2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Do What You Say: Steering Vision-Language-Action Models via Runtime Reasoning-Action Alignment Verification

Yilin Wu[2*], Anqi Li[1], Tucker Hermans[1,3], Fabio Ramos[1,4], Andrea Bajcsy[2§], Claudia Pérez-D'Arpino[1§]

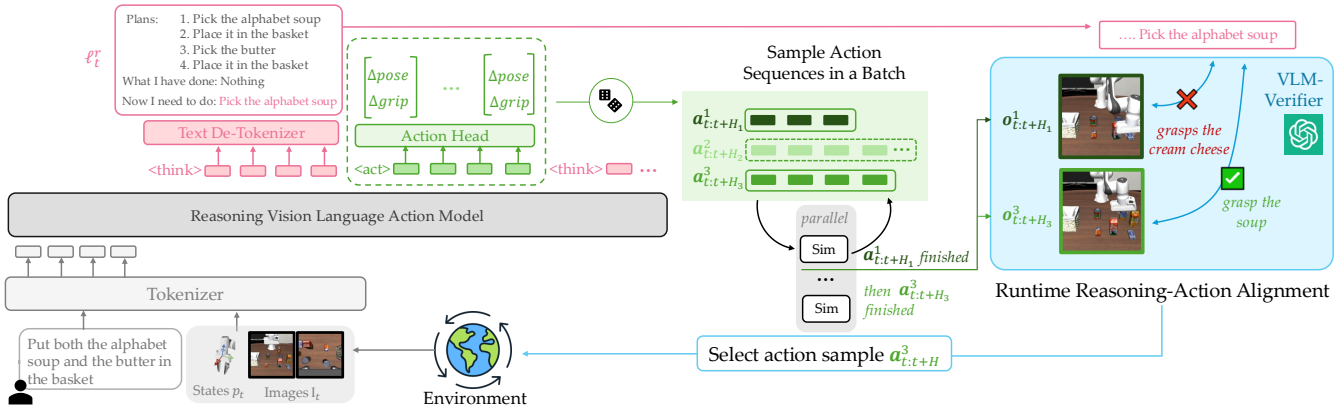


Fig. 1: **Method Overview.** Reasoning Vision Language Action (VLA) models interleave textual planning and action generation. After generating a text plan which describes intermediate goals, we sample a batch of action sequences, forward simulate their outcomes until the model switches to think again. We then use a Vision Language model (VLM) verifier to score alignment between the action's outcomes and the text plan. This improves the *embodied CoT faithfulness* at runtime by executing only action samples that achieve the outcome of the text plan.

***Abstract*— Reasoning Vision Language Action (VLA) models improve robotic instruction-following by generating step-by-step textual plans before low-level actions, an approach inspired by Chain-of-Thought (CoT) reasoning in language models. Yet even with a correct textual plan, the generated actions can still miss the intended outcomes in the plan, especially in out-of-distribution (OOD) scenarios. We formalize this phenomenon as a lack of *embodied CoT faithfulness*, and introduce a training-free, runtime policy steering method for reasoning-action alignment. Given a reasoning VLA's intermediate textual plan, our framework samples multiple candidate action sequences from the same model, predicts their outcomes via simulation, and uses a pre-trained Vision-Language Model (VLM) to select the sequence whose outcome best aligns with the VLA's own textual plan. Only executing action sequences that align with the textual reasoning turns our base VLA's natural action diversity from a source of error into a strength, boosting robustness to semantic and visual OOD perturbations and enabling novel behavior composition without costly re-training. We also contribute a reasoning-annotated extension of LIBERO-100, environment variations tailored for OOD evaluation, and demonstrate up to 15% performance gain over prior work on behavior composition tasks. The overall framework scales with compute (347ms at $K = 10$ samples) and data diversity. Project Website at: https://yilin-wu98.github.io/steering-reasoning-vla/**

I. Introduction

Vision Language Action (VLA) models [1][2], which combine pre-trained vision-language backbones with action-generation heads fine-tuned on robot data [3], promise significant advances in generalizable robotic control. However, in practice, these models are not yet true generalists: their performance often degrades in novel scenes and over long horizons, and their ability to follow complex language instructions can be brittle [4][5][6].

To tackle these challenges and better exploit the vision-language backbone of the VLA, a growing body of work has introduced "reasoning VLAs" [7][8]. This approach is inspired by the success of Chain-of-Thought (CoT) in large language models (LLMs): by producing intermediate step-by-step textual reasoning before generating a textual response, LLMs can significantly increase the complexity and reliability of language generation [9]. Similarly, reasoning VLA's implement a form of "embodied CoT" [7]: instead of generating actions directly, the model first generates intermediate textual reasoning, often grounded by visual observations, before producing reasoning-conditioned actions. For example, given the task description of "Put both the alphabet soup and the butter in the basket" in Fig. 1, the model generates the textual reasoning "First I pick up the soup, then place it in the basket..., Now I need to pick up the soup" and then generates a low-level action sequence (e.g., a sequence of relative end-effector poses).

However, even with the perfect textual reasoning, the VLA-sampled action sequence may lead to different outcomes than the generated plan (e.g., grasping the cream cheese instead of the soup, as shown in Fig. 1). This misalignment—where the outcomes of the generated actions fail to match the textual plan—is particularly prevalent in out-of-distribution (OOD) scenarios (novel instructions, objects, backgrounds, etc.), where enforcing alignment shows

[1] NVIDIA. [2] Carnegie Mellon University. [3] University of Utah. [4] University of Sydney. [§] Equal advising. * Work done during an internship at NVIDIA.

up to a 20% performance gain in Sec. V-C. This also mirrors observations from prior work on training reasoning VLAs [10], where the reasoning capabilities of the models are stronger than language-conditioned action generation capabilities. Analogous to Chain-of-Thought (CoT) faithfulness in LLMs, which questions whether a model's generated reasoning accurately reflects the process used to derive an output [11], we formalize this reasoning-action misalignment as a robotics version of lack of *embodied CoT faithfulness*: the expectation that outcomes described by a VLA's textual reasoning will be reliably realized by its subsequent low-level actions.

To address this gap, we introduce a **runtime reasoning-action alignment** method. This approach enforces embodied CoT faithfulness by actively steering a reasoning VLA's actions to match its own textual reasoning during execution. Specifically, right before the reasoning is updated, we autoregressively generate multiple action sequences from the base model, conditioned on the current textual plan and the observation from the preceding step (center, Fig. 1). We predict the outcomes of action samples via parallel simulation, and leverage an off-the-shelf Vision Language Model (VLM) verifier to score the alignment between the outcome and the preceding text plan. Only the action sequence that induces the correct outcome is executed on the robot.

Our contributions are summarized as follows:

- **Runtime Policy Steering for Reasoning-Action Alignment:** we propose a new runtime framework for ensuring that a VLA's generated action samples induce the outcomes described by its own textual reasoning with an average inference time that scales favorably (347ms at $K = 10$ samples). Even in in-distribution (ID) scenarios, we find that reasoning-action alignment improves task success by 8% and preserves long-horizon semantic coherence without needing any additional fine-tuning data. More benefits are shown in OOD scenarios below.
- **Extensive Generalization Experiments and Scaling Analysis:** Through a controlled series of experiments on OOD shifts and behavior composition tasks in our extended version of LIBERO benchmark [12], we find that our method outperforms relevant baselines by up to 15%. This performance gap widens as the fine-tuning dataset grows, since a stronger policy generates a better set of candidate actions for our verifier to select from.
- **A Reasoning-Annotated VLA Dataset and an Extended Benchmark for Generalization Tests:** We contribute an open-sourced reasoning-annotated LIBERO-100 dataset for reasoning VLA training. We also contribute an extension of the LIBERO benchmark with new task descriptions, visual variations, and new behavior compositions. We use these for our evaluations but also plan to release it for the community to continue to study generalization capabilities of reasoning VLAs.

II. Related Work

Vision Language Action Models & Reasoning. Large robotics datasets [3] have enabled the creation of generalist Vision Language Action (VLA) models [1][2]. Despite their success, these VLA models often struggle with long-horizon tasks and out-of-distribution (OOD) generalization due to compounding errors and a limited ability to follow complex or abstract language instructions. To improve robustness, many approaches augment VLAs with Chain-of-Thought reasoning, either through hierarchical planner-controller dual systems [5] or an unified model that interleave intermediate reasoning and action generation [7][8][13][14]. We build on the unified approach but identify a critical failure mode: a reasoning-action mismatch, where motor actions do not faithfully execute the model's self-generated textual plan.

Runtime Optimization and Steering of Large Models. Runtime optimization improves model's performance without costly retraining. In robotics, runtime steering has improved the alignment between human intent and robot actions in diffusion policies [15], and in VLA policies by sampling multiple action chunks and verifying them against learned critic. These critics are typically either offline Q-functions [16] (which can struggle in novel scenarios), or specially fine-tuned VLMs from synthetic preference labels which may not correlate with true task success [6]. Critically, these methods verify actions in isolation and do not address the reasoning-action faithfulness gap—the challenge of ensuring that a sequence of low-level actions are semantically coherent with a model's self-generated textual plan. Inspired by faithfulness verification research in LLMs [11], our work explores a new direction: using runtime steering to explicitly enforce the alignment between a VLA's high-level reasoning and its low-level motor control.

OOD Robustness & Generalization. A central motivation for improving reasoning-action alignment is to enhance out-of-distribution (OOD) robustness and generalization. To systematically evaluate such scenarios, the community has developed challenging benchmarks and OOD-related taxonomies [4][12]. However, existing OOD studies often focus on robustness to visual or semantic shifts (e.g., novel instructions, objects and backgrounds), with less attention paid to compositional generalization: the ability to recombine learned skills to solve new tasks. We adopt the LIBERO benchmark [12] for its focus on long-horizon, compositional tasks. However, as it lacks the clear separation of different OOD factors, we extend it to systematically study OOD robustness and generalization. Furthermore, while recent work has studied OOD failure detection [17] and mitigation via observation interventions [18], our method focuses on improving compositional generalization via runtime verification, enabling the policy to solve unseen tasks by executing actions that achieve its own plan.

III. Problem Formulation

We contextualize our problem setting by first contrasting "vanilla" VLAs with reasoning VLAs. We then formally define the *embodied CoT faithfulness gap* between a reasoning VLA's plans and the outcome of its actions that we address in Sec. IV.

Setup and Notation. We consider general robotic manipulation settings. Let ℓ^g denote the high-level natural language instruction for a task (e.g., clean the table). At each real timestep t, the robot's observation is $o_t = (I_t, p_t)$, where I_t is the visual observation including, e.g., wrist and agent-view RGB images, and $p_t \in \mathbb{R}^d$ is the proprioceptive state of the robot. The robot's policy maps the current observation o_t and language instruction ℓ^g to action a_t.

Vanilla VLA Formulation. Traditional vision-language-action (VLA) models [2] learn policies end-to-end via a demonstration dataset of N trajectories, $\mathcal{D} = \{(\tau, \ell^g)_i\}_{i=1}^N$, where each trajectory τ consists of T observation-action pairs, $\tau = \{(o_t, a_t)\}_{t=1}^T$. The VLA policy π_θ^{vla} maps the observation and instruction (o_t, ℓ^g) to a distribution over actions a_t. The training objective is to maximize the log-likelihood of the policy under the expert data distribution:

$$\mathcal{L}_{\text{vla}}(\theta; \mathcal{D}) = \sum_{(o_t, a_t, \ell^g) \in \mathcal{D}} -\log \pi_\theta^{\text{vla}}(a_t \mid o_t, \ell^g). \quad (1)$$

During inference, at any timestep t, the policy samples an action to execute given the current observation and language instruction: $\hat{a}_t \sim \pi_\theta^{\text{vla}}(\cdot \mid o_t, \ell^g)$. However, vanilla VLA models often degrade on long-horizon tasks [5], as errors compound over extended trajectories and complex language instructions are difficult to decompose and ground in actions.

Reasoning VLA Formulation. To mitigate the challenges of long-horizon planning, reasoning VLAs [7][8][13] factor the problem by adding the generation of Chain-of-Thought (CoT) in the form of intermediate reasoning. To learn this "embodied" version of CoT, reasoning VLAs are trained on a reasoning-annotated demonstration dataset with fine-grained textual plans. Let this dataset be denoted by $\mathcal{D}_{\text{reason}} = \{(\tau^{\text{reason}}, \ell^g)_i\}_{i=1}^N$, where each trajectory τ_i^{reason} is divided into L_i segments: $\tau_i^{\text{reason}} = \{(\mathbf{o}_j, \mathbf{a}_j, \ell_j^r)\}_{j=1}^{L_i}$.Here, ℓ_j^r is the text plan for the j-th segment, with corresponding observation and action sequences $\mathbf{o}_j = o_{t_j:t_j+H_j}$ and $\mathbf{a}_j = a_{t_j:t_j+H_j}$ of varied lengths. For notational simplicity, we drop the subscript j from the start time t and use $\mathbf{o}_j = o_{t:t+H}$ and $\mathbf{a}_j = a_{t:t+H}$ hereafter.

Given the reasoning dataset $\mathcal{D}_{\text{reason}}$, a reasoning VLA $\pi_\theta^{\text{r-VLA}}$ is trained via supervised fine-tuning (SFT) to generate both (i) a textual plan as intermediate reasoning and (ii) the subsequent plan-conditioned action sequence [8][13]. Specifically, $\pi_\theta^{\text{r-VLA}}$ takes as input the tokenized observations, language instruction and intermediate text plans, and generates tokens that are decoded either as text or actions. The training objective is a weighted sum of the text loss and action loss:

$$\begin{aligned}
\mathcal{L}_{\text{r-vla}}(\theta; \mathcal{D}_{\text{reason}}) &= \sum_{(o_{t:t+H}, a_{t:t+H}, \ell_j^r, \ell_{j-1}^r, \ell^g) \in \mathcal{D}_{\text{reason}}} \lambda_{\text{reason}} \mathcal{L}_{\text{reason}} + \lambda_{\text{act}} \mathcal{L}_{\text{act}} \\
\mathcal{L}_{\text{reason}} &= -\log \pi_\theta^{\text{r-vla}}(\ell_j^r \mid o_t, \ell_{j-1}^r, \ell^g) \\
\mathcal{L}_{\text{act}} &= -\sum_{t'=t}^{t+H} \log \pi_\theta^{\text{r-vla}}(a_{t'} \mid o_{t'}, \ell_j^r, \ell^g)
\end{aligned} \quad (2)$$

where $0 < \lambda_{\text{reason}} < 1$ and $0 < \lambda_{\text{act}} < 1$ and $\lambda_{\text{reason}} + \lambda_{\text{act}} = 1$.

During inference, at real timestep t, given the text plan $\hat{\ell}_{\text{last}}^r$ from last reasoning step, the policy can either update the current textual plan (e.g, "pick up the soup can") via $\hat{\ell}^r \sim \pi_\theta^{\text{r-vla}}(\cdot \mid o_t, \hat{\ell}_{\text{last}}^r, \ell^g)$ or generate the action conditioned on the last textual reasoning: $\hat{a}_t \sim \pi_\theta^{\text{r-vla}}(\cdot \mid o_t, \hat{\ell}_{\text{last}}^r, \ell^g)$. This decomposition aims to leverage the strong reasoning and language understanding capabilities of the pre-trained VLM backbone. In Sec. IV-A, we discuss the specific reasoning VLA [8] we build our method on.

Embodied Chain-of-Thought (CoT) Faithfulness Gap. In factorized formulations of reasoning VLAs, textual reasoning and action generation are optimized together. Prior work [10] has shown that while the pre-trained VLM backbone of a VLA can be efficiently finetuned for textual planning with sparse data, the same is not necessarily true for low-level control. Our experiments confirm that acquiring robust control policies conditioned on the text plan is a far more challenging problem: for example, the text generation is largely reliable (100% accuracy for in-distribution tasks trained with LIBERO-10-R dataset in Sec. V-B), while most task failures stem from the action generation module failing to execute that plan.

Given that the textual plan is often correct or can be easily corrected by an external agent [7], the critical bottleneck is the policy's ability to reliably follow its own reasoning. We define this core challenge as the *Embodied Chain-of-Thought Faithfulness Gap*: the misalignment between a generated textual plan and the physical outcome of the associated low-level actions.

Goal. In this work, our goal is to minimize this gap at each timestep t, which can be formalized as minimizing the misalignment loss for candidate action sequence $\hat{a}_{t:t+H}$ sampled from $\pi_\theta^{\text{r-vla}}$:

$$\mathcal{L}_{\text{align}}(\theta; o_t, \hat{\ell}^r, \ell^g) = -\mathbb{E}_{\mathcal{P}, \pi_\theta^{\text{r-vla}}} \left[R_{\text{align}}(o_{t:t+H}, \hat{\ell}^r) \right], \quad (3)$$

where the expectation is taken over the stochastic outcomes of the environment's true dynamics $\mathcal{P}$ and the reasoning VLA model, i.e., $o_{t'+1} \sim \mathcal{P}(\cdot | o_{t'}, \hat{a}_{t'})$, $\hat{a}_{t'} \sim \pi_\theta^{\text{r-vla}}(\cdot | o_{t'}, \hat{\ell}^r, \ell^g)$, for all $t' \in [t, t+H]$ and Here, $R_{\text{align}}(o_{t:t+H}, \hat{\ell}^r)$ is a reward function that measures whether the future observation $o_{t:t+H}$ satisfies the subgoal described by the textual plan $\hat{\ell}^r$.

Crucially, this alignment objective is distinct from the standard behavior cloning loss (e.g., Eq. 2) used to train the reasoning VLA. Since the model is not explicitly optimized to maximize this alignment, the actions it generates can be imprecise at best or pursue an entirely different subgoal than the one articulated in the textual plan at worst.

IV. Approach: Steering for Embodied Reasoning-Action Alignment (SEAL)

Our key idea is to enforce *Embodied CoT Faithfulness* at runtime, thereby ensuring that actions produced by a reasoning VLA model semantically realize the model's own intermediate textual plans. To achieve this, we first train a

reasoning VLA model from prior open-source work [8] and then propose a runtime policy steering procedure to select the actions that maximize reasoning-action alignment. We call our overall steering approach **SEAL**: **S**teering for **E**mbodied reasoning-action **AL**ignment.

A. Training a Reasoning VLA Model

Throughout our experiments, we train reasoning VLAs in a controlled way to study the capabilities conditioned on a specific fine-tuning dataset. To enable this controlled fine-tuning, we first propose a pipeline to automatically annotate any given demonstration dataset with intermediate textual reasoning. Using this annotated dataset, we then fine-tune the π_0 base model [2], following the training recipe from [8], resulting in a VLA capable of interleaving text reasoning with action generation.

Reasoning Data Annotation Pipeline. Most robot demonstration datasets contain only a single high-level language text instruction ℓ^g per episode. To obtain intermediate reasoning labels, we introduce an automated annotation pipeline that decomposes long-horizon demonstrations into intermediate text plans. As shown in Fig. 1, we modify the reasoning format from prior work [8] and define an intermediate reasoning step ℓ^r to abide by the following format:

- *Plans*: <all text plans in the task>
- *What has been done*: <completed plans>
- *Now I need to do*: <the next text plan to execute>

To avoid the expensive manual annotation of sub-task boundaries required by prior work [8], we use Gemini [19] to automatically generate annotations given the input of demonstration videos and their high-level task instructions (ℓ^g). In a single pass, Gemini jointly generates a one-sentence textual plan ($\hat{\ell}^r_j$) and its corresponding end-timestep t'_j for each sub-task. Since one sub-task's end marks the next one's start, this process directly yields a sequence of annotations: $(\hat{\ell}^r_1, 1), (\hat{\ell}^r_2, t'_1), \ldots, (\hat{\ell}^r_L, t'_{L-1})$. The number of sub-tasks L varies per episode based on Gemini's decomposition, and we manually verify all generated annotations for quality (details in Appendix A.1 on our website).

Reasoning VLA Training. While our proposed runtime steering method is model-agnostic, we instantiate our system using a reasoning VLA based on recent work [8]. This architecture features an adaptive switching capability between text and action generation. The core of this mechanism lies in two special tokens, `<think>` and `<act>`, which function as switching signals. When the model generates `<think>`, subsequent tokens are decoded as textual reasoning until an end-of-sentence token is reached. When the model generates `<act>`, the following tokens are passed to an action head, which produces a continuous action sequence through a diffusion process in Fig. 1. This adaptive switching allows the VLA model to generate variable-length action sequences tailored to the complexity of the its own textual plan, effectively segmenting tasks into a series of text-plan-conditioned action sequences.

For training, we follow the recipe from [8], fine-tuning the pre-trained π_0 VLA on our reasoning-annotated dataset. The training data is formatted by prepending the `<think>` token to each textual reasoning label ℓ^r and prepending the `<act>` token to its corresponding action sequence. The model is then fine-tuned end-to-end with the combined loss function in Eq. 2, which uses a cross-entropy loss for text generation and a flow-matching loss for action generation.

B. Runtime Policy Steering via Semantic Verification

As formalized in Sec. III, ensuring that the action sequence will cause outcomes that match with the text plan requires solving the alignment objective in Eq. 3. This presents two fundamental challenges.

- **Direct Optimization**: The loss $\mathcal{L}_{\text{align}}$ is hard to optimize directly via backpropagation due to the non-differentiable dynamics $\mathcal{P}$ and action sampling from $\pi_\theta^{\text{r-vla}}$.
- **Reward Modeling**: The true alignment reward R_{align} is hard to model analytically or via heuristics, as it requires complex and contextual semantic judgment.

To overcome these obstacles, we introduce a novel runtime policy steering method that provides a tractable, approximation of the reasoning-action alignment problem. Our approach circumvents the direct optimization challenge using a best-of-N sampling strategy, an effective technique in LLMs [20]. We address the challenge of reward modeling by using the strong commonsense and physical reasoning capabilities of pre-trained VLMs as our open-world verifier [17][21][22]. Our framework is a three-stage process, which we detail next.

Runtime Policy Steering: Hypothesize, Predict, and Verify. Our runtime steering approach unfolds in a three-stage process at each reasoning step, after a new textual plan $\hat{\ell}^r_t$ is generated.

1. Hypothesize. In this stage, the policy generates multiple plausible sequences of actions. We sample a set of K candidate action sequences in parallel from the policy $\pi_\theta^{\text{r-vla}}$, conditioned on $(o_t, \hat{\ell}^r, \ell^g)$. The generation of each sequence is autoregressive, meaning that each action is sampled one step at a time. The process terminates when the model outputs a `<think>` token, resulting in variable-length action sequences (shown in center, Fig. 1). This process yields a set of hypotheses:

$$A_t = \{\hat{\mathbf{a}}_t^{(k)} \sim \pi_\theta^{\text{r-vla}}(\cdot \mid o_t, \hat{\ell}^r, \ell^g)\}_{k=1}^K \tag{4}$$

where each $\hat{\mathbf{a}}_t^{(k)} = a^{(k)}_{t:t+H_k}$ has a different horizon H_k[1].

2. Predict. The *Predict* and *Hypothesize* stages are tightly coupled in an interleaved loop. This is necessary because the policy is autoregressive: generating the action $\hat{a}_{t'}$ requires the predicted observation $\hat{o}_{t'}$. We use a dynamics model, $\hat{\mathcal{P}}$, that approximates the ground-truth dynamics $\mathcal{P}$ to generate the

[1]The sequence $\hat{\mathbf{a}}_t^{(k)}$ is generated autoregressively from the model, rather than in a single forward pass.

observation sequence $\hat{\mathbf{o}}_t^{(k)}$ given $\hat{\mathbf{a}}_t^{(k)}$. In simulation, we use parallel environment instances for high-fidelity predictions [12]; in the real world, a learned world model [23] or digital twin [24] would serve this role. This *Hypothesize-Predict* loop continues until the next think token is generated.

3. Verify. In the final stage, we select the most aligned action sequence by leveraging a pre-trained VLM (e.g., GPT-4o) as a proxy R_ψ for the ground-truth alignment reward, R_{align}. The VLM verifier assigns a score to each candidate sequence by evaluating its predicted outcome against the textual plan.

The action sequence that receives the highest alignment score from the VLM is chosen for execution. The VLM is prompted to return a binary score $R_\psi \in \{0, 1\}$ indicating whether the predicted outcome successfully fulfills the plan. For a practical speedup, we only input the initial image of the episode I_1, the predicted final image from the candidate sequence $\hat{I}_{t+H_k}^{(k)}$, and the textual plan ($\hat{\ell}^r$) to the VLM. More details are in Appendix A.4 on our website.

Runtime Considerations. Our framework is optimized for low latency using parallel and asynchronous execution. The K action sequences are generated in parallel via a tightly coupled *Hypothesize-Predict* loop. Crucially, the *Verify* stage runs asynchronously, immediately evaluating each candidate as it completes generation. Since the sequences have variable lengths and finish at different times, this enables an early-exit strategy: we execute the **first** sequence that the VLM successfully verifies, rather than waiting to evaluate all K candidates. This significantly reduces decision-making latency (see Appendix A.5 on our website).

V. Experiments

We conduct a series of experiments in a simulation benchmark V-A to carefully study: (1) the benefit of enforcing reasoning-action alignment at runtime (Sec. V-B), (2) robustness to out-of-distribution (OOD) shifts and generalization to new behavior compositions (Sec. V-C), and (3) performance and runtime scaling as a function of the number of samples to be verified (Sec. V-D). Videos can be found on our website.

A. Benchmark & Implementation Details

Training Dataset. We use the LIBERO benchmark [12] due to its diverse, long-horizon manipulation tasks. To investigate how performance scales with the size and diversity of the VLA fine-tuning data, we use three datasets of increasing scale. Each dataset is processed with the reasoning annotation pipeline described in Section IV-A. ***Libero-10-R***: Our smallest reasoning dataset, consisting of demonstration data for the *LIBERO-10* evaluation tasks. ***Libero-100-Basket-R***: A medium-sized dataset we create to test skill composition, supplementing *Libero-10* with a curated subset of tasks from *Libero-90* that involve "pick and place into basket" skills. ***Libero-100-R***: Our largest dataset, generated from annotating all demonstrations from the full *LIBERO-100* suite.

Evaluation. We evaluated our method in three suites of tasks, each of which involved running 50 trials. ***(1) In-Distribution (ID) Performance*** (Sec. V-B) uses the test set from the *LIBERO-10* benchmark which has 10 long-horizon manipulation tasks. ***(2) Robustness to OOD Shifts*** (Sec. V-C) varies the visual and language instruction based on the taxonomy proposed by [4]. We construct four OOD variants of the *LIBERO-10* suite, each of which contains 10 tasks from the base suite. Among them, two variants are modified with *semantic OOD* changes where instructions are rephrased but the object descriptions are the same (**LIBERO-10-Lang-Rephrase**), or object descriptions are changed using alternate properties (**LIBERO-10-Lang-Object-Property**). We also have *visual OOD* changes, where objects are added or non-target objects are replaced (**LIBERO-10-Visual-Scene**), or the background and camera viewpoints are changed (**LIBERO-10-Visual-Viewpoint**).

Finally, ***(3) Generalization to Behavior Composition*** (Sec. V-B) is tested in a suite of new long-horizon tasks that recombine learned skills, while sharing the same initial states as the original training demonstrations. For example, a model trained on tasks like "put A and B in the basket" and "put C in the basket" is then evaluated on the unseen instruction "put A and C in the basket." **LIBERO-10-Compose** has two new compositional basket-related pick-and-place tasks, **LIBERO-100-Basket-Compose** has nine novel compositional basket-related tasks, and **LIBERO-100-Compose** has thirteen novel compositional tasks across a wider range of objects and tasks. The full list of these tasks and their videos are on our website.

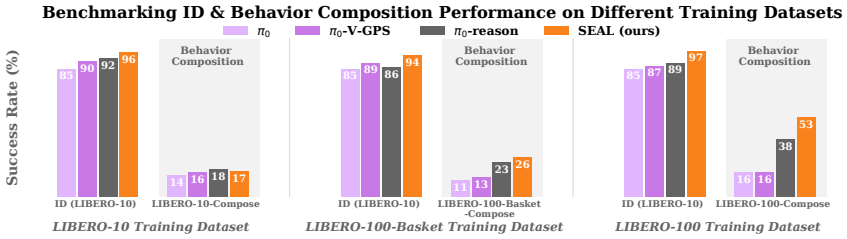
Baselines. We compare our approach, **SEAL**, to three baselines all based on the same π_0 [2] architecture. π_0 is a vanilla VLA [2] that directly maps visual-language inputs to actions without any intermediate textual reasoning. **π_0-reason** is the base reasoning VLA model [8] trained with our reasoning-annotated data without runtime verification and steering. It is a direct ablation of our primary contribution. **π_0-V-GPS** [16] is a state-of-the-art runtime steering method[2]. It trains a critic Q-function using offline reinforcement learning [25] and, at runtime, executes the action chunk with the highest Q-value from a set of policy samples. In contrast, **SEAL** verifies an entire action sequence against a textual plan using a pre-trained VLM as the critic. For **π_0-V-GPS** and **SEAL**, we use $K = 10$ candidate samples per reasoning step unless otherwise specified. We choose the temperature of 1 for **π_0-V-GPS** as it has the best average performance with a detailed discussion of **π_0-V-GPS** in the Appendix A.3.

Training. Our baseline reasoning VLA model, **π_0-reason**, is created by fully fine-tuning the π_0 [2] backbone on our reasoning-annotated dataset with $\lambda_{\text{reason}} = 0.5, \lambda_{\text{act}} = 0.5$ as loss coefficients. Training **π_0-reason** takes approximately 20 hours on 8 A100 GPUs, whereas finetuning π_0 takes about 6 hours. More details are in Appendix. A.2

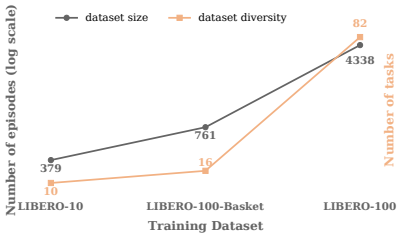
B. On The Value of Verifying Reasoning-Action Alignment

We first study the value of verifying reasoning-action alignment by comparing **SEAL** against the three baselines

[2]Another recent runtime steering method is [6], but its pre-trained action verifier for LIBERO has not been released, precluding a fair comparison.



(a) **In-distribution & Compositional Generalization Success Rates.** Each model is evaluated by averaging the performance of all tasks in its corresponding suite in Sec V-A, with 50 trials per task.



(b) **Dataset Statistics.** Comparison of diversity (number of tasks) and sizes (number of episodes) of our three datasets.

Fig. 2: **Dataset Scaling Results.** Our method, **SEAL**, consistently outperforms all baselines by enforcing reasoning-action alignment with increased performance as the training data scales, especially on challenging behavior composition tasks.

from above on both in-distribution tasks and the challenge of novel behavior composition tasks.

Results: In-Distribution (ID) Tasks. On ID tasks from *LIBERO-10*, Fig. 2a (left panel for each dataset), our method **SEAL** achieves the highest success rates of 94-97%, and both the base reasoning VLA π_0**-reason** and π_0**-V-GPS** outperform the no-reasoning π_0 policy.

Since these task instructions match the training distribution of π_0**-V-GPS**, this method can provide reliable guidance, with the steady increase in Q-values shown in Fig. 3 (left). The place where **SEAL** shines is addressing the inherent limitations of chunk-based steering. Since the Q-function evaluates short fixed-length action chunks, it can oscillate between choices between timesteps. In contrast, our method verifies at the *plan-level*—where the variable-length action sequences correspond to the model's textual plan—filtering out executions that are imprecise or correspond to the wrong subgoal, thus achieving a higher success rate.

Results: Novel Behavior Composition. To test the ability to reuse learned low-level behaviors for novel instructions, we evaluate on behavior composition tasks from Sec. V-A.

In Fig. 2a (right panel for each dataset), we see that as the VLA training dataset grows, the reasoning-enabled models (π_0**-reason** base and **SEAL** verified) show substantial performance gains while π_0 and π_0**-V-GPS** stagnate at a low success rate (∼15%). We hypothesize that this is because the weak language understanding of π_0**-V-GPS** (MUSE encoder [26]) fails to generalize to novel instructions and gives misleading value signals, shown in Fig. 3 (right).

In contrast, the reasoning in π_0**-reason** exhibits stronger generalization. We hypothesize this is due to the strong VLM backbone that can generate correct plans that decompose novel long-horizon problems into familiar subtasks. However, its performance is ultimately capped by a new challenge: as the training data and skill diversity grow, the policy's action generation becomes more varied, increasing the likelihood of misalignment with its own correct plan.

Our method, **SEAL**, is designed specifically to resolve this tension. By verifying and selecting the action sequence that faithfully executes the generated plan, it leverages this action diversity as a strength rather than a source of errors. Overall, **SEAL** exhibits a positive scaling trend, especially on large and diverse training datasets like *LIBERO-100-R*.

C. On the Value of Verification for Robustness to OOD shifts

We further evaluate the OOD robustness of each method on four challenging distribution shifts, from Sec. V-A: two semantic shifts (*Lang-Rephrase*, *Lang-Object-Property*) and two visual shifts (*Visual-Scene*, *Visual-Viewpoint*).

Results: Semantic OOD Robustness. We find that reasoning-based models are most robust (Fig. 4), while non-reasoning baselines like π_0 and π_0**-V-GPS** prove brittle. For instance, the base π_0 policy's performance drops by 12% on rephrased instructions alone, while π_0**-V-GPS**'s success rate falls as low as 71%. In contrast, generating intermediate textual plans makes π_0**-reason** highly robust to paraphrasing. However, we found that spurious word-object correlations can still lead to action errors for π_0**-reason**. Our method, **SEAL**, inherits this linguistic robustness but further improves upon it by using the verification stage to filter out the resulting action errors, consistently maintaining a success rate above 91%.

Results: Visual OOD Robustness. We find that replacing distractor objects (*Visual-Scene*) minimally influences most methods, but causes a notable performance drop (4%) for π_0**-V-GPS**. We hypothesize that its Q-function has overfit to the specific visual context of the training demonstrations.

In contrast, changing the viewpoint and background (*Visual-Viewpoint*) is the most challenging test, causing performance to drop significantly across all methods. We hypothesize that this is influenced by the lack of viewpoint and background diversity in the training data for the same task. However, here we see the most pronounced benefit of our method: by filtering out noisy actions caused by the severe visual shift, **SEAL** maintains a 45% success rate, outperforming the next-best baseline by over 17%.

Summary. Across four stress-tests, **SEAL** is consistently the most robust method, maintaining a strong quantitative and qualitative performance improvement (shown in Fig. 5). Our results also suggest that current VLA models are far more vulnerable to major visual shifts (like viewpoint and background changes) than to semantic or minor scene changes.

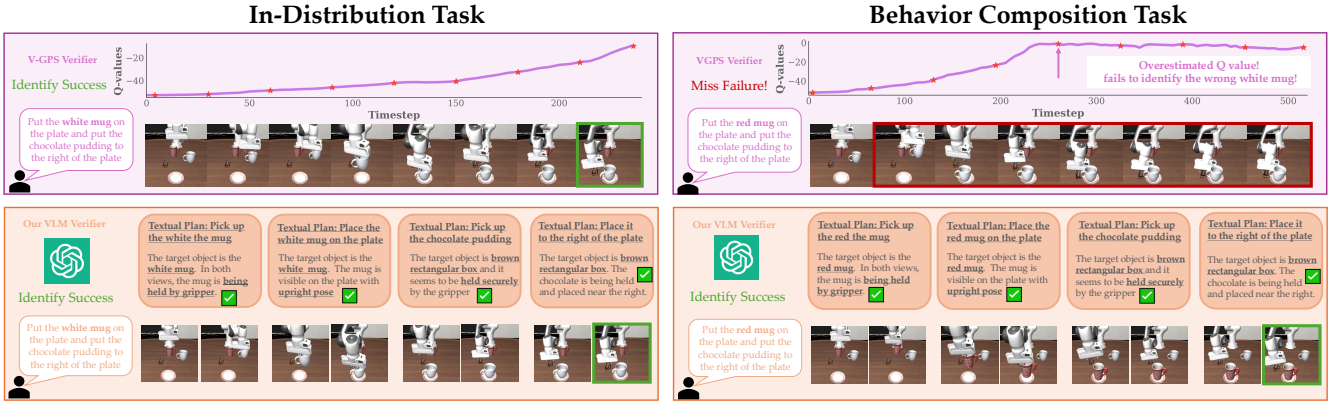


Fig. 3: **Comparison Between Verification Methods.** (left) For ID tasks, both the Q-value based verifier in π_0-V-GPS and our VLM verifier can accurately predict task progress. (right) For behavior composition tasks requiring stronger generalization capabilities, the Q-value overestimates task progress, failing to provide useful signal to guide π_0. However, our VLM verifier preserves the commonsense reasoning and correctly guides the base π_0-reason policy towards the task goal.

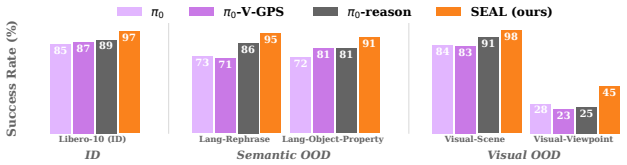


Fig. 4: **Robustness to Semantic and Visual OOD Shifts.** We evaluate four different OOD variations for each task in *Libero-10*. The performance is averaged across 10 tasks in the task suite with 50 trials per task. SEAL maintains the best performance across all OOD shifts, even in the most challenging Visual-Viewpoint OOD scenario.

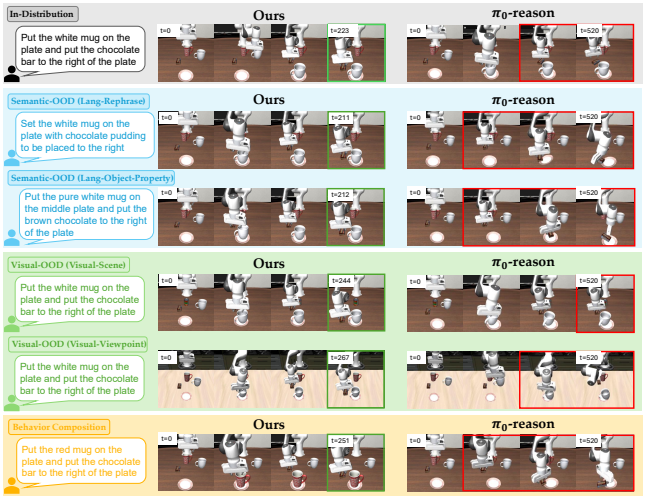


Fig. 5: **OOD Shifts & Behavior Composition Examples.** We visualize SEAL and π_0-reason. We mark a successful rollout with a green box and a failure with red. SEAL (on the left) can reliably generate successful rollouts to complete the tasks. π_0-reason policy (on the right) fails due to imprecision or by completing a totally different textual plan.

D. Runtime Scaling Law

We find that increasing the number of candidate action sequences (K) at runtime directly boosts performance. As K increases from 1 to 10, SEAL enjoys higher success rates–with a notable 15% gain on challenging compositional tasks—and shorter episode lengths (Fig. 6). Moreover, our method's inference time scales linearly with the number of samples with a slower growth than that of [6]. The main bottleneck is the VLM query time. This linear trend shows that we can obtain a practical performance boost at runtime simply by increasing the number of samples for verification.

However, the performance gains exhibit diminishing returns as the performance is ultimately bounded by two factors: the occasional inaccuracy of the VLM verifier, and more fundamentally, the quality of the base model's proposals. Our steering method can only select the best option from the provided candidates; if the underlying π_0-reason policy fails to generate any viable action sequences within the batch, verification cannot succeed.

VI. Limitations

While our method substantially improves robustness, execution consistency, and generalization by verifying reasoning–action alignment at runtime, it is not without limitations and a more detailed discussion is shown in Appendix C

Dependence on Base VLA Quality. Our runtime steering method selects among candidate actions sampled from reasoning VLAs. Thus, overall performance is ultimately capped by the quality and diversity of base VLA's proposals.

Verifier Reliability. Our method utilizes the GPT-4o as a VLM verifier for measuring reasoning–action alignment. While we find it as effective or more than specialized Q-value based verifiers, VLMs can produce incorrect judgments, particularly for fine-grained gripper-object interactions in wrist-view images or scenes with severe occlusion.

Latency and Compute Overhead. Although we employ asynchronous verification to mitigate the time of rollout and obtain lower latency than prior work [6], our method still introduces additional inference-time latency and cost due to sampling and evaluating multiple candidate sequences. Future work should investigate optimizing the inference infrastructure or use quantization to push the real-time nature of this approach.

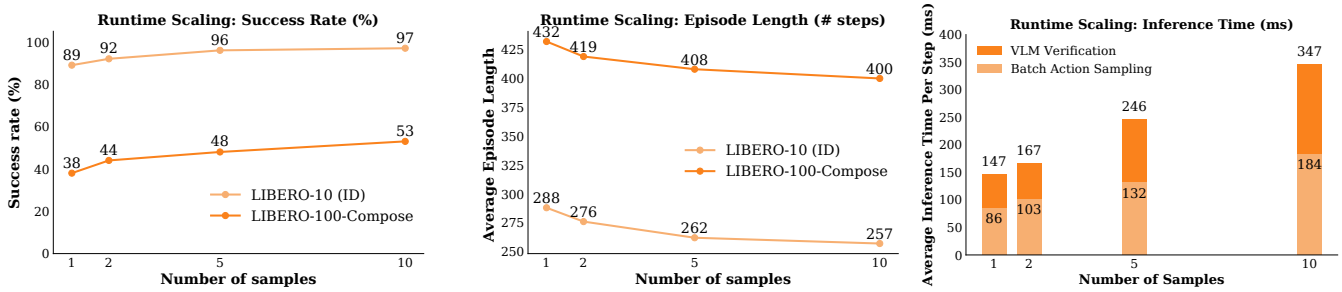


Fig. 6: **Runtime Scaling Law.** Increasing the number of candidate sequences (K) boosts success rates (**Left**) on in-distribution (ID) and behavior composition tasks, while reducing steps for task completion. (**Middle**). The inference time (**Right**) scales favorably, with our latency at $K = 10$ (347ms) lower than the $\sim$520ms in [6]. Results are averaged over 50 trials per task.

VII. Conclusion

This work addresses what we term the embodied CoT faithfulness gap, where the outcomes described by a VLA's textual reasoning mismatch its subsequent low-level actions. We introduce a training-free, runtime steering method that enforces this faithfulness by using a VLM to verify multiple candidate action sequences. By selecting the best execution of a plan out of many sampled action sequences, our method converts the policy's natural action diversity from a source of error into a strength to achieve the reasoning-action alignment. Our approach yields performance gains of up to 15% over strong baselines on challenging OOD and behavior composition tasks, demonstrating that verifying what a robot does against what it says is a critical step towards building more robust and trustworthy agents.

References

[1] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint*, 2025.

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

[3] O. X.-E. Team, "Open X-Embodiment: Robotic learning datasets and RT-X models," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[4] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh, "A taxonomy for evaluating generalist robot policies," *arXiv preprint*, 2025.

[5] C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang, "Thinkact: Vision-language-action reasoning via reinforced visual latent planning," *arXiv preprint*, 2025.

[6] J. Kwok, C. Agia, R. Sinha, M. Foutter, S. Li, I. Stoica, A. Mirhoseini, and M. Pavone, "Robomonkey: Scaling test-time sampling and verification for vision-language-action models," in *Conference on Robot Learning (CoRL)*, 2025.

[7] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, "Robotic control via embodied chain-of-thought reasoning," *Conference on Robot Learning (CoRL)*, 2024.

[8] F. Lin, R. Nai, Y. Hu, J. You, J. Zhao, and Y. Gao, "Onetwovla: A unified vision-language-action model with adaptive reasoning," *arXiv preprint*, 2025.

[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, 2022.

[10] Z. Zhou, Y. Zhu, M. Zhu, J. Wen, N. Liu, Z. Xu, W. Meng, R. Cheng, Y. Peng, C. Shen, *et al.*, "Chatvla: Unified multimodal understanding and robot control with vision-language-action model," *arXiv preprint*, 2025.

[11] M. Turpin, J. Michael, E. Perez, and S. Bowman, "Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting," *Advances in Neural Information Processing Systems*, 2023.

[12] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *Advances in Neural Information Processing Systems*, 2023.

[13] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, *et al.*, "pi 0.5: a vision-language-action model with open-world generalization," *arXiv preprint*, 2025.

[14] W. Chen, S. Belkhale, S. Mirchandani, O. Mees, D. Driess, K. Pertsch, and S. Levine, "Training strategies for efficient embodied reasoning," *Conference on Robot Learning (CoRL)*, 2025.

[15] Y. Wang, L. Wang, Y. Du, B. Sundaralingam, X. Yang, Y.-W. Chao, C. Perez-D'Arpino, D. Fox, and J. Shah, "Inference-time policy steering through human interactions," *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[16] M. Nakamoto, O. Mees, A. Kumar, and S. Levine, "Steering your generalists: Improving robotic foundation models via value guidance," in *Conference on Robot Learning (CoRL)*, 2024.

[17] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg, "Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress," in *Conference on Robot Learning (CoRL)*, 2024.

[18] P. Gupta, H. Admoni, and A. Bajcsy, "Adapting by analogy: Ood generalization of visuomotor policies via functional correspondence," *Conference on Robot Learning (CoRL)*, 2025.

[19] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," *arXiv preprint*, 2025.

[20] C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling llm test-time compute optimally can be more effective than scaling model parameters," *arXiv e-prints*, 2024.

[21] Y. Wu, R. Tian, G. Swamy, and A. Bajcsy, "From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment," in *Robotics: Science and System*, 2025.

[22] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh, "Physically grounded vision-language models for robotic manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[23] H. A. Alhaija, J. Alvarez, M. Bala, T. Cai, T. Cao, L. Cha, J. Chen, M. Chen, F. Ferroni, S. Fidler, *et al.*, "Cosmos-transfer1: Conditional world generation with adaptive multimodal control," *arXiv preprint*, 2025.

[24] Y. Mu, T. Chen, Z. Chen, S. Peng, Z. Lan, Z. Gao, Z. Liang, Q. Yu, Y. Zou, M. Xu, *et al.*, "Robotwin: Dual-arm robot benchmark with generative digital twins," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025.

[25] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2021.

[26] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-h. Sung, *et al.*, "Multilingual universal sentence encoder for semantic retrieval," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.