions), scene layouts, camera viewpoints, and robot embodiments. We further improve a strong, task-finetuned VLA using only SOLE-R1 rewards for RL post-training, and validate SOLE-R1 through large-scale offline evaluations.

Our contributions include the following:

- We introduce **SOLE-R1**, a video-language reasoning model designed for guiding online RL with per-timestep CoT reasoning and progress prediction.
- We introduce a video trajectory and reasoning synthesis pipeline to generate training data for video-based spatiotemporal CoT reasoning and progress estimation using videos from real-world and simulation.
- We propose a **hybrid training framework** combining SFT for strong multi-frame spatial and temporal CoT reasoning with RLVR to emphasize progress prediction quality and calibration.
- We perform extensive experiments showing SOLE-R1 reasoning can serve as the **sole signal for learning unseen tasks through online RL**, generalizing to new tasks and environments better than strong baseline vision-language reasoning models.
- We release the SOLE-R1 model checkpoints and full training dataset, along with the online RL code and algorithms for reproducing all experiments.

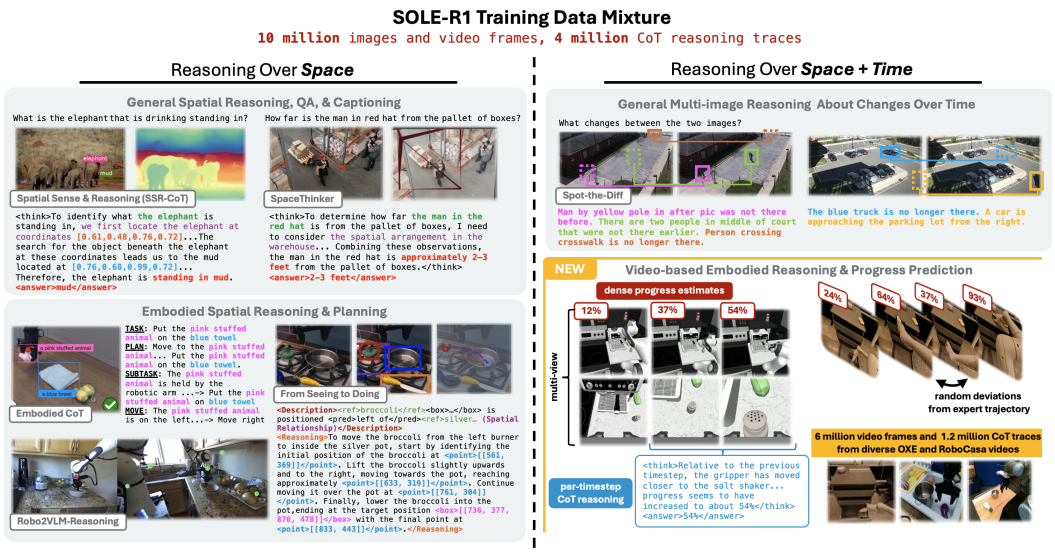


Figure 2: SOLE-R1 training data mixture. The dataset combines foundational spatial reasoning, multi-frame temporal reasoning, and our synthesized video trajectories with chain-of-thought explanations and dense progress supervision, jointly enabling reasoning over space and time for progress prediction.

2 SOLE-R1

SOLE-R1 is a vision-language model designed to perform grounded natural language reasoning over videos of robot trajectories and predict task progress estimates, which can directly serve as dense rewards for online RL. To develop this form of reasoning, we construct a video trajectory and reasoning synthesis pipeline to generate video-based spatiotemporal CoT reasoning and progress estimation training data, layered on top of a foundational mixture of general spatial and multi-frame temporal reasoning, and integrated with a hybrid training framework.

Section 2.1 defines progress prediction and its use as an RL reward. The training pipeline has four components: (1) generating expert, failure, and recovery trajectories; (2) constructing dense progress targets and temporally grounded explanations; (3) combining these examples with foundational spatial and multi-frame reasoning data (Section 3); and (4) SFT followed by RLVR for progress calibration (Section 4).

2.1 Video-native temporal progress reasoning

We design SOLE-R1 to perform *video-native* temporal reasoning for goal-conditioned tasks. Given a natural-language goal $g \in G$ and a video stream of observations $\{o_t\}_{t=1}^{T}$, the model produces (i) a per-timestep CoT explanation $\{m_t\}_{t=1}^{T}$ describing what has changed since the last timestep and what remains to be done, and (ii) a dense scalar progress estimate $\{p_t\}_{t=1}^{T}$ used as a reward signal for online RL.

Temporal conditioning. At timestep t, the input x_t comprises the goal g, a sliding temporal context window of frames up to the present and the previous prediction, $x_t = [g,\ o_0; o_{t-K+1:t},\ p_{t-1}]$, where o_0 is the first video frame, $o_{t-K+1:t}$ denotes the most recent K frames (or all available frames if $t < K$), and p_{t-1} is the model's previous progress prediction. Conditioning on a *multi-frame* window encourages explicit reasoning about motion, contact events, and state transitions (e.g., grasping, opening, insertion) rather than single-frame captioning. During training, the choice of K can vary to elicit greater flexibility in the granularity of task progress increments. Further, we sparsely use $K = 0$ and dropout p_{t-1} during training to force the model to reason about the video globally, avoiding the bias from its previous prediction.

Outputs and format. SOLE-R1 autoregressively generates language tokens that form a structured response, $y_t = [\texttt{<think>}\ m_t\ \texttt{</think>},\ \texttt{<answer>}\ p_t\ \texttt{</answer>}]$, where m_t is free-form natural-language reasoning and $p_t \in [-100, 100]$ denotes task progress at time t. The reasoning m_t is trained to focus on (1) salient visual changes since $t-1$, (2) whether those changes advance or regress the goal, and (3) the next to-be-completed subgoal implied by g.

Dense reward for online RL. To use SOLE-R1 reasoning as a dense reward function, we convert, at each timestep t, the predicted task progress p_t into a reward, $r_t = \psi \,\mathrm{clip}(p_t,\, -c,\, c)$, where ψ is a scaling parameter and c is used to clip the value range. In practice, inference can occur at a lower frequency than the control frequency, in which case we derive a dense reward by linearly interpolating the predicted rewards for all action timesteps. Finally, while we test potential-based reward function shaping [42], we find that simply using the absolute reward is sufficient.

3 SOLE-R1 training data synthesis

SOLE-R1 produces (i) multi-frame CoT explanations grounded in visual evidence and (ii) a dense progress signal suitable for online RL. To elicit robust reasoning, we build the training data in two stages: (1) *foundational* reasoning over **space** (single-image + depth) and **time** (multi-image/video), and (2) *robot-video* spatiotemporal reasoning specialized for **dense progress estimation**.

3.1 Video-based spatiotemporal progress reasoning

3.1.1 Non-expert trajectory generation

We first outline our approach which, given a set of expert demonstrations $\mathcal{D}_{src}$ in **simulation** or **real-world**, generates a dataset of video trajectories, $\mathcal{D}$, exhibiting varying levels of expertise, ranging from nearly expert to fully random action sequences.

For **videos in simulation**, we generate non-expert trajectories by injecting random deviations into expert demonstrations. Given an expert trajectory $\tau^E = \{(s_t^E, a_t^E)\}_{t=1}^{T^E}$, we form τ^N by selecting deviation start times $q \in \mathcal{Q} \subseteq \{1, \ldots, T^E\}$ and, for each q, executing w uniformly sampled actions $a_{q,j}^N \sim \mathcal{U}(A)$ for $j = 0, \ldots, w-1$, producing diverging states $\{s_{q,j}\}_{j=0}^{w}$. After the deviation, the trajectory either terminates or recovers by interpolating from the final deviated state $s_{q,w}^N$ to a downstream expert state: $s_{q,w,z}^N = (1-\alpha_z)\cdot s_{q,w}^N + \alpha_z \cdot s_{q+h}^E$, where $\alpha_z = \frac{z}{n_{\text{interp}}}$ and $z = 1, \ldots, n_{\text{interp}}$.

For **real-world videos**, we do not have access to ground-truth simulator states or the ability to re-simulate injected actions. Thus, we construct non-expert trajectories directly in the *observation space* by temporally perturbing the expert videos. Concretely, given an expert video of frames $\{o_t\}_{t=1}^{T}$, we sample a set of reversal points $\mathcal{Q} \subseteq \{1, \ldots, T\}$ and, for each $q \in \mathcal{Q}$, replace the forward segment with a reversed segment of length w: $o_{q:q+w-1}^N = \mathrm{reverse}\big(o_{q-w+1:q}^E\big)$, where $\mathrm{reverse}(\cdot)$ denotes temporal reversal of the selected window (with boundary handling when $q - w + 1 < 1$). This produces trajectories that visually "undo" recent progress (e.g., objects moving away from a goal configuration) and induces clear regression events using only frames, without requiring robot actions or state access. We further increase diversity by varying the number of reversal points (including nested reversals) and the reversal window length w.

3.1.2 Ground-truth CoT reasoning and progress estimates

For each timestep of the trajectories, we generate CoT reasoning about what has changed relative to the previous timestep, along with an estimate of current progress.

For **videos in simulation**, we leverage the ground-truth simulator state to produce CoT reasoning and progress estimates. Progress is computed from continuous geometric distances: (i) the summed end-effector–object contact-point distance $y_t^{r,e} = \sum_{j=1}^{C} \| p_t^{r_j} - p_t^{l_j} \|_2$ and (ii) the target object's distance to its goal $y_t^{e,f} = \| p_t^{e_\tau} - p_f^{e_\tau} \|_2$. We combine them as $y_t = (1-\beta)\, y_t^{r,e} + \beta\, y_t^{e,f}$ with task-tuned $\beta \in [0, 1]$, then invert and rescale over the trajectory to obtain normalized progress $v_t = \frac{-y_t + \max(y)}{-\min(y) + \max(y)}, \quad y = \{y_t\}_{t=1}^{T}$. Using the distance measures involved in computing $y^{r,e}$ and $y^{e,f}$, we generate dense CoT traces $\{m_t\}_{t=1}^{T}$ that reason about the changes occurring between timestep t and $t-1$ based on templated language. For example, if $y_t^{r,e} < y_{t-1}^{r,e}$ and $y_t^{e,f} \approx y_{t-1}^{e,f}$, the templated language for timestep t would include statements about how the gripper "has moved closer" to the object, but the object "remains in the same position as the previous timestep". We then further enrich the diversity of the template language by refining it with foundation VLMs [52, 14] grounded on the generated language and frames from the current and previous timestep.

For **real-world videos**, we lack privileged robot/environment state information, so we cannot compute geometric distance-based progress. Instead, treating unperturbed videos as expert trajectories, we use *temporal order* as a progress proxy [72, 35, 74, 23]: for a video of length T, we set $v_t = \frac{t-1}{T-1} \in [0, 1]$. For temporally perturbed non-expert trajectories (e.g., reversal windows), we inherit supervision from the originating expert timestep, so reversed segments map to lower v and yield explicit negative-progress events. We then map v_t to the SOLE-R1 progress scale (e.g., $p_t \in [-100, 100]$) and use a foundation VLM conditioned on $(g, o_{t-K+1:t})$ and the progress target to generate dense CoT traces $\{m_t\}_{t=1}^{T}$ that describe visual changes from $t-1$ to t, label them as advancing vs. regressing the goal, and propose the next subgoal. Importantly, the progress supervision anchors the reasoning to the intended temporal direction (forward progress for expert segments, regression for reversed segments), enabling us to produce reliable "advance vs. regress" explanations.

Further details on our data synthesis approach are provided in Appendix J. We also provide extensive video demonstrations of synthesis outputs at: `https://sole-r1.github.io/`

3.1.3 Robot video data sources

We curate real-robot videos from the Open X-Embodiment (OXE) dataset [44], an aggregation of trajectory data from 50 robot datasets spanning diverse tasks, robots, and camera viewpoints. OXE includes 22 embodiments and over 300 tasks. We use RoboCasa [41] as our simulation environment and leverage its provided demonstrations as expert trajectories.

3.2 Foundational spatiotemporal reasoning

Before specializing on progress prediction (Section 3.1), we first train SOLE-R1 for general spatiotemporal grounding.

Reasoning over space. We supervise spatial grounding with explicit 3D relations (relative position, distance, containment, occlusion). We leverage SSR-CoT [29], curating 1.2M image–depth–question–rationale–answer tuples covering relations such as *left/right*, *in front/behind*, *closest/farthest*, and *inside/on top of*. To diversify beyond depth cues, we add synthesized spatial CoT from SpaceOm and SpaceThinker (via VQASynth [45]), reproducing the SpatialVLM synthesis pipeline [8] and emphasizing explanations grounded in visual evidence. We also include robotics-relevant spatial rationales from embodied reasoning datasets (Embodied CoT [69], Robo2VLM-Reasoning [9]).

Reasoning over time (video/multi-image). To make reasoning temporal, we train on data that requires explicit comparison across moments. We include multi-image "spot-the-difference" data [20] to identify what *has and has not changed* (e.g., an object moved), and add embodied video QA/reasoning datasets with action-relevant temporal evidence (RoboVQA [49], From Seeing to Doing [68]). During preprocessing, we normalize all sources into the same multi-frame prompting and structured output format used by SOLE-R1 (Section 2.1), so the model learns to ground per-step rationales in frame-to-frame changes and maintain consistency across varying temporal contexts.

4 SOLE-R1 hybrid training framework

SOLE-R1 produces (i) temporally grounded CoT reasoning over video and (ii) a scalar progress value in `<answer>` to serve as a dense reward for online RL. To train SOLE-R1, we propose a two-stage hybrid recipe: **SFT** teaches high-quality CoT reasoning, while **RLVR** directly emphasizes accurate progress prediction, which is under-emphasized during SFT, as the final answer occupies only a small fraction of response tokens. To explore the impact of each stage of training, in Appendix E, we compare the SFT-only model performance with that of the full SFT+RLVR model.

4.1 Stage 1: Supervised fine-tuning (SFT)

We fine-tune on spatiotemporal reasoning data (Section 3). Each example is (i, q, r, a) with: image/video i, query q, rationale r, and answer a. SFT maximizes likelihood of the output $y = [r; a]$:

$$\mathcal{L}_{\text{SFT}}(\phi) = -\mathbb{E}_{(i,q,r,a)\sim\mathcal{D}}\left[\sum_{t=1}^{|y|} \log p_\phi(y_t \mid i, q, y_{<t})\right].$$

This supervision encourages attention to visually grounded state and changes across frames, yielding transferable video-native reasoning. However, it provides a relatively weak learning signal for *reward/progress prediction*, since the scalar in `<answer>` is a small part of the response.

4.2 Stage 2: RLVR for progress prediction

After strong CoT reasoning is learned through SFT, we refine the model with RL from verifiable rewards while preserving the `<think>`/`<answer>` interface. Using GRPO on the progress dataset (Section 3.1), for each query q we sample G candidates $\{o_i\}_{i=1}^{G} \sim p_{\phi_{\text{old}}}(\cdot \mid q)$, score them with rule-based rewards r_i, and compute within-group standardized advantages $A_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})}$.

GRPO optimizes

$$\mathcal{J}_{\text{GRPO}}(\phi) = \mathbb{E}_{q,\{o_i\}}[\frac{1}{G}\sum_{i=1}^{G} \min(\rho_i(\phi)A_i,\ \text{clip}(\rho_i(\phi), 1-\epsilon, 1+\epsilon)A_i)] - \beta\, D_{\text{KL}}(p_\phi \,\|\, p_{\text{ref}})\,,$$

where $\rho_i(\phi) = \exp(\log p_\phi(o_i \mid q) - \log p_{\phi_{\text{old}}}(o_i \mid q))$ and p_{ref} is the SFT reference p_{SFT}.

Verifiable reward. We define $r(o) = r_{\text{format}}(o) + r_{\text{acc}}(o)$, where r_{format} enforces a parseable output structure and r_{acc} measures progress accuracy by comparing the predicted value $\hat{p}_t$ in `<answer>` to the ground-truth progress p_t. The accuracy reward is defined from the absolute prediction error: $r_{\text{acc}}(o) = \alpha\, \exp\left(-\frac{|\hat{p}_t - p_t|}{\tau}\right)$. We scale $r_{\text{acc}}(o) \in [0, 1.5]$ and scale $r_{\text{format}}(o) \in [0, 0.5]$.

5 Experiments

We test the following hypotheses:

1. SOLE-R1 outperforms much larger general-purpose reasoning models and special-purpose reward models in zero-shot reward prediction for online RL (Section 5.3).
2. SOLE-R1 is less susceptible to reward hacking than current state-of-the-art reasoning models, including GPT-5 and Gemini-3-Pro (Section 5.4).
3. SOLE-R1's performance is specifically facilitated by CoT reasoning and by including authentic non-expert trajectories during training (Section 5.5).
4. Our data synthesis and training recipe follows a scaling law driven by task diversity (Section 5.6).
5. A strong VLA can improve through RL post-training with only SOLE-R1 rewards (Section 5.9).
6. SOLE-R1 outperforms strong baseline reasoning models in offline evaluations, including large-scale correlation-based analysis (Section 5.7) and general spatial reasoning (Section 5.8).

We provide camera images of all tasks in Appendix A, along with video demonstrations of our training data synthesis and all experimental results at: `https://sole-r1.github.io/`.

5.1 Data synthesis and model training

Our generated training data comprises 1.2M spatiotemporal CoT traces extracted from 41K videos. We combine this with a curated collection of foundational spatial and multi-frame temporal reasoning data, yielding a total of 10M images and video frames and 4M CoT reasoning traces. We fine-tune Qwen3-VL-8B-Instruct using the hybrid recipe described above. During SFT, each batch is balanced across (i) general reasoning, (ii) video embodied reasoning and planning, and (iii) progress prediction. We run SFT for one epoch over all data. We then run RLVR with the video-based progress prediction dataset. Full training hyperparameters and data synthesis details are in Appendix K and J, respectively.

5.2 Baselines

We test the current best **general-purpose reasoning models**, including GPT-5, Gemini-3-Pro, and Gemini Robotics-ER 1.5. We exclude open-source VLMs, such as Qwen3-VL-72B-Instruct, because they do not achieve meaningful success. We also include the strongest existing **special-purpose reward models**: Robometer [27], RoboReward [23], TOPReward [10], LRM [63], ReWiND [73], VLAC [74], and LIV [37]. Finally, Robo-Dopamine's standard adaptation uses one target-task demonstration [55], which falls outside our zero-shot protocol. We report both unadapted and one-demonstration results separately in Appendix O.

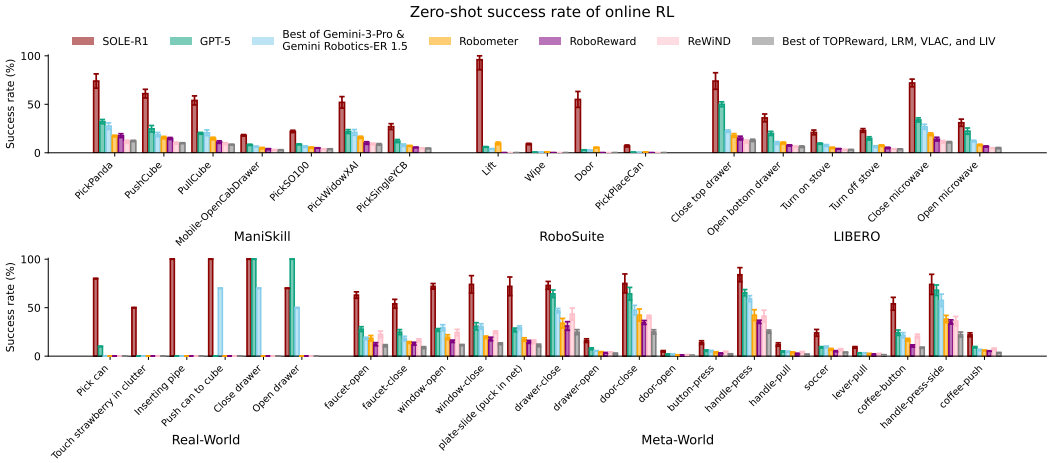


Figure 3: Zero-shot success rate of online RL across 40 tasks. We plot the mean and standard error across three random seeds (real-world experiments use a single seed, shown as a single value). In all experiments, the robot begins with a random policy and learns entirely through interaction with the task, guided only by the predicted rewards.

5.3 Zero-shot online RL

We evaluate whether SOLE-R1 can serve as the *sole* signal for learning manipulation skills from scratch via online RL. We run experiments across four simulation benchmark suites—RoboSuite, ManiSkill, Meta-World, and LIBERO—and in a real-world tabletop manipulation setting. Across all settings, we evaluate a total of 40 tasks, spanning pick-and-place, articulation, button/lever/knob interactions, and mobile manipulation. All hyperparameters and details are provided in Appendix G.

We use a SERL implementation of DrQv2. The policy observes two RGB streams (a wrist view and an external view) and proprioception. Actions are end-effector delta motions and a gripper open/close command. We do not use any additional privileged state, depth, object poses, or task-specific sensors.

Unlike prior work that (i) learns from ground-truth rewards and/or (ii) tunes reward models or policies on task demonstrations [37, 73, 74, 23, 54, 5], we evaluate in a fully **zero-shot** online RL setting:

- **No ground-truth rewards.** The policy never observes ground-truth/external rewards (dense or sparse) and receives no success labels during training.
- **No demonstrations or offline trajectories.** The policy starts with random actions and learns only from on-policy interaction.
- **No task-specific tuning.** Reward models are used as-is, with fixed prompting across tasks.

SOLE-R1 enables zero-shot online RL from scratch. SOLE-R1 achieves at least 50% success on 24 tasks, substantially outperforming all baselines (Figure 3). The strongest baselines include GPT-5 and Gemini, but they reach 50% success on only 7 and 5 tasks, respectively. The non-reasoning models achieve near-zero success on all tasks, with the exception of Meta-World tasks, where Robometer, RoboReward, and ReWiND achieve above 40% success rate on 4 tasks.

SOLE-R1 generalizes to unseen tasks and environments. SOLE-R1 succeeds with tasks that significantly differ from the task types seen during training, such as sliding a puck into a net, opening and closing windows, and manipulating unseen levers and handles in novel ways based on the natural language task specification. This suggests that SOLE-R1 does not merely memorize task templates, but instead learns reusable spatiotemporal progress primitives (e.g., establishing contact, aligning a grasp, changing articulation state, placing/settling objects) that transfer to unseen tasks.

SOLE-R1 generalizes to unseen embodiments and camera viewpoints. SOLE-R1 solves tasks with the Franka, along with embodiments not seen during training, including the Sawyer robot in Meta-World, the WidowX AI and Fetch Mobile Manipulator in ManiSkill, and the modified Franka with different gripper fingers and wrist camera angle in real-world. We also see SOLE-R1 solve tasks with camera views that were not used during training. This indicates that SOLE-R1 reward predictions are not narrowly tied to a particular kinematic chain or gripper appearance, but instead track goal-relevant object state changes across morphology and camera placement.

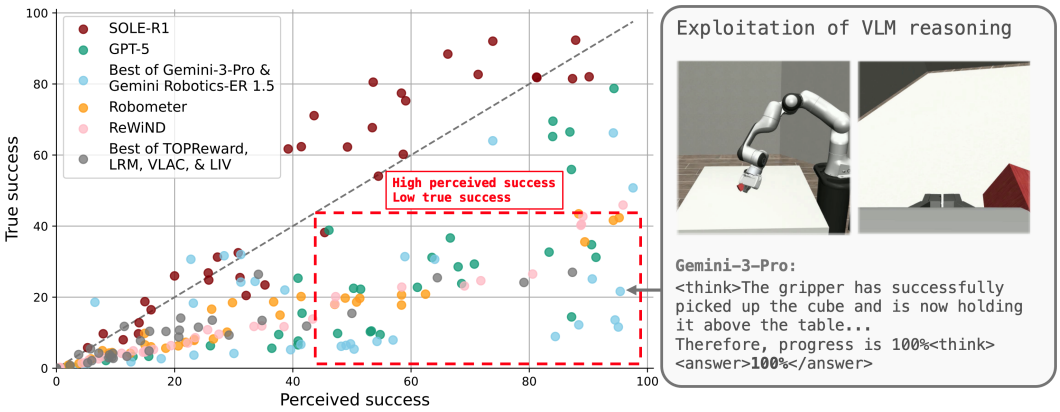


Figure 4: Perceived vs true success in zero-shot RL. Perceived success is the average max progress predicted (RoboReward is excluded as it does not provide dense rewards). True success is the average max ground-truth reward achieved.

5.4 Failure analysis

We analyze failures at the task level, where rewards may be exploitable, miscalibrated, or too weak for learning, and at the frame level, where incorrect progress estimates reflect missed visual evidence. Full details are in Appendix C.

Using perceived-vs.-true success (Figure 4), we distinguish *reward hacking* (high perceived, low true) from *signal-limited* failures (low perceived, low true). General-purpose VLM rewarders (e.g., GPT-5, Gemini) mainly fail through reward hacking: RL discovers behaviors that inflate predicted progress without solving the task. In contrast, SOLE-R1 failures are more often signal-limited, indicating that it usually recognizes non-success but sometimes provides rewards too flat or noisy to drive learning.

In our hand review, temporal under-detection and partial observability account for 34% and 29% of SOLE-R1's primary errors, respectively. Perceptual hallucination of success accounts for 9%, versus 42% for GPT-5, 38% for Gemini-3-Pro, and 35% for Gemini Robotics-ER (Table 2). These results indicate reduced exploitation of perceptual errors, while brief or occluded evidence remains a limitation.

5.5 Ablations

We test the following ablations of SOLE-R1. Full details in Appendix D.

No-CoT progress training: We remove the `<think>` channel and train to emit only `<answer>` progress. This degrades learning and shifts failures toward *signal-limited* (Figure 5): progress becomes flatter/noisier, with long plateaus that under-react to small but task-critical transitions. This supports the utility of reasoning supervision and generation, without establishing that each rationale faithfully explains the prediction.

Expert-only video progress supervision: We train only on expert trajectories (keeping simple reversal augmentation). This substantially increases *reward-hacking* failures: states that merely look goal-adjacent receive inflated progress, enabling exploitation (Figure 5).

No foundational spatial & multi-frame reasoning data: We remove the general spatial and temporal grounding mixture and train only on embodied/planning and robot progress data. This reduces overall success and again increases reward hacking (Figure 5).

5.6 Zero-shot scaling

We find that our data synthesis and training recipe follows a scaling law driven by the diversity of training tasks (Figure 6). We train variants of SOLE-R1 with an increasing number of task types included in our training data synthesis (details in Appendix M). Figure 6 plots the number of downstream tasks that achieve different success thresholds as a function of training task diversity.

5.7 OpenX Embodiment Value-Order-Correlation

Using 1,000 videos from 50 OXE datasets, we perform the same large-scale Value-Order-Correlation (VOC) analysis conducted in [35] for Generative Value Learning (GVL). SOLE-R1 achieves higher

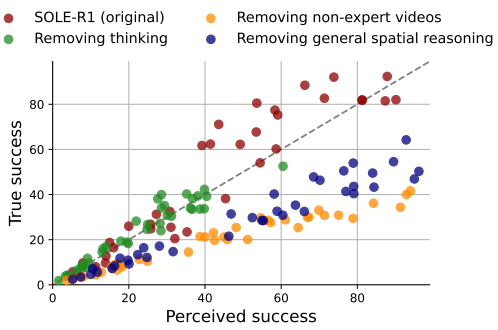


Figure 5: Ablated-models zero-shot success.

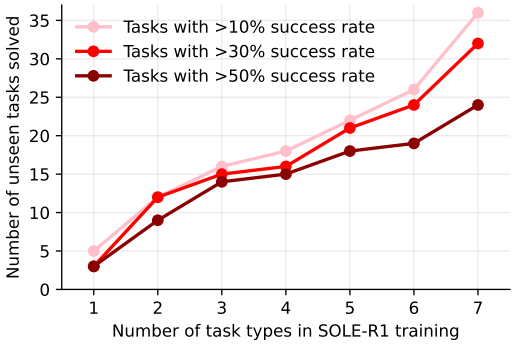


Figure 6: Tasks solved vs training task diversity.

VOC than GVL on real-world expert demonstrations (Table 8), particularly for known high-quality datasets collected from human teleoperators with fixed cameras, such as Bridge, RT-1, and Dobb-E.

5.8 General spatial and vision-language reasoning

SOLE-R1 achieves consistent gains over the strong SSR spatial reasoning model across SpatialBench, SSRBench, and CV-Bench (Table 7). This suggests our data synthesis and training recipe not only yields strong progress prediction, but also improves general-purpose reasoning in these settings.

5.9 RL post-training of a VLA

We initialize with 7B OpenVLA-OFT [21], pretrained on diverse robot data and fine-tuned on 50 demonstrations per target LIBERO task. We retain RIPT-VLA's policy-optimization framework [56] and replace its ground-truth reward with SOLE-R1 predictions from the task instruction and rollout video. No ground-truth reward or success signal is available during post-training. Each final policy is evaluated on 100 independent rollouts per task. Success increases on all four tasks, from 93.25% to 98.00% on average (Table 1), a 4.75-point gain and 70.4% relative reduction in failure rate. This tests reward-only RL improvement of an already task-finetuned policy. Additional details are in Appendix N.

Table 1: OpenVLA-OFT success before and after RIPT-VLA post-training with SOLE-R1 rewards. Rates are percentages; Δ is in percentage points. Means are unweighted across tasks.

LIBERO task	Before	After	Δ
Kitchen 3: turn on stove; place pot	95	98	+3
Kitchen 4: bowl in bottom drawer; close	94	99	+5
Living room 1: soup and cream cheese in basket	91	97	+6
Study 1: book in caddy compartment	93	98	+5
Mean	93.25	**98.00**	+4.75

6 Related work

On-robot RL. Seminal works that perform RL on physical hardware rely on hand-crafted reward functions, reset policies, and safety constraints [25, 26]. Recent works improve the scalability of RL using pre-trained policies [24, 19], shaped rewards [65, 5] and human-in-the-loop learning [32]. Despite these advances, previous work in on-robot RL requires significant task-specific engineering.

VLM reward learning. VLMs are increasingly used to supervise robot learning [47, 36, 48]. Closest to our work, RoboReward [23], Robometer [27], and LRM [63] predict progress from videos, but rely on expert trajectories, lack intermediate reasoning, and are evaluated mainly on narrow task families rather than zero-shot online RL across diverse environments. Robo-Dopamine [55] uses one-demonstration reward-model adaptation in its standard RL pipeline. We distinguish this from zero-shot reward use in Appendix O. **More discussion in Appendix F.**

7 Limitations

Although SOLE-R1 generalizes across many unseen tasks, embodiments, and viewpoints, it remains constrained by the quality and coverage of its video-based progress supervision. Its rewards can

still be flat, noisy, or miscalibrated when visual evidence is brief, occluded, or ambiguous. Further, SOLE-R1 is not a substitute for robot safety mechanisms: even with reduced reward hacking relative to strong VLM baselines, learned policies can still exploit reward errors under distribution shift, so deployment on physical robots requires external constraints, monitoring, and validation.

Broader impact

SOLE-R1 advances robotics by enabling zero-shot RL and reducing the effort required to train robots for new tasks. However, reward models also pose risks: they may be misused to increase unsafe robot autonomy and create safety-critical failures, especially under distribution shift or reward hacking. Addressing these risks requires continual consideration of safety and thorough robustness checks.

8 Conclusion

We introduced SOLE-R1, a video-language reasoning model for guiding online RL via per-timestep CoT reasoning and dense progress prediction. SOLE-R1 is trained using a video trajectory and reasoning synthesis pipeline that generates spatiotemporal CoT and progress supervision, combined with a foundational mixture of spatial and multi-frame temporal reasoning and a hybrid training framework. We show that SOLE-R1 enables zero-shot learning of unseen tasks from scratch and substantially outperforms both general and specialized reward models. Together, these results indicate that grounded spatiotemporal reasoning is a promising route toward general, reusable reward models.

9 Acknowledgements

This work was conducted at the Robotics & AI Institute (RAI) as part of the first author's summer internship and was supported by RAI. No additional external funding was received and the authors declare no other competing interests.

References

[1] Alakuijala, M., McLean, R., Woungang, I., Farsad, N., Kaski, S., Marttinen, P., and Yuan, K. (2024). Video-language critic: Transferable reward functions for language-conditioned robotics. arXiv preprint arXiv:2405.19988.

[2] Ankile, L., Jiang, Z., Duan, R., Shi, G., Abbeel, P., and Nagabandi, A. (2025a). Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*.

[3] Ankile, L., Simeonov, A., Shenfeld, I., Torne, M., and Agrawal, P. (2025b). From imitation to refinement—residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE.

[4] Baumli, K., Baveja, S., Behbahani, F., Chan, H., Comanici, G., Flennerhag, S., Gazeau, M., Holsheimer, K., Horgan, D., Laskin, M., et al. (2023). Vision-language models as a source of rewards. arXiv preprint arXiv:2312.09187.

[5] Biza, O., Weng, T., Sun, L., Schmeckpeper, K., Kelestemur, T., Ma, Y. J., Platt, R., van de Meent, J.-W., and Wong, L. L. (2025). On-robot reinforcement learning with goal-contrastive rewards. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4797–4805. IEEE.

[6] Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. (2024). pi 0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164.

[7] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. (2022). Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

[8] Chen, B., Xu, Z., Kirmani, S., Ichter, B., Driess, D., Florence, P., Sadigh, D., Guibas, L., and Xia, F. (2024). Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. *arXiv preprint arXiv:2401.12168*.

[9] Chen, K., Xie, S., Ma, Z., Sanketi, P. R., and Goldberg, K. (2025a). Robo2vlm: Visual question answering from large-scale in-the-wild robot manipulation datasets. *arXiv preprint arXiv:2505.15517*.

[10] Chen, S., Harrison, C., Lee, Y.-C., Yang, A. J., Ren, Z., Ratliff, L. J., Duan, J., Fox, D., and Krishna, R. (2026). Topreward: Token probabilities as hidden zero-shot rewards for robotics. *arXiv preprint arXiv:2602.19313*.

[11] Chen, Y., Tian, S., Liu, S., Zhou, Y., Li, H., and Zhao, D. (2025b). Conrft: A reinforced fine-tuning method for vla models via consistency policy. arXiv preprint arXiv:2502.05450.

[12] Dalal, M., Chiruvolu, T., Chaplot, D., and Salakhutdinov, R. (2024). Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. *arXiv preprint arXiv:2405.01534*.

[13] Du, Y., Konyushkova, K., Denil, M., Raju, A., Landon, J., Hill, F., de Freitas, N., and Cabi, S. (2023). Vision-language models as success detectors. arXiv preprint arXiv:2303.07280.

[14] Google DeepMind (2025). Gemini 3 Pro Model Card. Technical report, Google DeepMind. Accessed: 2025-12-23.

[15] Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE.

[16] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

[17] Hu, J., Hendrix, R., Farhadi, A., Kembhavi, A., Martín-Martín, R., Stone, P., Zeng, K.-H., and Ehsani, K. (2025). Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3617–3624. IEEE.

[18] Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. (2024). Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

[19] Intelligence, P., Amin, A., Aniceto, R., Balakrishna, A., Black, K., Conley, K., Connors, G., Darpinian, J., Dhabalia, K., DiCarlo, J., et al. (2025). pi0.6: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*.

[20] Jhamtani, H. and Berg-Kirkpatrick, T. (2018). Learning to describe differences between pairs of similar images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[21] Kim, M. J., Finn, C., and Liang, P. (2025). Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*.

[22] Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. (2024). Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

[23] Lee, T., Wagenmaker, A., Pertsch, K., Liang, P., Levine, S., and Finn, C. (2026). Roboreward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*.

[24] Lei, K., Li, H., Yu, D., Wei, Z., Guo, L., Jiang, Z., Wang, Z., Liang, S., and Xu, H. (2025). Rl-100: Performant robotic manipulation with real-world reinforcement learning. *arXiv preprint arXiv:2510.14830*.

[25] Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373.

[26] Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research*, 37(4-5):421–436.

[27] Liang, A., Korkmaz, Y., Zhang, J., Hwang, M., Anwar, A., Kaushik, S., Shah, A., Huang, A. S., Zettlemoyer, L., Fox, D., et al. (2026). Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*.

[28] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2023). Let's verify step by step. arXiv preprint arXiv:2305.20050.

[29] Liu, Y., Ma, M., Yu, X., Ding, P., Zhao, H., Sun, M., Huang, S., and Wang, D. (2025). Ssr: Enhancing depth perception in vision-language models via rationale-guided spatial reasoning. *arXiv preprint arXiv:2505.12448*.

[30] Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., Tang, Y., and Zhang, D. (2025a). Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. arXiv preprint arXiv:2308.09583.

[31] Luo, J., Hu, Z., Xu, C., Tan, Y. L., Berg, J., Sharma, A., Schaal, S., Finn, C., Gupta, A., and Levine, S. (2024). Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16961–16969. IEEE.

[32] Luo, J., Xu, C., Wu, J., and Levine, S. (2025b). Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033.

[33] Luo, R., Zheng, Z., Wang, L., Wang, Y., Ni, X., Lin, Z., Jiang, S., Yu, Y., Shi, C., Chu, R., et al. (2025c). Unlocking multimodal mathematical reasoning via process reward model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[34] Luu, T. M., Lee, Y., Lee, D., Kim, S., Kim, M. J., and Yoo, C. D. (2025). Erlvlm: Enhancing rating-based reinforcement learning to effectively leverage feedback from large vision–language models. arXiv preprint arXiv:2506.12822.

[35] Ma, Y. J., Hejna, J., Fu, C., Shah, D., Liang, J., Xu, Z., Kirmani, S., Xu, P., Driess, D., Xiao, T., et al. (2024a). Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations*.

[36] Ma, Y. J., Hejna, J., Wahid, A., Fu, C., Shah, D., Liang, J., Xu, Z., Kirmani, S., Xu, P., Driess, D., Xiao, T., Tompson, J., Bastani, O., Jayaraman, D., Yu, W., Zhang, T., Sadigh, D., and Xia, F. (2024b). Vision language models are in-context value learners. arXiv preprint arXiv:2411.04549.

[37] Ma, Y. J., Kumar, V., Zhang, A., Bastani, O., and Jayaraman, D. (2023). Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pages 23301–23320. PMLR.

[38] Mark, M. S., Gao, T., Sampaio, G. G., Srirama, M. K., Sharma, A., Finn, C., and Kumar, A. (2024). Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. arXiv preprint arXiv:2412.06685.

[39] Mendonca, R., Panov, E., Bucher, B., Wang, J., and Pathak, D. (2024). Continuously improving mobile manipulation with autonomous real-world rl. arXiv preprint arXiv:2409.20568.

[40] Nakamoto, M., Mees, O., Kumar, A., and Levine, S. (2024). Steering your generalists: Improving robotic foundation models via value guidance. arXiv preprint arXiv:2410.13816.

[41] Nasiriany, S., Maddukuri, A., Zhang, L., Parikh, A., Lo, A., Joshi, A., Mandlekar, A., and Zhu, Y. (2024). Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*.

[42] Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In Bratko, I. and Dzeroski, S., editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287. Morgan Kaufmann.

[43] Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Xu, C., Luo, J., Kreiman, T., Tan, Y. L., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. (2024). Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, Delft, Netherlands.

[44] O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., et al. (2024). Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE.

[45] remyxai (2024). Spacethinker dataset. `https://huggingface.co/datasets/remyxai/SpaceThinker`. Hugging Face Datasets.

[46] Riedmiller, M., Gabel, T., Hafner, R., and Lange, S. (2009). Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73.

[47] Rocamonde, J., Montesinos, V., Nava, E., Perez, E., and Lindner, D. (2023). Vision-language models are zero-shot reward models for reinforcement learning. arXiv preprint arXiv:2310.12921.

[48] Schroeder, P., Biza, O., Weng, T., Luo, H., and Glass, J. (2026). Rover: Recursive reasoning over videos with vision-language models for embodied tasks. *Advances in Neural Information Processing Systems*, 38:140623–140669.

[49] Sermanet, P., Ding, T., Zhao, J., Xia, F., Dwibedi, D., Gopalakrishnan, K., Chan, C., Dulac-Arnold, G., Maddineni, S., Joshi, N. J., et al. (2024). Robovqa: Multimodal long-horizon reasoning for robotics. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 645–652. IEEE.

[50] Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300.

[51] Singh, A., Bhaskar, A., Yu, P., Chakraborty, S., Dasyam, R., Bedi, A., and Tokekar, P. (2025a). Varp: Reinforcement learning from vision–language model feedback with agent-regularized preferences. arXiv preprint arXiv:2503.13817.

[52] Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al. (2025b). Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

[53] Singh, A., Yang, L., Hartikainen, K., Finn, C., and Levine, S. (2019). End-to-end robotic reinforcement learning without reward engineering. *Robotics: Science and Systems*.

[54] Tan, H., Chen, S., Xu, Y., Wang, Z., Ji, Y., Chi, C., Lyu, Y., Zhao, Z., Chen, X., Co, P., et al. (2025a). Robo-dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*.

[55] Tan, H., Chen, S., Xu, Y., Wang, Z., Ji, Y., Chi, C., Lyu, Y., Zhao, Z., Chen, X., Co, P., Xie, S., Yao, G., Wang, P., Wang, Z., and Zhang, S. (2025b). Robo-dopamine: General process reward modeling for high-precision robotic manipulation. arXiv preprint arXiv:2512.23703.

[56] Tan, S., Dou, K., Zhao, Y., and Krähenbühl, P. (2025c). Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*.

[57] Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

[58] Venkataraman, S., Wang, Y., Wang, Z., Erickson, Z., and Held, D. (2024). Real-world offline reinforcement learning from vision language model feedback. arXiv preprint arXiv:2411.05273.

[59] Venuto, D., Islam, S. N., Klissarov, M., Precup, D., Yang, S., and Anand, A. (2024). Code as reward: Empowering reinforcement learning with vlms. arXiv preprint arXiv:2402.04764.

[60] Wagenmaker, A., Nakamoto, M., Zhang, Y., Park, S., Yagoub, W., Nagabandi, A., Gupta, A., and Levine, S. (2025). Steering your diffusion policy with latent space reinforcement learning. arXiv preprint arXiv:2506.15799.

[61] Wang, D., Jia, M., Zhu, X., Walters, R., and Platt, R. (2022). On-robot learning with equivariant models. In Liu, K., Kulic, D., and Ichnowski, J., editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 1345–1354. PMLR.

[62] Wang, Y., Sun, Z., Zhang, J., Xian, Z., Biyik, E., Held, D., and Erickson, Z. (2024). Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*.

[63] Wu, Y., Yuan, W., Qi, A., Guizilini, V., Mao, J., and Wang, Y. (2026). Large reward models: Generalizable online robot reward generation with vision-language models. *arXiv preprint arXiv:2603.16065*.

[64] Xu, P., Wang, S., Zhu, Y., Li, J., and Zhang, Y. (2025). Spatialbench: Benchmarking multimodal large language models for spatial cognition. *arXiv preprint arXiv:2511.21471*.

[65] Yang, D., Tjia, D., Berg, J., Damen, D., Agrawal, P., and Gupta, A. (2024a). Rank2reward: Learning shaped reward functions from passive video. arXiv preprint arXiv:2404.14735.

[66] Yang, Y., Chen, M., Qiu, Q., Wu, J., Wang, W., Lin, B., Guan, Z., and He, X. (2024b). Adapt2reward: Adapting video–language models to generalizable robotic rewards via failure prompts. arXiv preprint arXiv:2407.14872.

[67] Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. (2022). Mastering visual continuous control: Improved data-augmented reinforcement learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[68] Yuan, Y., Cui, H., Chen, Y., Dong, Z., Ni, F., Kou, L., Liu, J., Li, P., Zheng, Y., and Hao, J. (2025). From seeing to doing: Bridging reasoning and decision for robotic manipulation. *arXiv preprint arXiv:2505.08548*.

[69] Zawalski, M., Chen, W., Pertsch, K., Mees, O., Finn, C., and Levine, S. (2024). Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*.

[70] Zeng, A., Song, S., Lee, J., Rodriguez, A., and Funkhouser, T. (2020). Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319.

[71] Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., and Funkhouser, T. A. (2018). Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pages 4238–4245. IEEE.

[72] Zhang, J., Luo, Y., Anwar, A., Sontakke, S. A., Lim, J. J., Thomason, J., Biyik, E., and Zhang, J. (2025a). RewiND: Language-guided rewards teach robot policies without new demonstrations. In *9th Annual Conference on Robot Learning*.

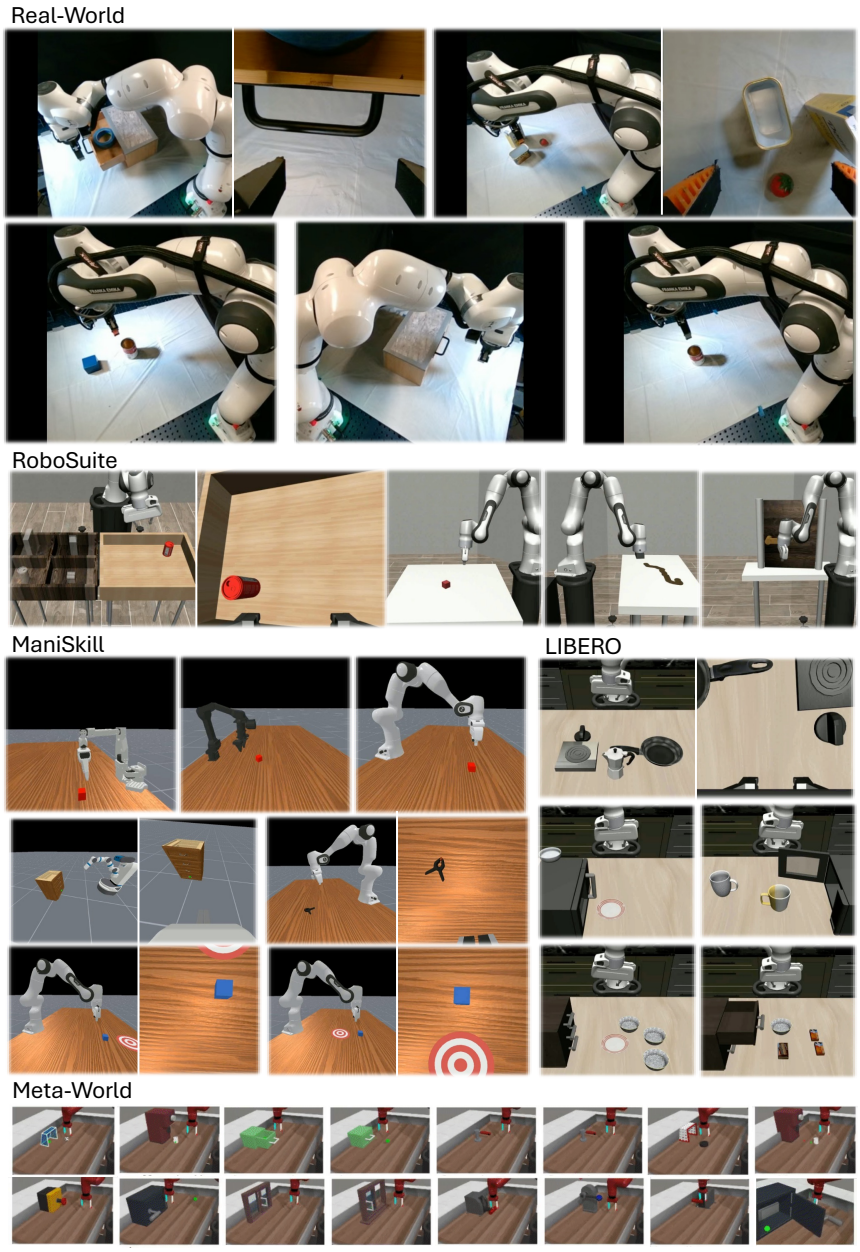
[73] Zhang, J., Luo, Y., Anwar, A., Sontakke, S. A., Lim, J. J., Thomason, J., Bıyık, E., and Zhang, J. (2025b). Rewind: Language-guided rewards teach robot policies without new demonstrations. arXiv preprint arXiv:2505.10911.

[74] Zhang, Q., Zhai, S., Zhang, S., Liu, L., Huang, F., HaoranECNU, Z., Zhou, M., Pang, J., et al. (2025c). Vlac: A generalist action-critic model via pair-wise progress understanding. *arXiv preprint arXiv:2509.15937*.

[75] Zhang, Z., Zheng, K., Chen, Z., Jang, J., Li, Y., Han, S., Wang, C., Ding, M., Fox, D., and Yao, H. (2024). Grape: Generalizing robot policy via preference alignment. arXiv preprint arXiv:2411.19309.

[76] Zhu, N., Dong, Y., Wang, T., Li, X., Deng, S., Wang, Y., Hong, Z., Geng, T., Niu, G., Huang, H., et al. (2025). Cvbench: Evaluating cross-video synergies for complex multimodal understanding and reasoning. *arXiv e-prints*, pages arXiv–2508.

A Visual overview of online RL evaluation suites



B Reasoning examples

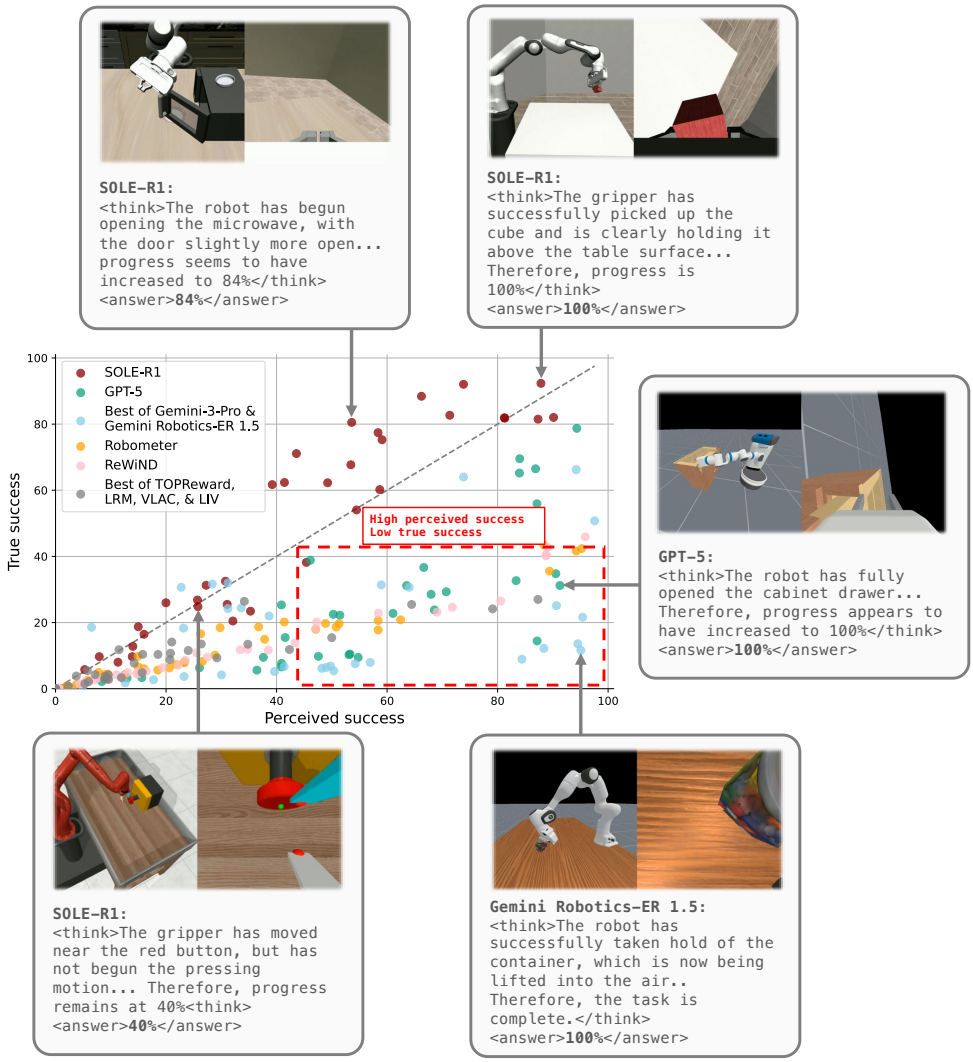


Figure 7: Extended reasoning examples across all models.

C Failure analysis

To understand when and why SOLE-R1 fails to learn a task, we analyze failures at two levels: (i) *task-level reward pathologies* (whether the reward signal is exploitable, miscalibrated, or simply too weak to drive learning), and (ii) *frame-level reasoning errors* (what the model is missing perceptually or temporally when it produces an incorrect progress estimate). Our goal is to distinguish failures due to *reward hacking* from failures due to *insufficient learning signal* under partial observability and distribution shift.

Perceived success vs. true success. Figure 4 plots, the final policy's *perceived success* based on the predicted rewards (x-axis) versus the *true success* it achieves for each task. This visualization separates failures into two qualitatively different types:

- *Reward-hacking failures (high perceived, low true).* Points in the lower-right quadrant indicate that the policy found behaviors that elicit high predicted progress without achieving the task. This is the dominant failure mode for general-purpose VLM rewarders (GPT-5, Gemini-3-Pro, Gemini Robotics-ER), consistent with online RL actively searching for reward loopholes.
- *Signal-limited failures (low perceived, low true).* Points in the lower-left quadrant indicate that the learned reward remains pessimistic and the policy never reaches states that the model considers meaningfully closer to the goal. These failures are more common for SOLE-R1 than for baseline VLMs, reflecting that SOLE-R1 is harder to "trick" into claiming success, but can still provide a reward that is too noisy or too weak to bootstrap exploration within the episode budget.

For SOLE-R1, the majority of failed tasks lie in the *signal-limited* failure type (lower-left quadrant), indicating that the model typically *recognizes* non-success rather than hallucinating completion. This contrasts with GPT-5 and Gemini variants, where many failures appear in the *reward-hacking* failure type (lower-right quadrant), where the learned policy reliably induces high progress predictions while true success remains near zero.

To quantify whether failure is driven by misalignment of the dense reward, we additionally compute the correlation between predicted rewards and ground-truth rewards along trajectories (Figure 8). We find that for SOLE-R1, many signal-limited failures also exhibit *low reward correlation*, suggesting that the reward does not provide a sufficiently shaped gradient toward success even when the model is not being exploited. In practice, this manifests as the agent oscillating among visually similar non-progress states (e.g., hovering near an object, repeatedly tapping a handle, or regrasping without changing articulation state) with consistently low predicted progress.

A possible source of weak learning signals. We have not established that suboptimal-trajectory synthesis causes signal-limited failures. One hypothesis is that smooth geometric progress targets underemphasize discrete task milestones: accurate continuous progress can still provide weak credit assignment near a completion threshold. More milestone-sensitive progress targets or discrete subgoal bonuses may help, but their effect remains to be evaluated.

Common failure modes from qualitative review. We hand-review 100 rollouts from experiments in simulation and real-world across all methods, and assign each failure to a primary error category (Table 2). We summarize the most common SOLE-R1 failure modes below, and contrast them with baseline VLM rewarders.

1. *Temporal under-detection of contact and state transitions.* Some tasks hinge on brief events (first contact, latch release, button actuation) that are hard to infer from RGB alone and may occur between model query steps. When these events are missed, progress remains flat despite real advancement, yielding weak learning signal. This occurs most often when (i) the event is visually subtle, (ii) the wrist camera is occluded by the gripper, or (iii) progress depends on a latent state (e.g., a switch toggled) with minimal pixel change.
2. *Ambiguous object state under partial observability.* In cluttered scenes, the model can be uncertain whether an object is *actually* grasped, fully inserted, or properly seated, especially when only one viewpoint clearly resolves the state. SOLE-R1 tends to respond

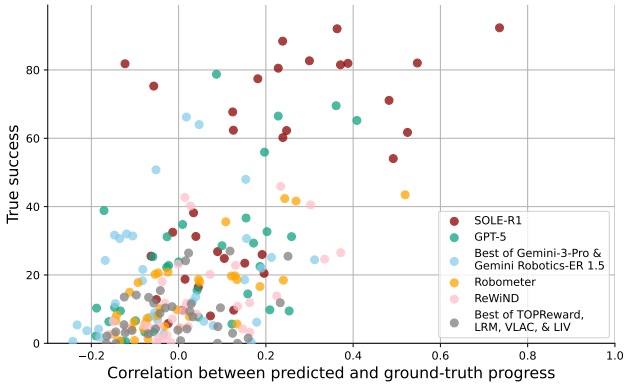


Figure 8: Zero-shot RL success vs correlation between predicted and ground-truth progress

Table 2: Quantitative summary of failure modes from qualitative review of real-robot and simulation rollouts.

Primary qualitative error mode	**SOLE-R1**	**GPT-5**	**Gemini-3-Pro**	**Gemini Robotics-ER**
Temporal under-detection of contact / state change	34%	14%	17%	19%
Ambiguous object state under partial observability	29%	11%	13%	15%
Relies on goal-consistent appearance cues	15%	18%	20%	21%
Perceptual hallucination of success	9%	42%	38%	35%
Other / unclassified	13%	15%	12%	10%

conservatively in these cases (low p_t), which reduces reward hacking but can prevent the agent from receiving positive reinforcement for near-miss states that are essential stepping stones.

3. *Over-reliance on goal-consistent appearance cues.* Rarely, SOLE-R1 assigns partial progress based on visual cues that correlate with success (e.g., the gripper near a handle, an object aligned with a receptacle) even when the underlying subgoal is not achieved (handle not pulled, object not released). Unlike baseline VLMs, these errors typically saturate at *moderate* progress and do not produce confident "100%" completion, but they can still bias exploration toward non-productive states.

In contrast, baseline VLM rewarders fail predominantly via **perceptual hallucination**: they often infer success even when observations contradict it (e.g., interpreting occlusion as lifting, interpreting proximity as grasp). This directly enables reward hacking, where the policy learns to place the camera or objects into configurations that elicit confident success statements without completing the task (Figure 4 and 7).

D Ablations

We ablate SOLE-R1 to isolate components responsible for reasoning that (i) is robust to reward hacking and (ii) provides usable learning signal for unseen tasks. We evaluate three ablations that remove (A) explicit natural language CoT reasoning during training, (B) authentic negative / non-expert trajectories from robot video progress prediction training, and (C) foundational spatiotemporal reasoning data used to induce general spatial and multi-frame temporal grounding.

Evaluation protocol. Unless otherwise stated, all ablations share the same backbone VLM, training protocol, compute budget, and online RL algorithm. We re-run the full zero-shot online RL benchmark suite from Section 5.3 with identical hyperparameters and compute (i) task success rate, (ii) perceived-vs-true success scatter (Figure 5), and (iii) reward/ground-truth alignment metrics. We also report the

fraction of failures categorized as *reward-hacking* (high perceived / low true) versus *signal-limited* (low perceived / low true) using the same quadrant thresholds as Section 5.4.

Ablation 1: No CoT reasoning in progress training. We remove the `<think>` channel from progress training. The model is trained to directly emit only `<answer>` (the scalar progress) at each timestep, without generating per-timestep natural-language explanations:

$$y_t = [\texttt{<answer>}\ p_t\ \texttt{</answer>}].$$

Generating temporally grounded rationales acts as an auxiliary task that forces the model to attend to state changes (contacts, releases, articulation transitions) rather than shortcut correlations. This should improve calibration and reduce variance in dense progress estimates, especially under occlusion or viewpoint shift.

Results. Removing CoT consistently degrades learning across tasks and shifts performance toward the *signal-limited* failure type (Figure 5): the reward becomes flatter and noisier, yielding weaker incremental credit assignment. Qualitatively, thinking-ablated models often under-react to small but task-critical transitions (e.g., closing fingers to grasp an object), producing progress curves with long plateaus and delayed spikes.

Ablation 2: Expert-only robot-video progress supervision. We remove authentic unsuccessful / non-expert trajectories from the robot-video progress dataset, training only on near-expert rollouts. However, we keep the artificial non-expert trajectories, generated by randomly reversing the video at random moments. This augmentation prevents the model from learning to predict monotonically increasing progress values, but is limited in the diversity of failure states it can mimic.

In online RL, the agent frequently visits off-distribution states and executes partial, incorrect, or exploitative behaviors. Training only on successful trajectories can lead to overly optimistic extrapolation: states that are merely *goal-adjacent* (gripper near handle, object aligned with receptacle) may be scored as progress even when no causal subgoal is achieved. We hypothesize that authentic negative and recovery trajectories (generated by inserting real random actions within expert trajectories as described in Section 3.1) are critical for familiarizing the model with real failure states and helping it learn counterfactual distinctions ("looks close" vs. "is achieved") in a way that supports robustness to distribution shift.

Results. Expert-only training substantially increases the *reward-hacking* failure types (Figure 5). The learned reward becomes easier to exploit: agents discover configurations that mimic visual correlates of success (e.g., hovering near target objects, aligning poses, occluding failure) that trigger inflated progress predictions. On the perceived-vs-true plot, these failures move from the lower-left quadrant (signal-limited) toward the lower-right quadrant (reward hacking), indicating higher predicted progress without corresponding true success. This mirrors the baseline VLM rewarder pathology, though typically less extreme than GPT-5/Gemini.

Ablation 3: Remove general spatial & multi-frame reasoning data. We remove the general spatial reasoning and multi-image temporal reasoning datasets used in the foundational stage (Section 3), and train only on embodied reasoning and planning, along with robot-video progress estimation data.

Robot progress estimation requires recognizing subtle spatial relations (contact, containment, alignment, occlusion) and temporally local changes. Without broad spatial + multi-frame reasoning pretraining, the model may rely on dataset-specific appearance cues and fail under new scenes, viewpoints, embodiments, and task families. We hypothesize that foundational reasoning data improves generalization and reduces reliance on spurious correlations that online RL can exploit.

Results. Removing foundational reasoning data reduces overall success and again increases *reward-hacking* failure types. This ablation produces a model that is more sensitive to camera viewpoint and scene texture changes: the agent can more easily induce states that resemble training-time success patterns while failing at the underlying task (e.g., partial occlusion producing "hallucinated" grasp).

Summary. Removing CoT primarily harms *signal quality* (more signal-limited failures), while removing authentic negative trajectories or foundational reasoning data primarily harms *robustness* (more reward-hacking failures). These results support our central design: (i) temporally grounded language reasoning is an effective scaffold for dense progress prediction, and (ii) broad spatiotemporal grounding plus hard negatives are essential for resisting exploitation in zero-shot online RL.

E Effect of RLVR beyond SFT

We isolate the contribution of reinforcement learning with verifiable rewards (RLVR) by comparing a model trained with **SFT only** to the full **SFT+RLVR** SOLE-R1 model. Both models share the same backbone, training data, prompts, and inference procedure; the only difference is the additional RLVR stage applied to the latter. Figure 9 reports zero-shot online RL success rates for both variants.

Results. Across environments, adding RLVR yields a clear and consistent improvement in zero-shot RL performance. The SFT-only model is able to provide a usable reward signal on many tasks, indicating that supervised spatiotemporal CoT reasoning alone already induces partial alignment between predicted progress and true task advancement. However, its performance is less reliable than the full model. In contrast, the SFT+RLVR model achieves consistently higher success rates, solving more tasks and reaching higher peak performance on tasks that are partially solvable with SFT alone. This gap highlights a key limitation of pure SFT for reward modeling: during supervised training, the scalar progress value in the `<answer>` channel constitutes only a small fraction of the output tokens, and its numerical accuracy is weakly emphasized relative to generating fluent reasoning text. RLVR directly addresses this limitation by concentrating learning signal on the correctness and calibration of the predicted progress value. Importantly, RLVR operates on top of the strong spatiotemporal reasoning induced by SFT, refining *how much* progress is assigned without degrading the underlying reasoning structure.

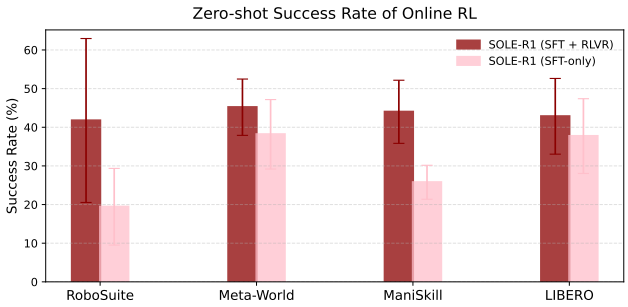


Figure 9: SFT-only vs SFT+RLVR model in zero-shot success rate of online RL

F Full related work

On-robot RL Applying reinforcement learning directly on physical robots has long been challenging due to sample inefficiency, safety concerns, and the difficulty of specifying robust reward functions. Seminal works in this area rely on carefully engineered rewards and resets in the real world [46, 25], massively parallelized actors [26] and asynchronous learning algorithms [15], spatial inductive biases encoded into neural networks [71, 61] and residual learning on top of planner trajectories [70], and reward learning by human-in-the-loop labeling [53].

Recent on-robot learning works increase sample efficiency and the complexity of solvable tasks by highly tuned and efficient off-policy learning implementations [31], human-in-the-loop learning [32], behavior priors in mobile manipulation [39] and shaped reward function learning [65, 5]. In parallel, the emergence of generalist robot policies trained on large-scale datasets [7, 43, 22, 6] has shifted attention toward methods that fine-tune, refine or steer such policies using RL during deployment [75, 38, 40, 11, 17, 3, 60, 2, 24].

Despite these advances, nearly all existing approaches still rely on either manually specified reward functions or human-provided supervision. Our primary from-scratch protocol uses no ground-truth rewards, demonstrations, or task-specific reward-model tuning. Separately, Section 5.9 tests SOLE-R1 rewards for RL post-training of a VLA already fine-tuned on target-task demonstrations.

Vision-language reward learning With the rise of vision-language models, recent work has explored using VLMs as sources of reward supervision. Preference-based methods query VLMs for

pairwise comparisons or ratings to learn reward functions [62, 58, 34, 51]. Other approaches directly score individual images or videos to produce sparse or dense rewards [13, 47, 4, 65, 1, 66, 59, 48].

Several recent methods are especially relevant to our setting. RoboReward [23], Robometer [27], and LRM [63] train vision-language models to predict task progress from robot videos. However, these models typically (i) do not produce explicit intermediate reasoning, (ii) are trained primarily on expert or near-expert trajectories, and (iii) are not evaluated as the *sole* reward signal for online RL starting from random policies. Our experiments show that these limitations make them vulnerable to reward hacking and miscalibration under distribution shift.

Temporal supervision and counterfactual trajectories. Several works exploit temporal structure in videos to derive supervision without explicit rewards. Using temporal order as a proxy for progress has been explored for value learning and reward relabeling [36, 73], and related ideas appear in non-robot domains such as video understanding and representation learning. However, prior methods generally ignore the semantic content of the trajectory or treat time as a weak ordinal signal.

In contrast, SOLE-R1 combines temporal-order supervision with explicit spatiotemporal reasoning grounded in video evidence. By training on both authentic non-expert trajectories and counterfactual regressions, our model learns to distinguish genuine causal progress from visually goal-adjacent but incorrect states, which is critical for robustness in online RL.

Process-level and reasoning-based reward models. Outside robotics, reinforcement learning from learned reward models has been extensively studied for post-training large language models [28, 50, 16, 30, 33]. These works highlight the importance of process-level supervision and verifiable rewards for stabilizing learning. Within robotics, Robo-Dopamine [55] learns a process reward model and adapts it to a target task using one expert demonstration in its standard RL pipeline. Our primary protocol uses reward models without target-task demonstrations or adaptation. Appendix O reports both unadapted and one-demonstration Robo-Dopamine results, explicitly separating these conditions.

G Reinforcement learning agent details

Our RL agent is a PyTorch re-implementation of a DrQv2 [67] agent from SERL [31] (JAX). For tasks in Meta-World and real world, we adapt a Soft Actor-Critic implementation to match the SERL implementation. For tasks in RoboSuite, LIBERO, and ManiSkill, we implement a DrQv2 agent that builds upon prior work from [12] that previously showed success for tasks based in the RoboSuite simulation environment. We train the actor and critic from scratch, without any pre-training or replay buffer demonstrations. Our hyper-parameters are listed in Table 3.

Table 3: Reinforcement learning agent details

DrQv2	Meta-World	RoboSuite, LIBERO, ManiSkill	Real-world
Buffer size	200000	200000	200000
Batch size	256	256	256
Critic to actor steps	4	4	4
Initial random steps	1000	2500	100
Start learning at	1000	2500	100
Critic ensemble size	10	10	10
Critic subsample size	2	2	2
Discount	0.96	0.99	0.9
Exploration schedule	Entropy tuning with $\alpha = 10^{-2}$	std=linear(1.0, 0.1, 500K)	std=linear(1.0, 0.1, 15k)
Learning rate	10^{-3}	10^{-4}	10^{-3}
Soft target update rate	$5 * 10^{-3}$	$5 * 10^{-3}$	$5 * 10^{-3}$
Learner steps per actor step	1	2	4

Table 4: Task groups and their natural language specifications

Task Name	Natural Language Task Specification
Real-World Tasks	
Pick can	Pick up the can
Touch strawberry in clutter	Touch the strawberry on the table
Push can to cube	Push the can near the cube
Close drawer	Close the drawer
Open drawer	Open the drawer
Insert pipe	Insert the pipe into the opening
Meta-World Tasks	
faucet-open	Open the faucet
faucet-close	Close the faucet
window-open	Open the window
window-close	Close the window
plate-slide (puck in net)	Slide the puck into the net
drawer-close	Close the drawer
drawer-open	Open the drawer
door-close	Close the door
door-open	Open the door
button-press	Press the button
handle-pull	Pull the handle
soccer	Push the soccer ball into the goal
lever-pull	Pull the lever up
coffee-button	Press the gray coffee machine button
handle-press	Press the handle down
coffee-push	Push the coffee mug toward the coffee machine
RoboSuite Tasks	
Lift	Pick up the cube from the table
Wipe	Wipe the dark ink off of the table
Door	Turn the door handle and pull the door open
PickPlaceCan	Pick up the can and place it at a target location
LIBERO Tasks (Environment Task ID)	
Close microwave (KITCHEN_SCENE6_close_the_microwave)	Close the microwave door
Open microwave (KITCHEN_SCENE7_open_the_microwave)	Open the microwave door
Turn on stove (KITCHEN_SCENE3_turn_on_the_stove)	Turn the stove knob to the on position
Turn off stove (KITCHEN_SCENE8_turn_off_the_stove)	Turn the stove knob to the off position
Close top drawer (SCENE10_close_the_top_drawer_of_the_cabinet)	Close the top drawer
Open bottom drawer (SCENE1_open_the_bottom_drawer_of_the_cabinet)	Open the bottom drawer
ManiSkill Tasks (Environment Task ID)	
PickPanda (PickCube-v1)	Pick up the cube from the table
PushCube (PushCube-v1)	Push the cube into the target area
PullCube (PullCube-v1)	Pull the cube into the target area
Mobile-OpenCabDrawer (OpenCabinetDrawer-v1)	Open the cabinet drawer
PickSO100 (PickCubeSO100-v1)	Pick up the cube from the table
PickWidowXAI (PickCubeWidowXAI-v1)	Pick up the cube from the table
PickSingleYCB (PickSingleYCB-v1)	Pick up the object from the table

G.1 Evaluation protocol and success rate computation

We compute task-level success from ground-truth task predicates, used only for evaluation and logging and never exposed to the policy or reward model during training. From-scratch training and evaluation use fixed horizons: 25 reward timesteps in simulation and 20 in the real world. Predicted success never terminates an episode. Evaluation counts ground-truth success at any timestep.

Evaluation episodes. For periodic evaluation, we typically run $N = 50$ independent episodes in simulation and $N = 10$ in the real world, initialized from the standard task reset distribution. From-scratch experiments use fixed horizons of 25 reward timesteps in simulation and 20 in the real world during both training and evaluation. During evaluation, exploration noise is disabled and the policy acts deterministically. The VLA final evaluations use 100 rollouts per task (Section 5.9); the temporal-input comparison uses the protocol in Appendix P.1.

Success definition. A rollout is considered *successful* if the task-specific ground-truth success condition is satisfied at *any* timestep during the episode. These conditions are provided by the

simulator (for RoboSuite, ManiSkill, Meta-World, and LIBERO) or by manually defined, externally measured criteria in the real world (e.g., drawer closed within a tolerance, object lifted off the table, object reaching a target region). Importantly, these success signals are used *only* for evaluation and logging, and are not accessible to the policy or the reward model during training.

Binary episode outcome. Let $s_i \in \{0, 1\}$ denote the binary success outcome of evaluation episode i, where $s_i = 1$ if the success condition is met at any timestep and $s_i = 0$ otherwise. We do not require the success condition to persist until the end of the episode; transient successes (e.g., briefly lifting an object) are counted as success, consistent with prior work in online manipulation RL.

Success rate. The success rate for a task is computed as

$$\text{SuccessRate} = \frac{1}{N} \sum_{i=1}^{N} s_i.$$

For the from-scratch benchmark, we use three training seeds in simulation and one in the real world, and report the mean success rate across seeds.

H Vision-language model inference details

H.1 Inference speed

We send a batch of 2 videos of 20 frames each to SOLE-R1, which is hosted on a node with a single H100, and to GPT-5 and Gemini-3-Pro using public APIs. We set the Gemini-3-Pro thinking setting to "LOW". On average, SOLE-R1 takes 40 seconds to annotate rewards (1 s per frame), GPT-5 takes 366 seconds (9.15 s per frame) and Gemini-3-Pro takes 322 seconds (8.05 s per frame). All three models produce a chain of thought for each frame, and the inference speed may vary according to the output length each model chooses to produce. We note that the speed of the public APIs varies day by day.

H.2 Prompts

Each model receives a system prompt followed by a user prompt. A sequence images follows. In user prompt, we replace "{task_description}" with the task description and "{prev_progress}́ with the previous model prediction. Since we give the model its previous prediction, we have to perform inference sequentially.

H.2.1 SOLE-R1

SYSTEM PROMPT TEMPLATE: You are an expert roboticist with the goal of predicting task progress percentages given frames from a video of a robot attempting to complete a task. You first think, in the form of an internal monologue, before providing your final answer. Your reasoning process MUST BE enclosed within <think> </think> tags and should include detailed reasoning. Your final answer MUST BE enclosed within <answer> </answer> tags and should be a integer (positive or negative) representing current task progress percentage. "Example output format: <think>[detailed reasoning process]</think><answer>[current task progress]%</answer>

USER QUESTION TEMPLATE: Here is an image containing multiple camera views of a robot attempting to complete a task. The views on the top are from an external camera. The views on the bottom are from the robot's wrist camera. The views from the very first timestep are shown to the left. The views from the previous timestep are shown in the middle. The views from the current timestep are shown to the right. The task description is: {task_description}. The task progress for the very first timestep is 0%. The task progress for the previous timestep is {prev_progress}%. Predict the task progress for the current timestep.

H.2.2 GPT-5

SYSTEM PROMPT: None.

USER PROMPT TEMPLATE GPT: Here is an image showing multiple timesteps of a robot attempting to complete a task. The first image is from the previous timestep. The second image is from the current timestep. The task description is: {task_description}. The predicted task progress for the previous timestep was {prev_progress}%. Predict the task progress for the current timestep. Before providing your final answer, first briefly provide a few sentences that reason about what is happening at the current timestep relative to the previous timestep. Your reasoning process should be no more than a few sentences and MUST BE enclosed within <think> </think> tags. Please refer to the images as timesteps instead of as separate images. Your final answer MUST BE enclosed within <answer> </answer> tags and should be an integer representing current task progress percentage. Example output format: <think>[detailed reasoning process]</think><answer>[current task progress]%</answer>

H.2.3 Gemini-3-Pro and Gemini Robotics ER 1.5

SYSTEM PROMPT: You are an expert roboticist with the goal of predicting task progress percentages given frames from a video of a robot attempting to complete a task.

USER PROMPT TEMPLATE GEMINI: Here is an image showing multiple timesteps of a robot attempting to complete a task. The first image is from the previous timestep. The second image is from the current timestep. The task description is: {task_description}. The predicted task progress for the previous timestep was {prev_progress}%. Predict the task progress for the current timestep. Note that the previous progress value might not be accurate. Please carefully assess the images to determine the correct progress for the current timestep. Also note that the performance of the robot is unknown so progress can increase or decrease at any timestep. Before providing your final answer, first briefly provide a few words that reason about what is happening at the current timestep relative to the previous timestep. Your reasoning process should be no more than one or two sentences and MUST BE enclosed within <think> </think> tags. IMPORTANT: Do not refer to 'the user' in your reasoning. Only reason as though it is an internal monologue thinking about the robot. If you are unsure about the progress, please try to avoid overestimation and provide a conservative estimate. Please refer to the images as timesteps instead of as separate images. Your final answer MUST BE enclosed within <answer> </answer> tags and should be an integer representing current task progress percentage. Example output format: <think>[detailed reasoning process]</think><answer>[current task progress]%</answer>

H.2.4 Testing different prompting strategies for GPT-5 and Gemini

Figure 10 compares several prompting strategies for GPT-5 and Gemini in zero-shot online RL, varying the camera viewpoints and temporal context provided to the model. We find that prompting with the *external camera view only*, using the *current and previous* timesteps, consistently yields higher task success than alternatives that include (i) wrist-view images, (ii) a combination of external and wrist views, or (iii) longer temporal context including the first timestep. This result suggests that, for general-purpose VLM rewarders, external views provide more reliable and less ambiguous cues for task progress than wrist-mounted views, which are frequently occluded or dominated by the robot end-effector. Additionally, providing longer temporal histories does not improve performance and can degrade it, likely due to increased visual clutter and reduced salience of task-relevant changes. All baseline results in the main paper therefore use the external-view (current + previous) prompting configuration.

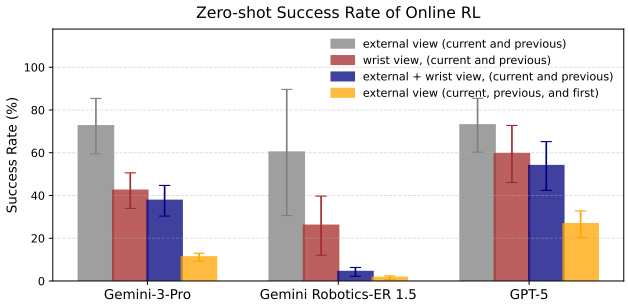


Figure 10: Zero-shot success rate of online RL on 10 simple Meta-World tasks across varying prompting strategies.

I Real robot experiment details

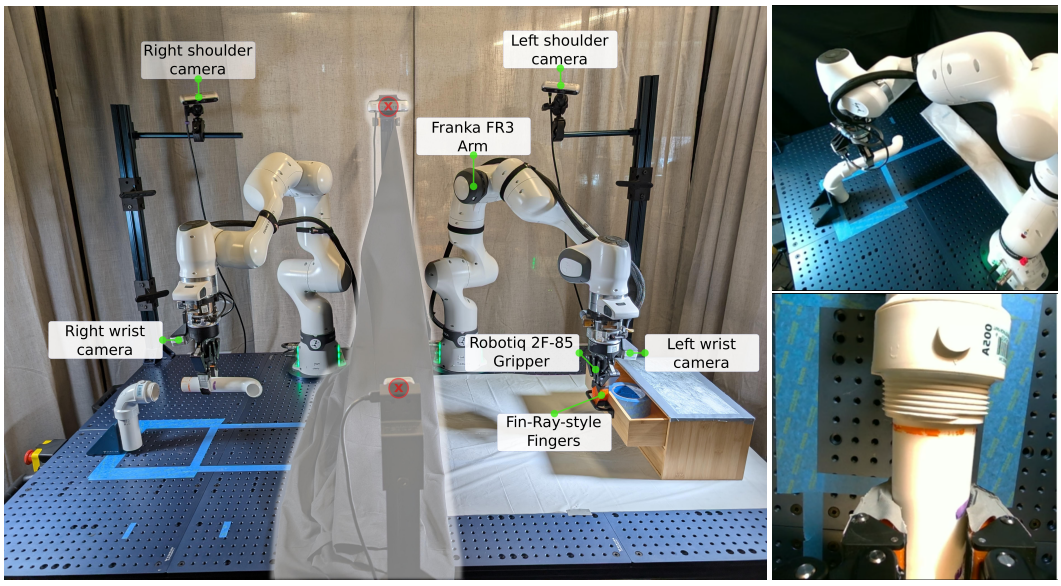


Figure 11: Left: workcell with two Franka FR3 arms. Right: example shoulder and wrist views from the right robot.

We conduct visuomotor reinforcement learning (RL) experiments on real tabletop Franka FR3 arms. Our system consists of an RL actor, which executes actions on the robot, an RL learner, which performs gradient descent based on collected data, and a reward labeling server, which provides reward annotations for collected episodes. The RL actor, RL learner and reward labeling server run as independent processes and communicate asynchronously via ZeroMQ. On average, our system executes around 1.5k RL actions and 6k RL learner steps per hour. We reset the environment manually; the arms are reset using either an automatic motion plan or manually via SpaceMouse teleoperation. There is no extrinsic reward signal, the only supervision comes from our reward labeling server, which uses either our SOLE-R1 vision-language model (VLM) or other baseline VLMs. We either self-host VLMs on nodes equipped with a single H100 GPU or use public APIs.

Hardware configuration We conduct our experiments on workstations with a pair of Franka FR3 arms mounted on a shared tabletop (Figure 11). The Franka end-effectors are equipped with a wrist-mounted RealSense D405 camera, a force-torque sensor, which is not used in our experiments, and a Robotiq 2F-85 gripper with Fin-Ray-style compliant fingers. While there are multiple cameras mounted to the workstation, we only use a pair of cameras looking over the shoulder of each of the robots and the pair of wrist-mounted cameras. We run RL experiments independently on each arm.

Observation space Our observation space combines visual inputs and proprioception. We record RGB images from a wrist-mounted camera and a over-the-shoulder camera. The wrist cameras are RealSense D405s and the shoulder cameras are D455s. We crop and rescale these images to (384, 384) for reward prediction and to (84, 84) for action prediction. We add barriers so that only one robot is seen at a time. We add proprioceptive state in the form of end-effector position (x, y, z) and gripper width.

Action space The RL actor commands Cartesian end-effector position offsets in (x, y, z) coordinates in the interval [-1, 1]. We internally rescale these actions to [-20 cm, +20 cm] for each axis. We set a fixed gripper orientation for each task and do not allow the RL actor to change it (conversely, we use the full $SE(3)$ action space in simulation). Depending on the task, the RL actor can also command gripper actions scaled between [-1, 1]. We threshold these actions as fully open [1, 0] and fully closed (0, -1]. While we have the ability to continuously command the gripper width, this mode is not conducive to exploration. On average, the RL actor half-closes the gripper, which often prevents it from grasping objects. We therefore use binary gripper control.

Control stack Our implementation combines an RL control loop and an internal robot control loop. Firstly, the RL actor interacts with the robot environment at 1 Hz: we execute an action, wait 1 s and record the next observation. Secondly, the internal robot control loop executes RL actions at 100 Hz using linear interpolation for positions and spherical linear interpolation (slerp) for quaternion orientations. Each RL action is interpolated into a 1 s trajectory with 100 points. Then, we perform `pink` inverse kinematics and execute the resulting joint-space actions using a hybrid joint impedance controller implemented in `ros2_control`. A reference implementation of this controller can be found in `polymetis`.

Reward Prediction For real-world tasks, our RL episodes are 20 timesteps long. SOLE-R1 labels all frames whereas GPT-5 and Gemini-3-Pro subsample to 10 frames. Even so, these baselines take more than twice as long to label the videos, resulting in experiment duration of up to 5 hours. We end SOLE-R1 experiments after 2 hours. SOLE-R1 receives a composite frame of the shoulder and wrist view, whereas GPT-5 and Gemini-3-Pro only use the shoulder view. We made this choice because both baselines struggle to understand wrist and composite views (Section H.2.4). In particular, they sometimes state that there are multiple robots in the scene based on the composite view.

I.1 Tasks

Touch strawberry in clutter We place a box, a can and a strawberry randomly in a $20\times20\times20$ cm workspace. The workspace is slightly higher than the ground to prevent the gripper from scraping the table. The robot starts in a fixed pose above the workspace. The task is successful if the gripper touches the strawberry at any point in the episode.

Push can to cube We use the same workspace as in the first task. We place a can on the left side of the workspace and a blue cube on the right side of the workspace. Both object positions vary with small random offsets. The robot starts in a fixed pose above the workspace. The task is successful if the can ends up within 1 cm of the cube.

Pick can We use the same workspace as in the first task. The can is placed approximately in the center with small random offsets. The robot starts in a fixed pose above the workspace. The task is successful if the robot visibly lifts the can off the table at any point in the episode.

Close drawer We use a $10\times20\times5$ cm workspace, we use a small height variation to protect the wrist camera. The cabinet is placed in approximately the same position on the right side of the

workspace and the drawer is open up to a point marked by tape. The agent is successful if it closes the drawer up to about the last 1 cm, marked by tape, at any point in the episode.

Open drawer We use the same workspace as in "close drawer". The drawer starts fully closed. The task is successful if the agent opens the drawer up to a point about 10 cm inside of the drawer marked by tape at any point in the episode.

J Training data synthesis and curation details

This appendix provides additional details on the construction, balancing, and curation of the training data used for SOLE-R1, expanding on the summary in Section 3. Our goal in designing this data mixture is to induce video-native spatiotemporal reasoning that (i) supports dense progress estimation under partial observability, and (ii) remains robust when deployed as the sole reward signal for online reinforcement learning. We provide extensive video demonstrations to illustrate what the outputs of our video synthesis approach look like at: `https://sole-r1.github.io/`

J.1 Design Principles

The data synthesis pipeline is guided by three core principles:

1. **Explicit coverage of partial success and failure states.** Online RL policies frequently visit intermediate, incorrect, or regressive states that are underrepresented in expert demonstrations. To prevent optimistic extrapolation and reward hacking, training data must include authentic non-expert behaviors spanning varying degrees of task completion.
2. **Temporal locality of supervision.** Progress supervision and reasoning are provided at the granularity of individual timesteps, forcing the model to reason about *what changed* between consecutive frames rather than relying on static appearance or final outcomes.
3. **Task-structure-aware decomposition.** Manipulation tasks admit natural decompositions into subgoals (approach, contact, grasp, transport, release, articulation). We explicitly encode this structure in the non-expert trajectory design so that the model learns reusable progress primitives rather than task-specific heuristics.

J.2 Non-expert trajectory level taxonomy

Table 5 enumerates the non-expert trajectory levels used in our synthesis pipeline. Each level corresponds to a prefix of a canonical task decomposition, where all preceding subgoals are assumed to be successfully completed, and the failure or termination occurs at the current level.

For example, in pick-and-place tasks, a level-3 trajectory (*Contact*) assumes the robot has successfully approached the object but fails to establish a stable grasp. This structure ensures that trajectories across levels are *semantically ordered* by task progress, enabling consistent supervision of advance versus regress events.

We allocate approximately uniform probability mass across levels *within each task family*. This prevents overrepresentation of near-expert states and ensures that the model is exposed to both early-stage failures (e.g., failure to approach) and late-stage failures (e.g., dropping an object after grasp). Uniform per-level sampling was empirically found to reduce optimistic progress extrapolation compared to expert-heavy mixtures.

J.3 Simulation-based trajectory synthesis

For simulation environments (RoboCasa), non-expert trajectories are generated by injecting random action deviations into expert demonstrations, as described in Section 3.1. Each deviation produces either: (i) a terminal failure trajectory, or (ii) a recovery trajectory that interpolates back to a downstream expert state.

Crucially, injected deviations are sampled across the full trajectory, not only near the start. This yields failures that occur after partial task completion (e.g., grasp achieved but object dropped), which are

particularly important for teaching the model to distinguish true completion from visually similar near-success states.

Progress supervision is derived from ground-truth simulator geometry and normalized per-trajectory, ensuring that progress values are comparable across different deviation points and trajectory lengths.

State interpolation can produce physically infeasible recovery segments, including objects moving back toward the gripper after a drop or motion through obstacles. Possible safeguards include adjusting deviation timing, rejecting disallowed collisions, and checking simulator-state compatibility before recovery. These safeguards are proposed improvements, not guarantees of the current synthesis procedure.

J.4 Real-world video perturbations

For real-world videos (OXE), where simulator state is unavailable, we synthesize non-expert behavior directly in observation space via temporal perturbations. Temporal reversal windows induce visually plausible regressions (e.g., an object moving away from a goal configuration) while preserving realistic appearance statistics.

By inheriting progress labels from the corresponding expert timestep, reversed segments are explicitly labeled as negative progress, anchoring the model's reasoning about regression events. This strategy avoids relying on heuristic visual similarity alone and enforces a consistent temporal ordering signal across all real-world data.

Naturally occurring real-world failures could capture behaviors absent from temporal reversals, but obtaining dense progress labels without simulator state or extensive annotation remains challenging. Future work could combine sparse human labels with automatic densification, proprioception or object tracking, and human correction of uncertain model-proposed labels.

J.5 Grounded chain-of-thought generation

For each timestep, we generate a chain-of-thought (CoT) explanation describing: (i) salient visual changes since the previous timestep, (ii) whether those changes advance or regress the specified goal, and (iii) the next implied subgoal.

In simulation, CoT traces are initially templated using ground-truth distance and contact signals, then paraphrased using foundation vision-language models to increase linguistic diversity while preserving factual grounding. Once these geometric quantities are available for progress supervision, templated descriptions reuse them to ground statements about state changes. Appendix P.3 specifies a comparison that removes the templates while retaining distance-based grounding, as well as an annotation variant without privileged-state grounding.

In real-world videos, CoT traces are generated directly by a foundation VLM conditioned on the temporal context and the known progress direction. Progress supervision constrains the explanation to remain temporally consistent (e.g., reversed segments must be described as regression), mitigating hallucinated success narratives.

J.6 Dataset mixture and balancing

The full dataset mixture is summarized in Table 6. We group data into three high-level categories:

- **Video-based CoT reasoning and progress prediction**, which directly supervises the SOLE-R1 reward signal.
- **Embodied reasoning and planning**, which provides object-centric and task-structured reasoning grounded in manipulation semantics.
- **General spatial and temporal reasoning**, which supplies broad visual grounding across viewpoints, textures, and domains.

Although the raw counts differ substantially across sources, we do not sample proportionally to dataset size. Instead, we apply category-level balancing during SFT (Section K), ensuring that progress prediction data remains a substantial fraction of each batch without overwhelming general reasoning capabilities.

Table 5: Levels for non-expert trajectories and video dataset counts. For each level, we assume that all task components corresponding to the preceding levels have already been successfully completed.

	Non-expert trajectory levels	**Percentage**
Pick-only	1: Fail to approach {obj}	25%
	2: Approach {obj}	25%
	3: Contact {obj}	25%
	4: Pick up {obj}	25%
Pick-and-place	1: Fail to approach {obj}	14%
	2: Approach {obj}	14%
	3: Contact {obj}	14%
	4: Pick up {obj}	14%
	5: Keep grasp of {obj}	14%
	6: Approach placing {obj} in {target_location}	14%
	7: Place {obj} in {target_location}	14%
Open/close doors/drawers	1: Fail to approach the door/drawer	20%
	2: Approach door/drawer	20%
	3: Contact door/drawer	20%
	4: Start opening/closing door/drawer	20%
	5: Finish opening/closing door/drawer	20%
Turn on/off appliances	1: Fail to approach the on/off button/lever	33%
	2: Approach the on/off button/lever	33%
	3: Successfully adjust the on/off button/lever	33%

J.7 Why authentic non-expert data matters

Ablation results in Section D show that removing authentic non-expert trajectories (while keeping synthetic reversals) significantly increases reward-hacking failures. This suggests that visually realistic but *causally incorrect* states—produced only by real random deviations—are critical for teaching the model distinctions such as:

- "near the handle" vs. "handle actuated",
- "object aligned" vs. "object released",
- "contact" vs. "stable grasp".

These distinctions are frequently exploited by online RL agents and are difficult to learn from expert-only or temporally monotonic data.

J.8 Summary

In summary, the training data synthesis pipeline for SOLE-R1 combines: (i) structured non-expert trajectory levels aligned with task decompositions, (ii) simulation-grounded progress supervision, (iii) observation-space perturbations for real-world video, and (iv) dense, temporally grounded chain-of-thought reasoning. Together, these design choices produce a dataset that explicitly exposes the model to the states most likely to be encountered—and exploited—during zero-shot online reinforcement learning.

Table 6: Dataset mixture

Data source ID	Total Dataset Count
Video-based CoT Reasoning and Progress Prediction	
OXE berkeley_autolab_ur5	4874
OXE berkeley_fanuc_manipulation	2689
OXE bridge	4602
OXE bridge_set2	32596
OXE bridge_set3	51939
OXE fractal20220817_data	9076
OXE fractal20220817_data_set2	44230
OXE fractal20220817_data_set3	23362
OXE fractal20220817_data_set4	35040
OXE jaco_play	4106
OXE jaco_play_set2	2140
OXE nyu_door_opening_surprising_effectiveness	4211
OXE ucsd_kitchen_dataset_converted_externally_to_rlds	1060
RoboCasa CloseDrawer	79328
RoboCasa OpenDrawer	80436
RoboCasa OpenSingleDoor	43601
RoboCasa PnPCabToCounterPickOnly	22057
RoboCasa PnPCabToCounter	62951
RoboCasa PnPCounterToCabPickOnly	27753
RoboCasa PnPCounterToCab	68311
RoboCasa PnPCounterToMicrowavePickOnly	25853
RoboCasa PnPCounterToMicrowave	67014
RoboCasa PnPCounterToSinkPickOnly	24995
RoboCasa PnPCounterToSink	55767
RoboCasa PnPCounterToStovePickOnly	25494
RoboCasa PnPCounterToStove	61348
RoboCasa PnPMicrowaveToCounterPickOnly	26006
RoboCasa PnPMicrowaveToCounter	63625
RoboCasa PnPSinkToCounterPickOnly	26767
RoboCasa PnPSinkToCounter	55855
RoboCasa PnPStoveToCounterPickOnly	26974
RoboCasa PnPStoveToCounter	68406
RoboCasa TurnOnMicrowave	52214
RoboCasa TurnOffMicrowave	51093
Embodied Reasoning and Planning	
fsd_level_1_2_3	279739
fsd_level_4_5	42074
robovqa	258280
robo2vlm1	678034
robo2vlm1_reasoning	4635
ecot_libero_all	504544
ecot_bridge_all	1346416
General Spatial and Temporal Reasoning	
spaceom	806
spacethinker	12412
spot_the_diff	12562
ssrcot_llavacot100k	98522
ssrcot_spatialqa_chunk1	166639
ssrcot_spatialqa_chunk2	166639
ssrcot_spatialqa_chunk3	166640
ssrcot_visualcot_chunk1	141295
ssrcot_visualcot_chunk2	141296
ssrcot_vocot_chunk1	158594
ssrcot_vocot_chunk2	158594

K SOLE-R1 training details

This appendix provides full details of the SOLE-R1 model training procedure, expanding on the summary given in Section 5.1. We describe the backbone architecture, input/output formatting, supervised fine-tuning (SFT) configuration, reinforcement learning with verifiable rewards (RLVR) setup, optimization hyperparameters, and implementation details required for reproducibility.

K.1 Backbone architecture and initialization

SOLE-R1 is initialized from **Qwen3-VL-8B-Instruct**, a vision-language transformer with a ViT-style visual encoder and an autoregressive language decoder. We retain the full multimodal architecture and do not freeze any layers during training. All parameters are updated jointly in both SFT and RLVR stages.

Video inputs are encoded by uniformly sampling frames from the temporal context window described in Section 2.1. Each frame is resized to 384×384 pixels and passed independently through the vision encoder; frame embeddings are concatenated in temporal order and fused with text tokens using the backbone's native cross-attention mechanism. No explicit temporal positional embeddings beyond the backbone defaults are introduced.

K.2 Input and output formatting

All training examples are normalized into a unified multimodal prompt format:

$$x = [\texttt{<goal>}\ g\ \texttt{</goal>},\ \texttt{<video>}\ o_{t-K+1:t}\ \texttt{</video>}],$$

optionally including <prev_progress> p_{t-1}</prev_progress> during training with probability $1 - \delta$, where $\delta = 0.3$.

The model is trained to emit a structured response:

$$y = [\texttt{<think>}\ m_t\ \texttt{</think>}, \texttt{<answer>}\ p_t\ \texttt{</answer>}],$$

where m_t is free-form natural language reasoning and p_t is a scalar progress value represented as a signed integer token in $[-100, 100]$. We discretize progress into integer values during training to ensure stable parsing and reward computation.

K.3 Supervised fine-tuning (SFT)

Training mixture. During SFT, each batch is balanced across three data categories, as stated in Section 5.1: (i) foundational spatial and multi-frame temporal reasoning, (ii) embodied reasoning and planning, (iii) robot video-based progress prediction. We enforce approximate per-batch proportions of 40%, 30%, and 30% respectively, using dataset-level sampling weights.

Objective. We minimize the standard autoregressive negative log-likelihood loss over the entire output sequence:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{|y|} \log p_\phi(y_t \mid x, y_{<t}).$$

No auxiliary losses are applied; in particular, we do not separately supervise the progress scalar outside of the language modeling objective.

Optimization. SFT is run for a single epoch over the full training mixture. We use the AdamW optimizer with parameters: $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay $= 0.1$. The base learning rate is 1×10^{-5} with cosine decay and a 2,000-step warmup. Global batch size is 64, distributed across 8 NVIDIA H100 GPUs.

K.4 Reinforcement learning with verifiable rewards (RLVR)

After SFT, we further train the model using GRPO-based RLVR, focusing exclusively on the robot video progress prediction dataset.

Sampling. For each input query q, we sample $G = 8$ candidate outputs from the frozen SFT policy $p_{\phi_{\text{old}}}$. Sampling uses temperature 0.7 and nucleus sampling with $p = 0.9$. Candidates that fail to produce a parseable `<answer>` token are assigned zero reward.

GRPO parameters. We use $\epsilon = 0.2$ for ratio clipping and $\beta = 0.01$ for KL regularization against the SFT reference model. RLVR is run for 5,000 steps over the progress dataset, with a learning rate of 1×10^{-6}.

K.5 Compute and infrastructure

Training is conducted on clusters of NVIDIA H100 GPUs. SFT requires approximately 1 week on a node of 8 GPUs, while RLVR requires an additional 2 days.

K.6 Reproducibility details

We release: (i) the full training mixture with dataset identifiers and sampling weights, (ii) exact prompt templates and output parsers, (iii) SFT and RLVR training scripts with configuration files, (iv) final model checkpoints after SFT and RLVR. This enables exact reproduction of the training procedure.

L General spatial and vision-language reasoning

In Table 7, we show that SOLE-R1 achieves improved performance on three spatial and general visual reasoning benchmarks relative to the backbone models and the strong SSR spatial reasoning model. The benchmarks include: SpatialBench [64], SSRBench [29], and CV-Bench [76]. We observe consistent gains across all three benchmarks. On SpatialBench, SOLE-R1 improves over the SSR 7B model by +2.4 points and over the backbone models by a larger margin. On SSRBench, SOLE-R1 achieves the strongest performance in both the General and Spatial subsets, outperforming SSR 7B by +3.5 and +3.1 points, respectively, indicating improved generalization and spatial reasoning capability.

Table 7: Performance improvement of SOLE-R1 compared to SSR 7B model and the backbone model on SpatialBench, SSRBench, and CV-Bench.

Method	Size	SpatialBench	SSRBench		CV-Bench
			General	Spatial	
Qwen2.5-VL	7B	64.7	69.4	53.6	73.0
Qwen3-VL	8B	65.3	73.1	57.2	73.7
SSR	7B	67.0	82.1	76.1	73.3
SOLE-R1 (Ours)	8B	69.4	85.6	79.2	74.4

Table 8: Value-Order-Correlation analysis (as done in on [35]) comparing SOLE-R1 to GVL on 1,000 videos from OpenX Embodiment datasets.

Dataset	GVL	SOLE-R1
utokyo_xarm_pick_and_place_converted_externally_to_rlds	0.74	0.92
utokyo_xarm_bimanual_converted_externally_to_rlds	0.83	0.86
nyu_door_opening_surprising_effectiveness	0.69	0.88
berkeley_autolab_ur5	0.77	0.85
utokyo_pr2_tabletop_manipulation_converted_externally_to_rlds	0.65	0.87
maniskill_dataset_converted_externally_to_rlds	0.72	0.84
utokyo_pr2_opening_fridge_converted_externally_to_rlds	0.79	0.88
fractal20220817_data	0.77	0.84
iamlab_cmu_pickup_insert_converted_externally_to_rlds	0.56	0.81
toto	0.55	0.78
ucsd_kitchen_dataset_converted_externally_to_rlds	0.53	0.79
utaustin_mutex	0.60	0.75
asu_table_top_converted_externally_to_rlds	0.46	0.74
austin_sirius_dataset_converted_externally_to_rlds	0.57	0.76
dobbe	0.51	0.73
berkeley_cable_routing	0.49	0.75
berkeley_rpt_converted_externally_to_rlds	0.48	0.73
viola	0.44	0.77
fmb	0.58	0.74
austin_buds_dataset_converted_externally_to_rlds	0.39	0.71
usc_cloth_sim_converted_externally_to_rlds	0.46	0.70
bridge	0.48	0.72
jaco_play	0.42	0.69
stanford_hydra_dataset_converted_externally_to_rlds	0.37	0.71
bc_z	0.41	0.73
berkeley_mvp_converted_externally_to_rlds	0.34	0.67
cmu_stretch	0.31	0.66
tokyo_u_lsmo_converted_externally_to_rlds	0.35	0.65
berkeley_fanuc_manipulation	0.27	0.63
roboturk	0.32	0.62
ucsd_pick_and_place_dataset_converted_externally_to_rlds	0.26	0.59
dlr_edan_shared_control_converted_externally_to_rlds	0.15	0.57
dlr_sara_pour_converted_externally_to_rlds	0.10	0.55
droid	0.04	0.53
taco_play	0.09	0.56
stanford_robocook_converted_externally_to_rlds	-0.02	0.51
imperialcollege_sawyer_wrist_cam	-0.01	0.46
kaist_nonprehensile_converted_externally_to_rlds	-0.05	0.41
austin_sailor_dataset_converted_externally_to_rlds	-0.07	0.38
kuka	0.28	0.35
cmu_play_fusion	-0.08	0.33
stanford_kuka_multimodal_dataset_converted_externally_to_rlds	-0.14	0.30
nyu_franka_play_dataset_converted_externally_to_rlds	-0.16	0.29
stanford_mask_vit_converted_externally_to_rlds	-0.11	0.27
uiuc_d3field	-0.20	0.18
robo_net	-0.24	0.12
columbia_cairlab_pusht_real	-0.26	0.09
dlr_sara_grid_clamp_converted_externally_to_rlds	-0.28	-0.01
cmu_franka_exploration_dataset_converted_externally_to_rlds	-0.25	-0.04

M Details of zero-shot scaling experiments

This appendix provides additional details for the *Zero-shot Scaling* experiment reported in Section 5.6. The goal of this experiment is to isolate how *training-task diversity* in the video-based progress

supervision affects downstream zero-shot online RL performance, while holding model architecture, optimization, and total training compute fixed.

M.1 Training task-type taxonomy

To study scaling with respect to task diversity, we group all robot-video progress supervision into a set of **seven canonical task types**. Each task type corresponds to a distinct manipulation primitive with characteristic spatiotemporal progress structure, contact dynamics, and failure modes. The full list is:

1. **PickOnly**: Grasping and lifting a single object from a support surface, without placement.
2. **Pick-and-Place (Pick $\rightarrow$ Counter)**: Picking an object from a receptacle (e.g., cabinet, microwave, sink, stove) and placing it onto a counter or open surface.
3. **Pick-and-Place (Counter $\rightarrow$ Receptacle)**: Picking an object from a counter and placing it into a target receptacle (e.g., cabinet, microwave, sink).
4. **Open/Close Drawer**: Articulated manipulation involving sliding drawers, including approach, contact, actuation, and completion states.
5. **Open/Close Door**: Articulated manipulation of hinged doors, requiring handle interaction and rotational motion.
6. **Button-Press**: Discrete actuation of buttons or switches (e.g., microwave, coffee machine), characterized by brief contact events and state changes with minimal visual motion.
7. **Lever / Knob Actuation**: Continuous rotational or pulling actions on levers or knobs, including turning appliances on or off.

M.2 Scaling protocol

We construct a sequence of SOLE-R1 variants by progressively increasing the number of task types included in the *video-based progress prediction* portion of the training data. Concretely:

- All models share the same backbone (Qwen3-VL-8B-Instruct), training recipe (SFT + RLVR), optimizer settings, and total number of training steps.
- Foundational spatial reasoning and multi-frame temporal reasoning data are held constant across all variants.
- Only the *set of task types* used to generate robot-video progress supervision is varied.

M.3 Evaluation metric

Each trained model is evaluated on the full zero-shot online RL benchmark described in Section 5.3, comprising tasks *not used* for training progress supervision. For each model, we report the number of downstream tasks achieving success rates above fixed thresholds (e.g., 10%, 30%, 50%). Figure 6 plots the number of tasks solved above threshold as a function of training task-type diversity.

N Additional VLA post-training details

The four target tasks in Table 1 are `KITCHEN_SCENE3`: turn on the stove and put the pot on it; `KITCHEN_SCENE4`: put the black bowl in the bottom drawer and close it; `LIVING_ROOM_SCENE1`: put alphabet soup and cream cheese in the basket; and `STUDY_SCENE1`: place the book in the caddy compartment. The OpenVLA-OFT initialization uses 50 demonstrations per target task; the subsequent RL stage uses only SOLE-R1 rewards.

In the observed post-training runs, performance remained stable during continued optimization, and qualitative rollout inspection did not reveal reward hacking. The observed improvements primarily involved more reliable grasping and fewer dropped objects. These observations concern the evaluated runs and do not establish immunity to reward exploitation.

O Robo-Dopamine with and without task adaptation

Our main comparison requires reward models to operate from task language and observations without target-task demonstrations or tuning. Robo-Dopamine [55] uses one-demonstration adaptation in its standard RL pipeline. We therefore report its unadapted (zero-shot) and one-demonstration conditions separately on three tasks spanning RoboSuite, Meta-World, and ManiSkill, averaged across 3 seeds each (Table 9). SOLE-R1 receives no target-task demonstration or reward-model adaptation.

Table 9: Policy success rates (%) for Robo-Dopamine and SOLE-R1. The one-demonstration condition uses additional target-task supervision for reward-model adaptation. Means are rounded to the nearest integer.

Reward model and setting	RoboSuite pick cube	Meta-World open window	ManiSkill push cube	Mean
Robo-Dopamine: zero-shot	0	0	0	0
Robo-Dopamine: one demonstration	55	32	41	43
SOLE-R1: zero-shot	**94**	**72**	**63**	**76**

One-demonstration adaptation improves Robo-Dopamine over its unadapted condition, but its success remains below SOLE-R1 on all three tasks. Qualitative rollout inspection identified reward-hacking behavior even after adaptation. These observations describe this evaluation setting.

P Additional controlled comparisons

P.1 Temporal inputs with matched data and CoT supervision

We independently train three SFT-only reward models from the same Qwen3-VL-8B initialization using the same synthesized data, CoT supervision, and training duration. The variants differ in their visual temporal input: the current frame alone, the initial and current frames, or the initial frame and a sliding window of temporally ordered recent frames. This comparison does not include the RLVR stage. Online policy learning uses three training seeds and identical RL budgets and hyperparameters, with 100 evaluation rollouts per task.

Table 10: Success rates (%) using independently trained SFT-only reward models with matched data and CoT supervision. Means are rounded to the nearest integer. The full-context row is SFT-only, distinct from the full SFT+RLVR SOLE-R1 model in other comparisons.

Visual temporal input	RoboSuite pick cube	Meta-World open window	ManiSkill push cube	Mean
Current frame only	23	8	29	20
Initial and current frames	57	44	32	44
Initial frame and sliding window	**83**	**71**	**58**	**71**

The improvement with additional temporal context cannot be attributed solely to synthesized data or CoT supervision, which are shared across the variants. The sliding window also outperforms the two-frame input, supporting the utility of intermediate observations. This experiment isolates the visual temporal-input configuration conditional on shared data and CoT supervision; it does not separately identify the effect of frame ordering versus the additional visual evidence.

P.2 Controlling reward-model camera access

The baseline prompting comparison in Appendix H.2.4 evaluates external-only, wrist-only, composite-view, and longer-context inputs. External-only input gives GPT-5 and Gemini their highest task success in that comparison; composite views can be misinterpreted as multiple robots. We additionally evaluate SOLE-R1 with only the external view on the three tasks in Table 11.

Removing the wrist view reduces SOLE-R1 performance, but external-only SOLE-R1 still exceeds both baselines on each tested task. Its advantage is therefore not explained solely by access to the wrist camera.

Table 11: Policy success rates (%) controlling reward-model camera access. Means are rounded to the nearest integer. This comparison matches camera access; models retain their respective temporal prompts.

Reward model and camera input	RoboSuite pick cube	Meta-World open window	ManiSkill push cube	Mean
SOLE-R1: external and wrist	94	72	63	76
SOLE-R1: external only	83	68	55	69
GPT-5: external only	4	26	21	17
Gemini-3-Pro: external only	2	28	18	16

P.3 Simulation annotation with reduced privileged supervision

We compare the standard simulation annotations against two variants: (i) a foundation VLM generates progress targets and CoT annotations from task language and video without privileged-state grounding; and (ii) a foundation VLM generates CoT annotations conditioned on distance-based progress estimates, without templated language. The comparison holds the underlying trajectories, model initialization, training duration, remaining data mixture, and online RL configuration fixed, and apply SFT followed by RLVR to each variant. The second variant retains privileged progress supervision and isolates the additional contribution of templated language. The first tests removing privileged annotation supervision. Neither variant by itself replaces simulation-based trajectory generation. The reported success rates below are averaged across three seeds.

Table 12

Simulation annotation	RoboSuite pick cube	Meta-World open window	ManiSkill push cube	Mean
Distance targets and templates (original)	**94**	**72**	**63**	**76**
VLM only, no privileged grounding	68	39	50	52
VLM with distance targets, no templates	81	53	56	63

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe the contributions of our work along with the scope of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We describe the limitations of our work in the section on limitations above.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details for all models, hyperparameters, and experiments in the experiments section or appendix in sufficient detail to reproduce all results (and the code is provided at the anonymous page: `https://sole-r1.github.io/`)

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All code and data (along with demos and steps for reproducing all experiments and datasets) are provided at the anonymous page: `https://sole-r1.github.io/`

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: For training and testing all models, we outline the details needed to reproduce training and RL experiments in the appendix, along with links to all the code/steps needed to reproduce experiments at `https://sole-r1.github.io/`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We plot error bars (and explain their meaning) wherever it is appropriate in the figures of the Experiments section and Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe compute resources and outline all details needed to reproduce experiments in the Appendix, along with links to all the code/steps needed to reproduce experiments at `https://sole-r1.github.io/`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research fully conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We describe the broader impact in the dedicated section above.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We are not aware of any such risks associated with this paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all papers/authors corresponding to any data or code that we used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We outline all steps, along with links to the data and the code, to download the training data, reproduce the training run, and reproduce the RL experiments at `https://sole-r1.github.io/`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: There was no crowdsourcing or human subjects involved in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: There were no study participants involved in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were only used in editing for grammar and spell checking of the manuscript draft.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.