Chain-of-Correction: Progress-Aware Policy Steering via Anchor-Grounded Predictive Reasoning

Xuening Zhang[1], **Xiang Deng**[*1], **Jiayi Lin**[1], **Qi Lv**[1], **Xingbo Liu**[2], **Weili Guan**[1]
[1]Harbin Institute of Technology (Shenzhen), Shenzhen, Guangdong 518055, China
[2]Shandong Jianzhu University, Jinan, Shandong 250101, China

Abstract

Imitation learning policies for robotic manipulation inherently suffer from compounding errors. To mitigate this, inference-time policy steering introduces predictive reasoning to evaluate and refine action proposals before execution. However, existing methods typically anticipate consequences in unstructured pixel or latent spaces, leading to physically inconsistent predictions and unreliable error detection. Furthermore, their blindness to global progress often yields locally plausible corrections that fail to complete the overall task. To address these limitations, we propose **Chain-of-Correction (CoC)**, an inference-time steering framework driven by a single VLM. Shifting away from unstructured state predictions, CoC explicitly leverages anchor-grounded scene graphs to track task-relevant 3D spatial relations. Built upon this representation, a progress-aware dual-check mechanism verifies historical execution and predicts future task advancement to systematically intercept myopic action proposals. Through a failure-augmented supervised fine-tuning pipeline, CoC seamlessly unifies scene graph extraction, progress evaluation, and precise kinematic correction, eliminating the need for task-specific world models. Extensive experiments across the COLOSSEUM benchmark, a newly introduced offline protocol for predictive reasoning, and real-world tasks demonstrate that CoC significantly outperforms state-of-the-art baselines in fine-grained error detection and robust error recovery.

1 Introduction

Imitation learning, particularly when powered by generative models (*e.g.*, Diffusion Policies [1] and Vision-Language-Action models [2, 3, 4]), has become a dominant approach for acquiring complex robotic manipulation skills directly from expert demonstrations [5]. However, these purely data-driven policies remain fragile at deployment. The fundamental challenge arises from compounding errors [6, 7]: once a minor execution error occurs, the robot is pushed into an unseen state. Without the necessary knowledge or mechanisms to recover, these small deviations accumulate, ultimately leading to task failures. To mitigate this issue, previous efforts typically focus on modifying the policy training pipeline, either by injecting diverse recovery data [8, 9] or enforcing strict trajectory constraints [10]. Yet, these *retraining-based* approaches introduce an inherent trade-off between precise imitation and error recovery under environmental perturbations [11]. Recently, *inference-time policy steering* [11, 12, 13] has emerged as a compelling *predict-then-act* alternative. By introducing a separate predictive module (*e.g.*, a learned world model [13]) to simulate action consequences and refine candidate proposals, this paradigm decouples base action generation from predictive refinement, allowing for the test-time enhancement of frozen base policies. Consequently, it bypasses this costly trade-off, offering a promising path to achieve both precise imitation and generalizable error recovery.

[*]Corresponding author: dengxiang@hit.edu.cn

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

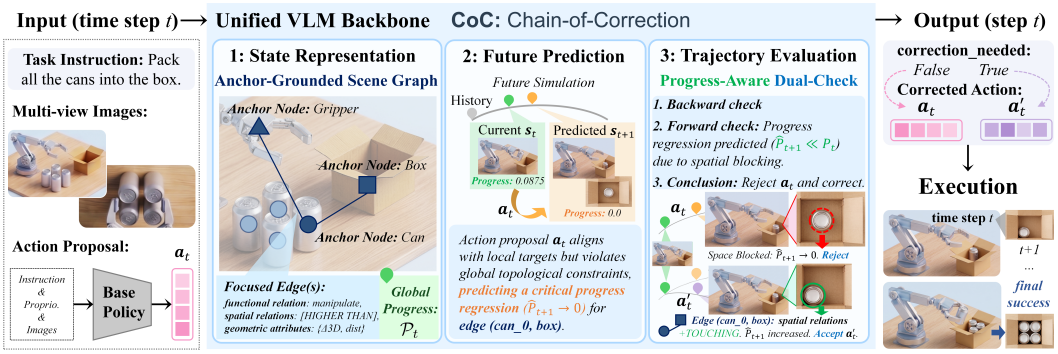


Figure 1: **Overview of CoC, a framework mitigating compounding errors via inference-time policy steering.** CoC performs anchor-grounded predictive reasoning, capturing explicit physical constraints missed by pixel or latent models. A progress-aware dual-check anticipates spatial consequences to intercept locally plausible yet globally detrimental actions, synthesizing valid corrections. A single VLM powers this pipeline without task-specific world models.

Despite the promise of inference-time policy steering, existing predictive modules are bottlenecked by two fundamental flaws in representing and assessing future states. First, they model the environment in either raw pixel spaces [13, 14] or abstract latent spaces [11, 12]. These representational spaces **lack object-centric geometric grounding**, yielding highly unreliable physical state predictions. Specifically, pixel-based methods are prone to generating physically infeasible frames that violate physical commonsense [15], while latent dynamics models inherently lose precise spatial details due to feature compression [16]. Consequently, as observed in our empirical analysis, the evaluator often rejects valid actions simply because their predicted future states are visually distorted. This triggers unnecessary corrections that disrupt the base policy.

Second, previous steering methods generally focus on reconstructing low-level visual features by minimizing pixel-wise differences or matching expert latent distributions, which leaves them **blind to global task progress**. This short-sightedness compromises the efficacy of policy correction in sequential tasks. For instance, consider the packing task illustrated in Figure 1, where a robot must pack four identical cans into a tightly constrained box. If the base policy erroneously proposes placing the first can in the center, standard evaluators tend to assess this action as "correct", since the predicted immediate state yields a low visual reconstruction error without obvious collisions. However, this myopic assessment ignores the global objective: the central placement physically blocks the remaining cans, guaranteeing eventual task failure.

To bridge these representational and evaluative gaps, we propose **Chain-of-Correction (CoC)**, a multi-phase embodied Chain-of-Thought framework driven by a single VLM. CoC explicitly grounds its reasoning on *anchor-grounded scene graphs* (Figure 1, Phase 1). It parses the scene into physical objects and explicitly tracks the geometric attributes of task-relevant edges, such as the dynamic 3D distance between the gripper and a target. Thus, this representation provides the explicit object-centric spatial grounding that compressed latents lack. Built upon this structured graph, CoC directly forecasts the future variations of these edges, such as whether the relative distance converges or undergoes a severe spatial deviation. By doing so, it avoids the physically impossible predictions that pixel-based models generate. Consequently, this anchor-grounded predictive reasoning ensures that the forward simulation evaluates rigorous physical consequences rather than mere unstructured feature reconstruction (Figure 1, Phase 2).

To further address the global progress blindness, CoC quantifies task progress using a continuous progress scalar derived from the scene graph. Driven by this metric, it structures the VLM's reasoning into a *progress-aware dual-check* mechanism (Figure 1, Phase 3). Before execution, a backward check analyzes the interaction history to identify errors already induced by previous steps, such as sudden progress regressions or unrecoverable physical stasis. Subsequently, a forward check verifies whether the base policy's proposal resolves these emerging deviations or exacerbates them. If the action proposal is deemed globally detrimental, CoC proactively rejects it and synthesizes a precise kinematic correction. Specifically, it computes a correction step along the 3D direction of focused edges, directly moving the robot toward the next milestone to advance global task progress. By

bridging exact geometric grounding with explicit progress tracking, CoC establishes an interpretable, geometry-anchored steering paradigm that reliably arrests compounding errors.

The main contributions of this work are summarized as follows:

- We propose Chain-of-Correction (CoC) to overcome global progress blindness in inference-time policy steering. Driven by a progress-aware dual-check mechanism, it identifies historical deviations via a backward check and evaluates task advancement via a forward check, intercepting myopic actions to mitigate compounding errors.
- We devise an anchor-grounded geometric reasoning mechanism that shifts error recovery from unstructured state prediction to structured geometric tracking. This decouples spatial topology from visual noise, circumventing the physical inconsistencies inherent in baseline predictive models.
- We develop a failure-augmented SFT pipeline that seamlessly unifies complex reasoning tasks, including geometric scene parsing, progress evaluation, and kinematic correction, within a single VLM, eliminating the need for task-specific world models.
- Evaluations across COLOSSEUM, a novel offline protocol for predictive reasoning, and real-robot tasks demonstrate the superiority of CoC in fine-grained error detection and robust recovery.

2 Related Work

Mitigating Compounding Errors in Imitation Learning. While behavior cloning (BC) is widely used for robotic imitation learning [4, 17, 18, 19], it inherently suffers from covariate shift. During deployment, minor execution deviations compound over time, causing the agent to enter unrecoverable out-of-distribution (OOD) states [6, 7]. Traditional solutions generally focus on retraining the BC policy. For example, *data augmentation* approaches aim to expand the expert distribution by querying human-in-the-loop interventions or injecting synthetic noise during training [8, 20]. Prominent examples of the former include DAgger [6] and its variants [21]. However, these corrective trajectories inevitably pollute the pure demonstrations required for precise imitation [11]. *Control-theoretic regularization* [10] incorporates dynamical penalties in the training objective, confining the policy to demonstrated states while sacrificing its generalization in unseen environments. In addition, *inverse reinforcement learning (IRL)* [22] relies on iterative retraining and on-policy rollouts to mitigate covariate shift, but incurs prohibitive RL exploration costs [23, 24]. Given the dilemmas of the above retraining-based methods, steering a frozen base policy at inference time emerges as a highly compelling alternative.

Inference-Time Policy Steering. By freezing the base policy and intervening only at deployment, inference-time policy steering avoids dataset pollution and costly online exploration. It enables generalizable error recovery via external predictive and reasoning models, which simulate action consequences either in abstract *latent spaces* [11, 12] or in *pixel spaces* [13, 14]. For instance, LPB [11] and FOREWARN [12] utilize latent dynamics models to predict deviations and pull the agent back to the expert manifold. GPC [13] and ReflectVLM [14] leverage video diffusion and VLMs to visually simulate and refine action proposals. However, both representational spaces lack object-centric geometric grounding: latent embeddings discard fine spatial details [16], while pixel models frequently generate physically unfeasible transitions [25]. This limitation prevents existing methods from effectively intercepting physical violations and achieving reliable error detection. Moreover, these methods operate without global task-progress awareness, often yielding myopic actions that trap the agent in unrecoverable dead-end states [26]. Consequently, robust inference-time steering requires a new framework that combines explicit spatial geometry to reliably detect physical deviations, with global progress awareness to ensure synthesized corrections lead to task completion.

Object-Centric Representations and Embodied Chain-of-Thought. To achieve object-centric geometric grounding, we turn to *scene graphs* [27, 28, 29], which encode objects as nodes and their physical relations as edges, offering stronger structural priors than pixels or latents [16]. Notably, MomaGraph [30] unifies spatial and functional relationships into a powerful task-driven representation. Alongside these representational advances, embodied Chain-of-Thought (CoT) via VLMs has advanced multi-step reasoning through active perception [31, 32], recursive memory [33], and dynamic graphs [34]. However, existing CoT frameworks predominantly function as *coupled action generators* that drive open-loop execution [32, 34], or *training-time supervisors* that provide reward signals [31, 33]. Lacking an independent evaluation mechanism, they cannot steer frozen policies or prevent error compounding in a closed-loop manner. To bridge this gap, CoC fundamentally

elevates embodied CoT into a *test-time, closed-loop evaluator*. It explicitly embeds fine-grained geometric attributes, such as relative 3D coordinates, into MomaGraph for predictive reasoning. This empowers the VLM to proactively intercept physical violations before execution and synthesize valid corrections for inference-time policy steering.

3 Method: Chain-of-Correction

This section details the CoC framework. We first formalize the inference-time policy steering problem and establish our core objectives: minimizing false positives and false negatives during error recovery (Section 3.1). To overcome the limitations of unstructured state prediction, we introduce an anchor-grounded state representation that explicitly extracts task-relevant spatial geometry (Section 3.2). Building upon this structured graph, Section 3.3 details a progress-aware dual-check mechanism that predictively intercepts physical violations and synthesizes precise kinematic recoveries. Finally, Section 3.4 describes the unified supervised fine-tuning pipeline that distills these sequential reasoning capabilities into a single VLM.

3.1 Problem formulation

We consider a rigid-object manipulation task specified by a language instruction $\ell \in \mathcal{L}$. At any time step t, the robot receives an observation history $O_t := o_{t-H+1:t}$ of length H. This history typically comprises multi-view images and proprioceptive states. The robot then executes a low-level kinematic action $a_t \in \mathcal{A}$. Let $\pi_{\text{base}}(a_t \mid O_t, \ell)$ denote a frozen, pre-trained base policy obtained via behavior cloning [1, 4]. During inference, minor execution deviations compound over time, gradually pushing the robot into unrecoverable states outside the expert distribution [6, 35, 36].

To mitigate compounding errors without retraining π_{base}, inference-time policy steering introduces an external evaluator f_{steer}. Given O_t and a proposed action $a_t \sim \pi_{\text{base}}(\cdot \mid O_t, \ell)$, f_{steer} employs predictive reasoning to verify the validity of a_t prior to execution. Formally, the steering problem seeks to determine the final executed action a_t^* via:

$$a_t^* = \begin{cases} a_t' & \text{if } f_{\text{steer}} \text{ predicts a compounding failure,} \\ a_t & \text{otherwise,} \end{cases} \tag{1}$$

where a_t' is a refined action obtained via optimization [13], candidate ranking [12], or synthesis.

While Eq. 1 establishes the fundamental paradigm, the efficacy of f_{steer} hinges on its reliability in error detection. Prior methods typically approximate transition dynamics $\hat{p}(x_{t+1} \mid O_t, a_t)$ in unstructured pixel [13, 14] or latent spaces [11, 12]. However, as empirically validated in Section 4, the unstructured prediction paradigm is bottlenecked by two fundamental errors: (1) **False positives (over-pessimistic interception)**: Lacking explicit geometric grounding, f_{steer} often misinterprets generative errors [37] or latent feature ambiguities as actual physical failures. This forces the rejection of valid a_t, triggering unnecessary corrections that disrupt fluid execution. (2) **False negatives (myopic approval)**: Operating without global task progress awareness, f_{steer} tends to approve immediately safe yet progress-hindering actions, such as placing an object that obstructs subsequent steps. This failure to identify long-horizon dead-ends undermines sequential task completion.

To overcome these bottlenecks, a robust evaluator must minimize both false positives and false negatives. CoC achieves this by abandoning unstructured state prediction in favor of explicit geometric tracking (Section 3.2) and a progress-aware dual-check mechanism (Section 3.3).

3.2 Anchor-grounded state representation

To address the false positives induced by unstructured predictions, CoC abandons the reliance on unconstrained pixel or latent generations. Instead, we project the observation O_t into an anchor-grounded scene graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ (Fig. 2, Left). This explicit geometric interface ensures that the steering module builds upon rigorous spatial tracking. To instantiate $\mathcal{G}_t$, three embodied augmentations are tailored for robotic manipulation:

Kinematic Chain Anchors ($\mathcal{V}_t$): The node set isolates the task-relevant kinematic chain, including the end-effector v_{gripper}, the manipulation object v_{MO}, and the spatial target v_{target}. This task-centric anchoring localizes the precise geometric constraints bounding potential execution deviations.

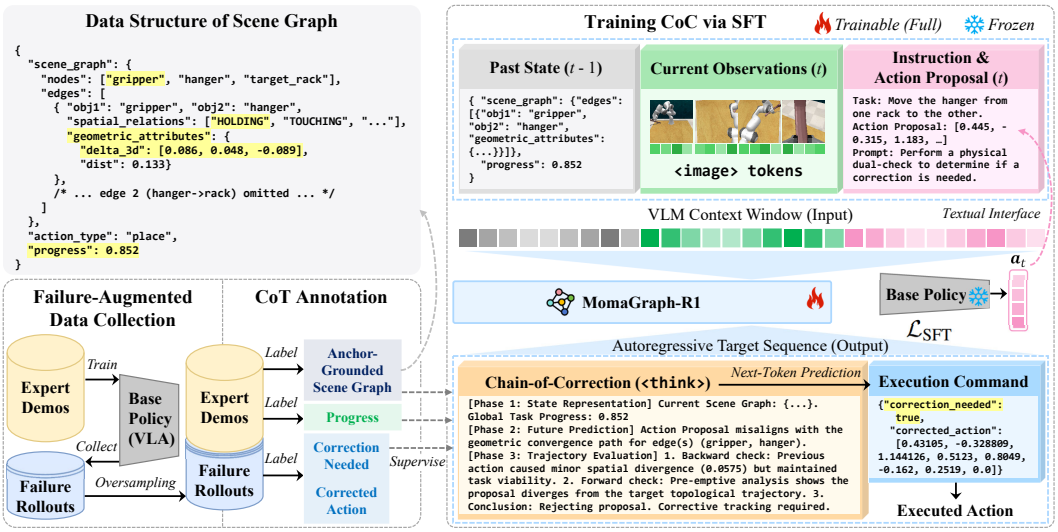


Figure 2: **Training pipeline and architecture of CoC. (Left)** Both expert demonstrations and oversampled failure rollouts are annotated with scene graphs, progress, and correction labels to construct SFT data. **(Right)** During SFT, a single VLM processes a unified context (the past state, current observations, task instruction, and a text-based action proposal from the frozen base policy) to autoregressively generate a step-by-step physical rationale followed by the execution command.

Geometrically-Enriched Edges ($\mathcal{E}_t$): To enable precise steering, the directed edges augment standard discrete relations [30] with continuous spatial metrics, explicitly extracting relative 3D coordinate offsets and Euclidean distances.

Explicit Contact Verification: Unstructured predictive models often mask mid-trajectory object slipping as stable grasps due to visual occlusions and contact ambiguities [38, 39], leading to undetected grasp failures. To mitigate this, $\mathcal{G}_t$ incorporates an explicit `HOLDING` state. This state represents a stable kinematic bond between the end-effector and the object, which is activated via an inter-frame displacement threshold. Further details are provided in Appendix B.2.

Ultimately, this anchor-grounded and geometrically enriched graph establishes a robust state representation. To construct the supervision, we compute a continuous global progress target $\mathcal{P}_t \in [0, 1]$ from the spatial relations and geometric attributes in $\mathcal{G}_t$ to quantify progress toward the task goal, as detailed in Appendix B.2.

Within each stage, local progress p_t rises from 0 as the gripper approaches the object, reaching 0.35 at TOUCHING and 0.5 when HOLDING is verified. It then increases from 0.5 to 0.85 as the object moves closer to its target during transport. AROUND marks the start of alignment, during which task-specific spatial criteria determine progress up to 1. For a task with N stages and $k < N$ completed stages, $\mathcal{P}_t = (k + p_t)/N$, and $\mathcal{P}_t = 1$ once all stages are completed. During training, the VLM learns to predict this target for the dual-check evaluation in Section 3.3.

3.3 Progress-aware dual-check evaluation

Building upon the geometrically-grounded graph $\mathcal{G}_t$ and the progress scalar $\mathcal{P}_t$, we design an algorithmic supervisor. This supervisor serves a dual purpose: it generates the ground-truth reasoning traces for SFT, and its logic is ultimately internalized by the VLM for autonomous steering. While the geometric graph inherently alleviates false positives, this supervisor is specifically designed to intercept false negatives and prevent physical stasis. To achieve this, it conducts a bidirectional verification that assesses both past deviations and future consequences:

Backward Check: Unlike prior methods blind to past errors [11, 13], it compares $\mathcal{G}_t$ against $\mathcal{G}_{t-1}$ to detect severe task regressions like losing the `HOLDING` state, or physical stasis where $\mathcal{P}_t$ remains frozen across consecutive frames. This explicitly flags wasteful actions trapped in local minima, assigning a `correction_needed = True` label to break stagnation.

Forward Check: Subsequently, it evaluates proposal a_t by analytically projecting future spatial distances and $\mathcal{P}_{t+1}$. Leveraging the explicit 3D coordinates of $\mathcal{G}_t$, this projection applies lightweight yet precise kinematic translation, entirely bypassing the overhead and hallucination risks of heavy generative dynamics models. If the prediction indicates imminent spatial divergences from the task goal or progress regressions, the forward check rejects a_t, effectively intercepting myopic approvals.

Synthesis of Kinematic Correction: Upon flagging an error, the supervisor heuristically synthesizes a ground-truth corrective action a'_t. Depending on the error type identified by the dual-check, it applies either a direct spatial convergence vector to pull the robot back on track, or a compliance lift with lateral noise $\mathcal{N}(0, \sigma^2)$ [40] to break physical deadlocks. The specific mathematical formulations and triggering conditions for these recoveries are detailed in Appendix A.

3.4 Unified failure-augmented SFT pipeline

To achieve autonomous steering, we distill the supervisor's physics and control logic into a single VLM, MomaGraph-R1 [30]. This unification eliminates the architectural overhead of chaining separate specialized modules for scene parsing, progress estimation, and correction synthesis.

Failure-Augmented Data Collection. Training the steering evaluator exclusively on expert demonstrations restricts its state-action distribution, hindering error identification and recovery from execution deviations [11, 12]. To capture a wider distribution including erroneous states, we deploy the frozen base policy to collect failure rollouts. The algorithmic supervisor processes both expert demos and failure rollouts for *CoT annotation* (Fig. 2, Left). It labels each frame with the anchor-grounded scene graph, task progress, a binary `correction_needed` flag, and the kinematically synthesized `corrected_action` a'_t. As detailed in Appendix A, this field retains the original proposal for valid frames, while replacing the rejected proposals with precise geometric recovery commands to physically guide the end-effector back on track.

Correction Data Balancing. Since valid execution frames heavily outnumber erroneous ones, training on the raw data distribution biases the VLM to trivially predict `correction_needed: false` by exploiting a statistical shortcut to minimize loss [41]. To counter this, we partition the dataset based on the `correction_needed` label into a valid pool $\mathcal{D}_{\text{valid}}$ and an erroneous pool $\mathcal{D}_{\text{erroneous}}$. We then oversample $\mathcal{D}_{\text{erroneous}}$ to construct a balanced training mixture $\mathcal{D}_{\text{mix}}$. This strategy prevents valid frames from dominating the loss, yielding sufficient optimization signals to successfully avoid majority-class degeneration [42].

Unified Supervised Fine-Tuning. We cast the reasoning process as an autoregressive next-token prediction task [31, 33]. The input context c_t (Fig. 2, Right) comprises the *past state* at $t-1$, *current observations* at t, task *instruction*, and the base policy's *action proposal* a_t. The VLM ψ_θ generates the target sequence $y = [e_t, a_t^*]$, where e_t represents the *Chain-of-Correction* trace detailing the bidirectional reasoning phases, and a_t^* is the *execution command* formatted as a JSON object. We optimize ψ_θ using the standard log-likelihood objective over the balanced dataset $\mathcal{D}_{\text{mix}}$:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(c_t, y)\sim\mathcal{D}_{\text{mix}}} \sum_{i=1}^{|y|} \log p_\theta(y_i \mid c_t, y_{<i}). \tag{2}$$

This yields an autonomous steering VLM ψ_{SFT} that executes the full Chain-of-Correction from the input context c_t, without relying on task-specific dynamics models. At deployment, this context consists of multi-view images, the language instruction, the base policy's action proposal, and a history comprising the previously executed action and the model's own previous structured output. Simulator states or external perception models are used only to construct offline SFT targets.

4 Experiments

This section evaluates the proposed CoC framework by answering the following questions:

- **Q1 (Fine-Grained Error Detection):** Does CoC achieve more reliable error detection than state-of-the-art steering methods and zero-shot VLMs, effectively minimizing the false positives caused by physically inconsistent predictions?
- **Q2 (Robust Error Recovery):** Can CoC effectively steer the base policy to mitigate compounding errors under OOD visual perturbations?

Table 1: **Evaluation of predictive reasoning and correction.** We introduce a novel offline protocol to evaluate SOTA VLMs and steering baselines against structured correction targets. The assessment spans three critical dimensions: Error Detection for binary failure identification, Progress Estimation for global task awareness, and Action Precision for kinematic refinement of action proposals. Unlike baselines lacking explicit progress awareness (marked as –), CoC couples anchor-grounded scene graphs with accurate progress estimation for error detection and kinematic refinement.

Models	Error Detection			Progress Estimation		Action Precision		
	F1-Score ↑	Precision ↑	Recall ↑	RMSE ↓	MAE ↓	Trans. (L2) ↓	Rot. (L2) ↓	Grip. (Err %) ↓
Zero-Shot General VLMs								
Qwen2.5-VL-7B-Instruct [44]	0.397 ±0.000	0.683 ±0.000	0.280 ±0.000	0.126 ±0.000	0.072 ±0.000	0.043 ±0.000	0.546 ±0.000	21.0 ±0.0
Llama-3.2-11B-Vision-Instruct [45]	0.539 ±0.000	0.496 ±0.000	0.590 ±0.000	0.359 ±0.000	0.242 ±0.000	0.085 ±0.000	0.542 ±0.000	22.8 ±0.0
Gemini 3.1 Pro [46]	0.547 ±0.024	0.672 ±0.066	0.463 ±0.017	0.092 ±0.012	0.044 ±0.004	0.100 ±0.020	0.773 ±0.251	27.8 ±0.4
GPT-5.4 [47]	0.546 ±0.022	0.622 ±0.029	0.487 ±0.017	0.070 ±0.003	0.049 ±0.001	0.155 ±0.014	0.553 ±0.043	28.0 ±1.8
Embodied Reasoning VLMs								
MomaGraph-R1 [30]	0.085 ±0.000	0.278 ±0.000	0.050 ±0.000	0.086 ±0.000	0.047 ±0.000	0.030 ±0.000	0.478 ±0.000	**20.5** ±0.0
VLM-3R [48]	0.472 ±0.000	0.475 ±0.000	0.470 ±0.000	0.335 ±0.000	0.230 ±0.000	0.084 ±0.000	0.470 ±0.000	25.6 ±0.0
Policy Steering Methods								
FOREWARN [12]	0.667 ±0.000	0.500 ±0.000	**1.000** ±0.000	–	–	0.089 ±0.002	**0.071** ±0.007	24.8 ±3.0
LPB [11]	0.450 ±0.010	0.367 ±0.007	0.580 ±0.016	–	–	0.222 ±0.035	0.241 ±0.039	32.9 ±3.1
GPC [13]	0.667 ±0.000	0.500 ±0.000	**1.000** ±0.000	–	–	0.096 ±0.008	0.092 ±0.007	26.9 ±4.0
CoC (Ours)	**0.851** ±0.000	**1.000** ±0.000	0.740 ±0.000	**0.037** ±0.000	**0.007** ±0.000	**0.020** ±0.000	0.121 ±0.000	23.0 ±0.0

- **Q3 (Component Necessity):** Are the *anchor-grounded scene graph* and *progress-aware dual-check* necessary for boosting the overall steering performance?
- **Q4 (Real-Robot Deployment & Long-Horizon Recovery):** How well does CoC intercept long-horizon task failures, namely false negatives, and achieve real-world error recovery?

To this end, we first introduce a novel offline protocol to evaluate predictive reasoning capabilities *w.r.t.* error detection, progress estimation, and action precision (Section 4.1). Next, we benchmark closed-loop policy steering performance against SOTA baselines across 14 perturbation axes on COLOSSEUM [43] (Section 4.2). Finally, we dissect our algorithmic design through ablation study (Section 4.3) and demonstrate the practical efficacy of CoC in real-robot manipulation (Section 4.4).[2]

4.1 Offline evaluation of predictive reasoning

To answer **Q1**, we evaluate predictive reasoning using an offline protocol with structured correction targets generated by the same annotation pipeline used for SFT (Appendix D). As Table 1 shows, both zero-shot general VLMs and specialized embodied VLMs struggle to achieve reliable error detection, yielding sub-optimal F1-scores (≤ 0.547). Meanwhile, prior steering methods that predict over unstructured states exhibit a false positive bias: while FOREWARN and GPC achieve perfect recall, these baselines yield poor precision of at most 0.500, frequently flagging valid actions for correction, as illustrated by the qualitative examples in Appendix D. Conversely, equipped with anchor-grounded scene graphs, CoC produces no false positives on this benchmark, achieving a perfect precision of 1.000 and the highest F1-score of 0.851. This supports CoC's reliability in identifying actions that require correction while avoiding false alarms on this benchmark. Furthermore, while the steering baselines do not output progress estimates, CoC accurately estimates global progress (MAE: 0.007) and synthesizes kinematic corrections with a translation error of 0.020. We assess practical steering effectiveness primarily through closed-loop task success in Sections 4.2 and 4.4.

4.2 Performance comparison on COLOSSEUM

To answer **Q2**, we benchmark inference-time policy steering on 7 COLOSSEUM [43] tasks under 14 OOD visual perturbations, as detailed in Appendix E. As Table 2 shows, CoC achieves the highest average success rate (71.6%). Notably, under the joint *All Perturbations* setting, the base policy drops to 24.8%, whereas CoC maintains 67.6% by robustly mitigating amplified compounding errors. By contrast, pixel- and latent-based baselines often perform worse than the base policy under specific perturbations, such as FOREWARN collapsing to 27.2% on *RLBench Variations*, as visual noise

[2]The main-text CoC results use CoC-Normal, which generates the full reasoning trace. CoC-Fast uses shorter outputs and is evaluated separately in Appendix C.

Table 2: **Comparison with SOTA methods on COLOSSEUM.** The "Avg. Success" column reports the average success rate (%) across 14 generalization settings on 7 manipulation tasks. For complete per-task success rates across all 14 perturbations, please refer to Appendix E. "All Perturbations" jointly applies the perturbations in a separate evaluation condition. Its success rate is not an average over individual perturbation settings. This condition need not be the hardest for every method.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Texture	RO Texture	MO Size
BridgeVLA (Base)	NeurIPS (2025)	66.7	73.1 ±33.3	24.8 ±30.4	58.5 ±36.1	60.5 ±33.9	43.4 ±36.7	**88.0** ±0.0	74.5 ±30.7
FOREWARN	RSS (2025)	50.8	74.5 ±35.8	67.4 ±39.6	68.0 ±39.4	58.9 ±46.5	**46.7** ±42.7	86.7 ±0.0	70.2 ±28.3
LPB	NeurIPS (2025)	66.9	**75.6** ±32.2	25.3 ±32.2	56.9 ±35.4	60.5 ±30.6	39.3 ±34.0	86.0 ±3.3	77.1 ±29.1
GPC	RA-L (2026)	67.6	73.7 ±32.6	27.8 ±33.8	60.2 ±36.3	**61.1** ±32.7	43.4 ±35.4	84.0 ±4.0	78.2 ±26.4
CoC-Normal	–	**71.6**	71.8 ±31.1	**67.6** ±37.1	**70.9** ±36.5	**61.1** ±34.0	**46.7** ±36.0	**88.0** ±0.0	**78.5** ±29.8

Models	Reference	Light Color	RLBench Variations	Background Texture	Table Texture	Camera Pose	Distractor Objects	Table Color	RO Size
BridgeVLA (Base)	NeurIPS (2025)	66.1 ±31.0	73.0 ±31.1	73.9 ±32.1	**74.1** ±32.1	**69.3** ±35.9	68.6 ±31.6	77.0 ±26.4	82.7 ±0.0
FOREWARN	RSS (2025)	45.7 ±30.2	27.2 ±27.1	30.3 ±27.8	34.9 ±31.2	27.4 ±32.6	36.0 ±31.8	41.1 ±35.3	70.7 ±0.0
LPB	NeurIPS (2025)	62.5 ±29.1	**75.2** ±30.0	**74.3** ±29.2	70.7 ±29.8	69.0 ±35.5	70.7 ±32.2	76.4 ±27.5	**92.0** ±0.0
GPC	RA-L (2026)	65.9 ±31.0	73.5 ±32.0	74.1 ±29.4	71.4 ±32.3	67.6 ±37.9	70.7 ±32.2	76.4 ±25.6	**92.0** ±0.0
CoC-Normal	–	**68.4** ±32.9	72.6 ±34.0	70.3 ±30.0	68.4 ±27.9	**69.3** ±37.9	**71.2** ±27.6	**77.1** ±25.8	**92.0** ±0.0

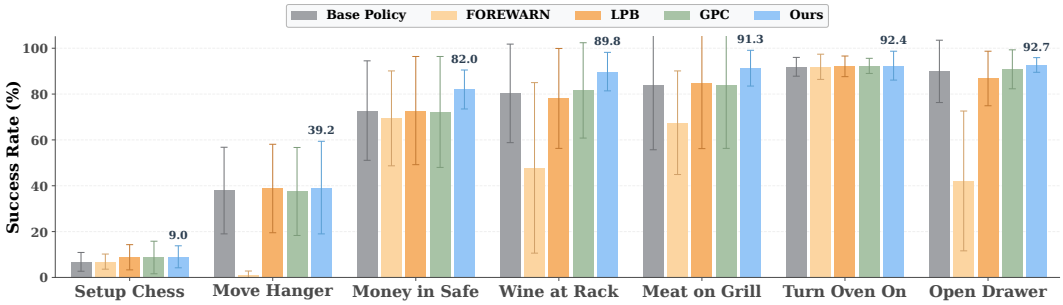


Figure 3: **Per-task success rates averaged across various perturbations.** CoC consistently outperforms all inference-time policy steering baselines, achieving an average improvement of over 3.8%.

corrupts their unstructured predictive reasoning. Furthermore, Fig. 3 shows that CoC achieves the highest mean success rate among the compared steering methods on all tasks. CoC also shows smaller success-rate fluctuations than LPB on *Money in Safe*. These results suggest that grounding predictive reasoning in 3D geometry can improve resilience to task-irrelevant visual perturbations and support robust OOD recovery. CoC achieves higher average success under perturbations, while under "No Perturbations", its 71.8% success is lower than BridgeVLA's 73.1%. This contrast highlights the importance of selective intervention: corrections can aid recovery from policy errors, while unnecessary changes to valid proposals may introduce errors.

4.3 Ablation study

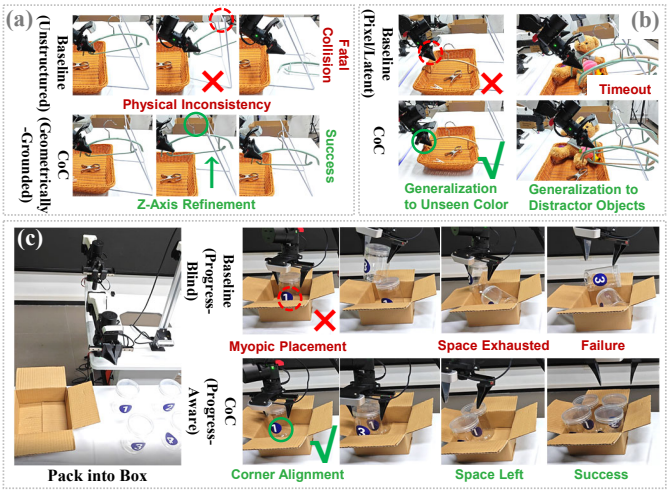
To answer **Q3**, we dissect CoC's algorithmic design on COLOSSEUM by systematically withholding key representational and reasoning modules, as detailed in Appendix F.

As Table 3 demonstrates, bypassing foresight (**w/o Prediction**) catastrophically drops the average success rate to 18.1%, as the reactive agent fails to preempt impending physical violations before execution. Similarly, disabling global task context (**w/o Progress**) degrades performance to 52.1%. Without holistic awareness, the evaluator suffers from false negatives, authorizing actions that are locally collision-free but globally counterproductive, ultimately leaving the robot stuck in unrecoverable situations. Furthermore, robust recovery strictly relies on precise spatial and temporal grounding. Removing fine-grained spatial metrics (**w/o Geometry**, dropping to 50.6%) limits the VLM to coarse topological cues, preventing accurate spatial adjustments under visual perturbations. Finally, masking past observations (**w/o History**) impairs the evaluator (51.9%), rendering it unable to track gradual drifts and rectify already-committed errors. These results confirm that effective inference-time steering demands both fine-grained 3D anchoring and bi-directional temporal reasoning.

Table 3: **Ablation study results.** All components of anchor-grounded scene graph and progress-aware dual-check are demonstrated necessary to realize effective interventions.

Model Variants	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Texture	RO Texture	MO Size
CoC w/o Geometry	50.6	71.4 ±37.9	67.2 ±40.8	66.3 ±38.4	57.6 ±43.9	46.0 ±38.1	86.7 ±8.2	68.2 ±29.3
CoC w/o History	51.9	**72.0** ±37.8	67.4 ±41.8	68.0 ±40.9	57.9 ±45.0	**46.7** ±39.9	85.3 ±5.5	74.7 ±30.1
CoC w/o Prediction	18.1	31.4 ±34.4	32.6 ±36.6	31.4 ±33.7	12.8 ±17.3	6.0 ±2.8	10.0 ±14.1	29.3 ±28.3
CoC w/o Progress	52.1	71.4 ±34.2	**67.6** ±37.7	66.3 ±40.8	59.2 ±43.5	46.0 ±42.0	**88.0** ±4.0	68.7 ±27.8
CoC	**71.6**	71.8 ±31.1	**67.6** ±37.1	**70.9** ±36.5	**61.1** ±34.0	**46.7** ±36.0	**88.0** ±0.0	**78.5** ±29.8

Model Variants	Light Color	RLBench Variations	Background Texture	Table Texture	Camera Pose	Distractor Objects	Table Color	RO Size
CoC w/o Geometry	45.1 ±29.3	28.0 ±29.0	29.1 ±26.8	36.0 ±30.9	28.6 ±32.1	34.9 ±31.8	42.9 ±31.3	72.0 ±0.0
CoC w/o History	44.6 ±31.9	26.3 ±28.2	28.6 ±28.0	36.6 ±35.3	28.0 ±35.0	38.9 ±32.5	44.0 ±36.2	80.0 ±0.0
CoC w/o Prediction	19.4 ±21.4	15.4 ±22.4	14.9 ±21.9	18.3 ±27.8	16.6 ±26.9	19.4 ±26.2	18.9 ±27.4	8.0 ±0.0
CoC w/o Progress	48.6 ±29.7	30.9 ±26.1	30.3 ±28.1	35.4 ±30.0	28.6 ±31.8	34.9 ±30.7	44.6 ±32.8	80.0 ±0.0
CoC	**68.4** ±32.9	**72.6** ±34.0	**70.3** ±30.0	**68.4** ±27.9	**69.3** ±37.9	**71.2** ±27.6	**77.1** ±25.8	**92.0** ±0.0



(d) Quantitative Evaluation

Method	SR (↑)	FPR (↓)	FNR (↓)
Task 1: Move Hanger			
GPC	75.0%	15.0%	10.0%
LPB	80.0%	10.0%	10.0%
CoC	**95.0%**	**0.0%**	**5.0%**
Task 2: Unseen MO Color			
GPC	25.0%	60.0%	15.0%
LPB	35.0%	55.0%	10.0%
CoC	**85.0%**	**10.0%**	**5.0%**
Task 3: Visual Distractors			
GPC	5.0%	90.0%	**5.0%**
LPB	10.0%	85.0%	**5.0%**
CoC	**80.0%**	**15.0%**	**5.0%**
Task 4: Pack into Box			
GPC	15.0%	**5.0%**	80.0%
LPB	20.0%	**5.0%**	75.0%
CoC	**85.0%**	**5.0%**	**10.0%**

Figure 4: **Results of real-world error recovery. (a)-(c) Qualitative Results:** CoC demonstrates superiority in overcoming (a) physical inconsistencies leading to fatal collisions, (b) false-positive action rejections induced by OOD visual shifts, and (c) myopic placements causing long-horizon dead-ends. **(d) Quantitative Results:** Success Rate (SR), False Positive Rate (FPR), and False Negative Rate (FNR) across four tasks. CoC effectively balances error detection, avoiding both catastrophic freezing (high FPR) and short-sighted approvals (high FNR).

4.4 Real-robot deployment and long-horizon recovery

To answer **Q4**, we evaluate inference-time correction of a frozen BridgeVLA policy on a physical Agilex ALOHA robot [49] across four representative tasks. CoC is fine-tuned on real-robot SFT data constructed from demonstrations and failure rollouts for Move Hanger (Task 1) and Pack into Box (Task 4). We also evaluate CoC on Move Hanger variants with an unseen object color (Task 2) or distractor objects (Task 3), without additional task-specific CoC fine-tuning. See Appendix G for comprehensive details.

Physical Consistency: In the basic *Move Hanger* task, GPC and LPB frequently suffer fatal collisions due to slight Z-axis deviations (Fig. 4a). Through geometric grounding, CoC executes Z-axis spatial refinements to prevent clipping, boosting success rate to 95.0% (vs. 75.0% and 80.0% for baselines).

Visual Generalization (FPR Reduction): As shown in Figs. 4b and d, GPC and LPB suffer severe false-positive freezing under unmodeled visual shifts in Tasks 2 and 3. Specifically, their False Positive Rate (FPR) spikes to 90.0% and 85.0% respectively in Task 3. This failure stems from their intrinsic entanglement of physical dynamics with unstructured visual features, which misclassifies benign visual variations as geometric violations. Conversely, CoC maintains success rates between

80.0% and 85.0% with substantially lower FPRs. It achieves this fluent execution because its anchor-grounded reasoning natively decouples spatial topology from visual noise.

Long-Horizon Recovery (FNR Interception): In the sequential task *Pack into Box*, GPC and LPB yield 80.0% and 75.0% False Negative Rates (FNRs) by approving myopic placements that exhaust spatial capacity. This exposes their inherent global progress blindness. In contrast, CoC effectively intercepts these false negatives and reduces FNR to 10.0%. Empowered by progress-aware steering, CoC preemptively refines trajectories such as corner alignment to preserve physical space, ensuring maximum global progress (Fig. 4c).

5 Conclusion

This paper introduces CoC, an inference-time steering framework to mitigate compounding errors in robotic manipulation. CoC grounds its predictive reasoning in anchor-based scene graphs, thereby decoupling geometric evolution from visual noise and reducing physically inconsistent predictions. By embedding progress-aware dual-checks, CoC successfully prevents local reasoning failures, correcting myopic actions to ensure task completion. Extensive offline, simulated, and real-world evaluations confirm that this unified reasoning chain enables superior reliability in error detection and robust OOD recovery, providing a generalized paradigm for dependable robotic manipulation.

Limitations and Future Work. Currently, CoC's inference latency is bottlenecked by the autoregressive generation of reasoning tokens, constraining its deployment in ultra-high-frequency control loops. In future, we will explore reasoning distillation or speculative decoding techniques for acceleration. CoC currently focuses on rigid-object manipulation with visually observable failures that can be corrected through end-effector actions. Deformable-object manipulation and tasks requiring sustained reasoning about contact forces or slip remain outside the current scope. The same applies to dexterous in-hand or multi-arm manipulation involving unmodeled joint or contact states. Extending to these settings calls for appropriate changes to the representation, sensing, or action interface, followed by separate empirical validation. Furthermore, since the closed-loop evaluations reported here use BridgeVLA as the base policy, future work will apply CoC to diverse generative base policies to validate its broader applicability.

Acknowledgments and Disclosure of Funding

This work is supported by National Natural Science Foundation of China (Grant No. 62406092), National Natural Science Foundation of China (Grant No. U24B20175), Shenzhen Science and Technology Program (Grant No. KJZD20240903100017022), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2025A1515010169), Shenzhen Science and Technology Program (Grant No. KQTD20240729102207002). The authors declare no competing interests.

References

[1] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026.

[2] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th*

Conference on Robot Learning, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023.

[3] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2024.

[4] Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. BridgeVLA: Input-output alignment for efficient 3d manipulation learning with vision-language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[5] Zezeng Li, Alexandre Chapin, Enda Xiang, Rui Yang, Bruno Machado, Na Lei, Emmanuel Dellandrea, Di Huang, and Liming Chen. Robotic manipulation via imitation learning: Taxonomy, evolution, benchmark, and challenges, 2025. URL `https://arxiv.org/abs/2508.17449`.

[6] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[7] Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. In Nika Haghtalab and Ankur Moitra, editors, *Proceedings of Thirty Eighth Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pages 5248–5351. PMLR, 30 Jun–04 Jul 2025.

[8] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 143–156. PMLR, 13–15 Nov 2017.

[9] Allan Zhou, Moo Jin Kim, Lirui Wang, Pete Florence, and Chelsea Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17907–17917, 2023. doi: 10.1109/CVPR52729.2023.01717.

[10] Shaunak A. Mehta, Yusuf Umut Ciftci, Balamurugan Ramachandran, Somil Bansal, and Dylan P. Losey. Stable-bc: Controlling covariate shift with stable behavior cloning. *IEEE Robotics and Automation Letters*, 10(2):1952–1959, 2025. doi: 10.1109/LRA.2025.3526439.

[11] Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=FUd016XD4d`.

[12] Yilin Wu, Thomas Tian, Gokul Swamy, and Andrea Bajcsy. From Foresight to Forethought: VLM-In-the-Loop Policy Steering via Latent Alignment. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.076.

[13] Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 11(5):5534–5541, 2026. doi: 10.1109/LRA.2026.3673995.

[14] Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation. In *9th Annual Conference on Robot Learning*, 2025. URL `https://openreview.net/forum?id=iq0wUf6dUy`.

[15] Fanqing Meng, Jiaqi Liao, Xinyu Tan, Quanfeng Lu, Wenqi Shao, Kaipeng Zhang, Yu Cheng, Dianqi Li, and Ping Luo. Towards world simulator: Crafting physical commonsense-based benchmark for video generation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=dIjMswSzgF.

[16] Bo Ai, Stephen Tian, Haochen Shi, Yixuan Wang, Tobias Pfaff, Cheston Tan, Henrik I Christensen, Hao Su, Jiajun Wu, and Yunzhu Li. A review of learning-based dynamics models for robotic manipulation. *Science Robotics*, 10(106):eadt1497, 2025.

[17] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=rif3a5NAxU6.

[18] Xuening Zhang, Qi Lv, Xiang Deng, Miao Zhang, Xingbo Liu, and Liqiang Nie. Cortical policy: A dual-stream view transformer for robotic manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eWe8zqGvs5.

[19] Qi Lv, Hao Li, Xiang Deng, Rui Shao, Yinchuan Li, Jianye Hao, Longxiang Gao, Michael Yu Wang, and Liqiang Nie. Spatial-temporal graph diffusion policy with kinematic modeling for bimanual robotic manipulation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17394–17404, 2025. doi: 10.1109/CVPR52734.2025.01621.

[20] Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. CCIL: Continuity-based data augmentation for corrective imitation learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LQ6LQ8f4y8.

[21] Xiaomeng Xu, Yifan Hou, Zeyi Liu, and Shuran Song. Compliant residual DAgger: Improving real-world contact-rich manipulation with human corrections. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=cjcm5LYVWm.

[22] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkHywl-A-.

[23] Juntao Ren, Gokul Swamy, Steven Wu, Drew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=2zI2scD2Iz.

[24] Nicolas Espinosa Dice, Gokul Swamy, Sanjiban Choudhury, and Wen Sun. Efficient inverse reinforcement learning without compounding errors. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*, 2024. URL https://openreview.net/forum?id=j8XVuNuU0Q.

[25] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=D5RNACOZEI.

[26] Hongyu Yan, Qiwei Li, Jiaolong Yang, and Yadong Mu. Progressvla: Progress-guided diffusion policy for vision-language robotic manipulation, 2026. URL https://arxiv.org/abs/2603.27670.

[27] Chenyangguang Zhang, Alexandros Delitzas, Fangjinhua Wang, Ruida Zhang, Xiangyang Ji, Marc Pollefeys, and Francis Engelmann. Open-vocabulary functional 3d scene graphs for real-world indoor spaces. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19401–19413, 2025. doi: 10.1109/CVPR52734.2025.01807.

[28] Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural*

Information Processing Systems, volume 37, pages 5285–5307. Curran Associates, Inc., 2024. doi: 10.52202/079017-0171. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/098491b37deebbe6c007e69815729e09-Paper-Conference.pdf.

[29] Juexiao Zhang, Gao Zhu, Sihang Li, Xinhao Liu, Haorui Song, Xinran Tang, and Chen Feng. Multiview scene graph. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 17761–17788. Curran Associates, Inc., 2024. doi: 10.52202/079017-0565.

[30] Yuanchen Ju, Yongyuan Liang, Yen-Jen Wang, Gireesh Nandiraju, Yuanliang Ju, Seungjae Lee, Qiao Gu, Elvis Hsieh, Furong Huang, and Koushil Sreenath. Momagraph: State-aware unified scene graphs with vision-language models for embodied task planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3eTr9dGwJv.

[31] Heng Zhou, Li Kang, Yiran Qin, Xiufeng Song, Ao Yu, Zilu Zhang, Haoming Song, Kaixin Xu, Yuchen Fan, Dongzhan Zhou, Xiaohong Liu, Ruimao Zhang, Philip Torr, Lei Bai, and Zhenfei Yin. Ego to world: Collaborative spatial reasoning in embodied systems via reinforcement learning, 2026. URL https://arxiv.org/abs/2603.14811.

[32] Chaoyang Wang, Wenrui Bao, Sicheng Gao, Bingxin Xu, Yu Tian, Yogesh S. Rawat, Yunhao Ge, and Yuzhang Shang. Vla-thinker: Boosting vision-language-action models through thinking-with-image reasoning, 2026. URL https://arxiv.org/abs/2603.14523.

[33] Yuelin Zhang, Sijie Cheng, Chen Li, Zongzhao Li, Yuxin Huang, Yang Liu, and Wenbing Huang. Recurrent reasoning with vision-language models for estimating long-horizon embodied task progress, 2026. URL https://arxiv.org/abs/2603.17312.

[34] Yi Zhang, Qiang Zhang, Xiaozhu Ju, Zhaoyang Liu, Jilei Mao, Jingkai Sun, Jintao Wu, Shixiong Gao, Shihan Cai, Zhiyuan Qin, Linkai Liang, Jiaxu Wang, Yiqun Duan, Jiahang Cao, Renjing Xu, and Jian Tang. Embodiedvsr: Dynamic scene graph-guided chain-of-thought reasoning for visual spatial tasks, 2025. URL https://arxiv.org/abs/2503.11089.

[35] Seokin Seo, Byung-Jun Lee, Jongmin Lee, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Mitigating covariate shift in behavioral cloning via robust stationary distribution correction. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 109177–109201. Curran Associates, Inc., 2024. doi: 10.52202/079017-3466.

[36] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL https://proceedings.neurips.cc/paper_files/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

[37] Taiye Chen, Xun Hu, Zihan Ding, and Chi Jin. Learning world models for interactive video generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FzfYoUp8F1.

[38] Meizhong Wang, Wanxin Jin, Kun Cao, Lihua Xie, and Yiguang Hong. Contactgaussian-wm: Learning physics-grounded world model from videos, 2026. URL https://arxiv.org/abs/2602.11021.

[39] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TFbwV6I0VLg.

[40] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1634–1644. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/raffin22a.html.

[41] Paul Pacaud, Ricardo Garcia, Shizhe Chen, and Cordelia Schmid. Scaling cross-environment failure reasoning data for vision-language robotic manipulation, 2026. URL https://arxiv.org/abs/2512.01946.

[42] Xiang Li, Ya-Li Li, Yuan Wang, Huaqiang Wang, and Shengjin Wang. Tcot: Trajectory chain-of-thoughts for robotic manipulation with failure recovery in vision-language-action model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(8):6486–6494, Mar. 2026. doi: 10.1609/aaai.v40i8.37577. URL https://ojs.aaai.org/index.php/AAAI/article/view/37577.

[43] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. THE COLOSSEUM: A Benchmark for Evaluating Generalization for Robotic Manipulation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.133.

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

[45] Meta. Llama-3.2-11B-Vision-Instruct, September 2024. URL https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct. Accessed: 2026-04-20.

[46] Google DeepMind. Gemini 3.1 Pro - Model Card, February 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/. Accessed: 2026-04-20.

[47] OpenAI. Introducing GPT-5.4, March 2026. URL https://openai.com/index/introducing-gpt-5-4/. Accessed: 2026-04-20.

[48] Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Shijie Zhou, Dilin Wang, Zhicheng Yan, Hongyu Xu, Justin Theiss, Tianlong Chen, Jiachen Li, Zhengzhong Tu, Zhangyang Wang, and Rakesh Ranjan. Vlm-3r: Vision-language models augmented with instruction-aligned 3d reconstruction. In *2026 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[49] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

[50] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition @ CoRL2023*, 2023. URL https://openreview.net/forum?id=rbLbPlKiOK.

[51] Philipp Scholl, Maged Iskandar, Sebastian Wolf, Jinoh Lee, Aras Bacho, Alexander Dietrich, Alin Albu-Schäffer, and Gitta Kutyniok. Learning-based adaption of robotic friction models. *Robotics and Computer-Integrated Manufacturing*, 89:102780, 2024. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2024.102780. URL https://www.sciencedirect.com/science/article/pii/S0736584524000668.

[52] Eric Rohmer, Surya P. N. Singh, and Marc Freese. V-rep: a versatile and scalable robot simulation framework. In *IROS*, pages 1321–1326, 2013.

[53] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, RISHI HAZRA, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollar, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=r35clVtGzw.

A Construction of the SFT Dataset

This section details the pipeline for collecting, annotating, and balancing the failure-augmented SFT dataset used to train the CoC framework.

A.1 Task Setup and Expert Demonstrations

The training dataset is built upon the COLOSSEUM simulation benchmark. We select seven diverse manipulation tasks: `meat_on_grill`, `move_hanger`, `open_drawer`, `put_money_in_safe`, `place_wine_at_rack_location`, `turn_oven_on`, and `setup_chess`. These tasks encompass a wide range of spatial precision requirements and varying task horizons. We utilize 100 expert demonstrations per task to yield the optimal execution trajectories.

A.2 Collection of Failure Rollouts

Training exclusively on expert demonstrations restricts the state-action distribution, hindering error identification and recovery from OOD states. To capture a broader distribution that includes erroneous states, we deploy the frozen base policy BridgeVLA [4] to autonomously execute the seven tasks. Due to compounding errors, the base policy generates various failed rollouts, including missed grasps, collisions, and stagnation. From 100 evaluation runs per task, we collect a total of 118 failure trajectories, which serve as the core candidates requiring correction in our failure-augmented dataset. Notably, as shown in Table 4, only a small fraction of frames within these failure trajectories exhibit actual execution errors. This stark imbalance underscores the necessity of our precise frame-level dual-check annotation, as opposed to naive trajectory-level failure labeling.

A.3 Unified CoT Annotation and Correction Synthesis

Both expert demonstrations and failure rollouts are processed through an algorithmic supervisor for unified annotation. The COLOSSEUM simulator's privileged API provides object poses as 3D positions and quaternions. From these poses, the supervisor extracts the 3D spatial coordinates of task-relevant kinematic chain anchors. This filtered data is used to construct the scene graph $\mathcal{G}_t$ and compute the progress scalar $\mathcal{P}_t$ according to the rules defined in Appendix B.2.

During the Chain-of-Correction annotation of each frame, the supervisor assumes the optimality of expert demonstrations and assigns `correction_needed = False`. Specifically, temporary progress fluctuations in expert data are treated as intentional maneuvers to bypass diverse obstacles [50]. Conversely, for failure rollouts, the supervisor strictly applies a dual-check mechanism to identify errors. For frames requiring correction, it selects a recovery strategy based on the identified error type. Accordingly, the corrective action a'_t is synthesized as follows:

- **Spatial Divergence Correction:** For errors identified via the forward check, the translational correction is computed as $\Delta\mathbf{p}_t = s \cdot (\mathbf{d}^t_{i\to j}/\|\mathbf{d}^t_{i\to j}\|_2)$. The end-effector position is then updated to $\mathbf{p}'_t = \mathbf{p}_t + \Delta\mathbf{p}_t$. Here, the term $\mathbf{d}^t_{i\to j}/\|\mathbf{d}^t_{i\to j}\|_2$ represents the unit direction vector along the focused edge, and $s = \min(s_{\max}, \alpha \cdot D^t_{i\to j})$ is a dynamic step size to prevent spatial overshooting.
- **Physical Stasis Recovery:** For physical stasis detected via the backward check, directly pulling toward the target often fails due to unmodeled collisions or environmental friction [16, 51]. To break this deadlock, the supervisor executes a compliance lift: it reverses the approach vector, injects lateral noise $\mathcal{N}(0, \sigma^2)$ [40], and enforces a strict upward movement along the Z-axis. This lift clears horizontal contact constraints, freeing the arm from stagnation before re-attempting the approach, thereby providing the VLM with a robust recovery template.

Action Parameterization. To ensure compatibility with the base policy, the synthesized corrective action $a'_t \in \mathbb{R}^8$ is represented as a concatenated vector comprising: a 3D translation vector (x, y, z), a 4D rotation vector represented as a quaternion, and a 1D binary gripper state (1 for open, 0 for closed). During the heuristic correction synthesis, only the 3D translation component is mathematically manipulated by the supervisor, while the remaining dimensions are frozen to maintain stable kinematic transitions.

Table 4: **Detailed frame-level statistics of the SFT dataset before and after balancing.** `Corr=T/F` indicates the binary `correction_needed` label. The composition percentages show the data origins: **Expert (%)** denotes frames from expert demonstrations (which exclusively contain `Corr=F` frames), while **Rollout(F) %** and **Rollout(T) %** represent valid and erroneous frames extracted from the base policy's failure trajectories, respectively. The balancing process dynamically oversamples the minority `Corr=T` frames to reach an aggregate target ratio of approximately 20%.

Task Name	Raw Dataset						After Balancing					
	Total	Corr=T	Corr=F	Expert (%)	Rollout(F) %	Rollout(T) %	Total	Corr=T	Corr=F	Expert (%)	Rollout(F) %	Rollout(T) %
meat_on_grill	13194	1	13193	99.45%	0.54%	0.01%	13209	16	13193	99.34%	0.54%	0.12%
move_hanger	17313	453	16860	96.38%	1.00%	2.62%	24108	7248	16860	69.21%	0.72%	30.06%
open_drawer	10701	2	10699	99.96%	0.02%	0.02%	10731	32	10699	99.68%	0.02%	0.30%
place_wine_at_rack_location	19016	35	18981	99.24%	0.57%	0.18%	19541	560	18981	96.58%	0.56%	2.87%
put_money_in_safe	17965	128	17837	99.06%	0.22%	0.71%	19885	2048	17837	89.50%	0.20%	10.30%
setup_chess	25581	1163	24418	91.68%	3.77%	4.55%	43026	18608	24418	54.51%	2.24%	43.25%
turn_oven_on	13375	42	13333	99.28%	0.40%	0.31%	14005	672	13333	94.82%	0.39%	4.80%
Grand Total	**117145**	**1824**	**115321**	**97.23%**	**1.21%**	**1.56%**	**144505**	**29184**	**115321**	**78.83%**	**0.98%**	**20.20%**

A.4 Dataset Balancing and Composition

As shown in Table 4, valid execution frames heavily outnumber erroneous ones. Since training on the raw data distribution would bias the VLM to trivially predict `correction_needed = False` by exploiting a statistical shortcut to minimize loss [41], we partition the annotated dataset into a valid pool and an erroneous pool. We empirically set a target ratio of $r_{\text{target}} = 0.2$ for the erroneous frames within the training mixture. This oversampling ratio provides a stable trade-off between error recovery robustness and execution fluency.

A.5 Supplementary Visualizations

To offer an intuitive visualization of the unified annotation pipeline, we provide a demonstration video showcasing the SFT dataset construction process. This video `sft_data_visualization.mp4` is included in the supplementary materials. It highlights the frame-by-frame evolution of the extracted anchor-grounded scene graph and the continuous progress scalar $\mathcal{P}_t$.

B Implementation Details of CoC and Baselines

This section details the SFT protocols, architectures, prompt designs, and hyperparameters for CoC and all evaluated baselines. To ensure reproducibility, these configurations are kept consistent across simulated and real-world experiments unless otherwise specified.

B.1 SFT Training Details

The CoC framework is optimized using the unified failure-augmented dataset described in Appendix A. To ensure maximum capacity for multi-modal reasoning and structural relationship extraction, we perform **full-parameter fine-tuning** on the base VLM. To manage the substantial memory footprint of updating a 7B model with a sequence length of 4096, we utilize gradient checkpointing, FlashAttention-2, and the 8-bit AdamW optimizer. The hyperparameters for this phase, empirically tuned to prevent catastrophic forgetting while ensuring robust error-recovery behavior, are summarized in Table 5.

Specifically, by employing a per-device batch size of 1 with 32 gradient accumulation steps across four NVIDIA A800 (80GB) GPUs, we maintain an effective global batch size of 128. The total training process takes approximately 26.4 hours to reach convergence.

B.2 Implementation Details of the CoC Framework

The CoC framework is deployed as an asynchronous runtime server using a Flask-based API, ensuring minimal interference with the high-frequency base policy control loop.

Algorithmic Annotation Rules: Kinematic Thresholds and Progress Estimation. To explicitly implement contact verification, the `HOLDING` state constraint is activated only if the `TOUCHING` spatial relation exists and the inter-frame relative displacement between the end-effector v_{gripper} and the

Table 5: **Hyperparameters for full-parameter SFT.**

Hyperparameter	Value
Optimization Configurations	
Base Model	MomaGraph-R1 (7B)
Training Strategy	Full-Parameter Fine-Tuning
Optimizer	8-bit AdamW
Peak Learning Rate	2×10^{-5}
Learning Rate Scheduler	Cosine Annealing
Minimum Learning Rate	2×10^{-6}
Warmup Steps	1000
Weight Decay	0.01
Max Gradient Norm	1.0
Training Epochs	4
Batching & Sequence Size	
Max Sequence Length	4096
Per-Device Batch Size	1
Gradient Accumulation Steps	32
Number of GPUs	4
Global Batch Size	128

manipulation object v_{MO} remains strictly below a predefined tolerance threshold τ:

$$\|\mathbf{d}^t_{\text{gripper}\to\text{MO}} - \mathbf{d}^{t-1}_{\text{gripper}\to\text{MO}}\|_2 < \tau.$$

In our experiments, this kinematic tolerance τ is empirically set to 2 mm.

To quantify global task progress, the algorithmic supervisor maps the spatial relations and geometric distances from $\mathcal{G}_t$ into a continuous scalar $\mathcal{P}_t \in [0, 1]$. To ensure generalizability across diverse manipulation tasks, this scalar is sequentially evaluated through three universal phases. In the initial reaching phase, $\mathcal{P}_t$ increases from 0.0 to 0.35 as the distance $D^t_{\text{gripper}\to\text{MO}}$ decreases, reaching 0.35 upon `TOUCHING`, and stepping to 0.5 upon `HOLDING` verification. During the subsequent transport phase, $\mathcal{P}_t$ interpolates from 0.5 to 0.85 as $D^t_{\text{MO}\to\text{target}}$ decreases. Once the object nears the target zone, an `AROUND` relation is triggered, initiating the final alignment phase where $\mathcal{P}_t$ scales up to 1.0 based on specific spatial alignment criteria (*e.g.*, $\mathcal{P}_t = 0.852$ for the hanger closely approaching the target rack in Fig. 2). For multi-stage tasks like `setup_chess`, the supervisor employs a hierarchical mapping: it sets a baseline of k/N for k completed stages out of N total stages, and adds the current stage's local progression scaled by $1/N$.

Model Architecture and Compute Resources. To instantiate the CoC evaluator, we adopt **MomaGraph-R1** [30] as the base model prior to our SFT pipeline. Initialized from the foundational `Qwen2.5-VL-7B-Instruct` [44], this architecture was originally optimized via DAPO reinforcement learning on the *MomaGraph-Scenes* dataset [30]. Consequently, it excels at extracting compact, task-relevant spatial and functional relationships from embodied scenes, making it an ideal initialization for our framework. For efficient deployment, the fully fine-tuned model is loaded in `bfloat16` precision with FlashAttention-2 enabled. The asynchronous runtime server operates efficiently on a single NVIDIA A800 GPU. Finally, a comprehensive analysis of the system's inference latency is provided in Appendix C.

Sliding Window Memory. To prevent context overflow and excessive inference latency, CoC utilizes a memory window of exactly one historical step. The system extracts the scene graph and global task progress from the preceding output, which are formatted as a lightweight JSON block alongside the executed action. This structured history is then concatenated with the new RGB observations to form the current prompt.

Prompt Engineering and Output Formatting. At each execution step, the system prompt explicitly instructs the VLM to act as a progress-aware evaluator. To enforce a standardized response, the VLM is strictly supervised to autoregressively output a JSON dictionary comprising four keys:

scene_graph, progress, correction_needed, and the refined corrected_action. The exact prompt templates driving this structured bidirectional evaluation across initial and subsequent stages are detailed below:

System Prompt Templates for CoC Evaluator

```
=== For Initial Step (t = 0) ===
Task instruction: {instruction}
[Time Step: 0] Task initiation. Analyze the multi-view images to extract
the initial scene graph. Evaluate the initial task progress, and predict
the resulting spatial changes if the Action Proposal {base_proposal_list}
is applied.

=== For Subsequent Steps (t > 0) ===
[Time Step: {time_step}] Previous action executed. Given the new
observations and the Action Proposal: {base_proposal_list}, perform a
structured physical dual-check:
1. Backward check: Compare History vs. Current to quantify geometric
   distance changes and identify any physical stasis or unintended
   deviations.
2. Forward check: Predict how the Current Scene Graph (specifically edge
   distances and spatial relations) and task progress will change if the
   Proposal is executed.
Synthesize these checks to objectively determine if a correction is needed.
Output the structured evaluation and the corrected action.
```

B.3 Implementation Details of Baselines

To ensure a rigorous and fair comparison, all inference-time steering baselines are implemented faithfully following their official open-source repositories. Below, we detail the training and deployment pipelines of these methods, as well as their specific implementations.

Task-Specific Training. A critical implementation detail of the evaluated baselines is their dependency on task-specific predictive models. As specified in their methodologies, LPB [11], FOREWARN [12], and GPC [13] all require training distinct world models for each manipulation environment. Furthermore, achieving accurate predictions demands extensive auxiliary data beyond the expert demonstrations. Specifically, GPC utilizes random exploration rollouts to learn video dynamics [13], while LPB and FOREWARN require collecting suboptimal policy rollouts for their respective latent dynamics training [11, 12]. To handle this, we split the failure-augmented training set of CoC into seven task-specific datasets, each of which is utilized to train a specialized, independent world model for these baselines. This task-specific training phase is executed on a computing node equipped with four NVIDIA A800 (80GB) GPUs. All task-specific retraining or fine-tuning strictly adheres to the guidance of the corresponding official repositories.

Inference-Time Steering Pipeline. After the task-specific training, these steering models are deployed on dedicated asynchronous servers, each operating on a single NVIDIA A800 GPU, to avoid bottlenecking the base policy control loop. Unlike CoC, which synthesizes a single corrective action through autoregressive reasoning, the evaluated baselines require evaluating a diverse set of future possibilities.

To accommodate this algorithmic requirement, we meticulously adapt the base policy's decoding strategy for each baseline to maximize their respective performance. Specifically, for GPC and FOREWARN, we enable temperature-scaled multinomial sampling to generate a diverse set of action proposals. For GPC, we sample $N = 8$ candidates at a temperature of $T = 0.5$. For FOREWARN, we sample $N = 8$ candidates at a more deterministic temperature of $T = 0.1$. For LPB, strictly adhering to its algorithmic design, we expand the single deterministic policy output into $N = 64$ candidates, infused with its native dynamics noise scale of $\sigma = 0.05$. These candidate actions are then unrolled through each baseline's respective world model to simulate future states.

Following the state simulation, the diffusion world model of GPC processes visual observations dynamically resized to 96×96 resolution, with action sequences padded to a fixed horizon of $H = 4$.

Table 6: **Quantitative latency analysis on the long-horizon `setup_chess` task.** Results are averaged over 3 independent runs on a single NVIDIA A800 (80GB) GPU. Note that *Total Latency* = *Base Policy* + *Steering*.

Method	Base Policy (ms)	Steering/Rank (ms)	Total Latency (ms)	Control Freq. (Hz)
BridgeVLA (w/o Steering)	156.9 $\pm$0.9	–	156.9 $\pm$0.9	6.37
LPB [11]	338.3 $\pm$0.7	**30.6** $\pm$0.5	368.9 $\pm$0.2	2.71
GPC [13]	193.6 $\pm$1.7	260.8 $\pm$4.2	454.4 $\pm$5.9	2.20
CoC-Normal (Ours)	**156.9** $\pm$0.9	12467.1 $\pm$96.7	12624.0 $\pm$95.9	0.08
CoC-Fast (Ours)	160.9 $\pm$4.1	157.0 $\pm$4.9	**317.8** $\pm$8.8	**3.15**

The predicted states are then scored by the zero-shot `Qwen2.5-VL-7B-Instruct` model utilizing a generation temperature of $\tau = 0.2$. For FOREWARN, visual observations are dynamically padded to bypass sequence dimension mismatch. It employs a Dreamer-v3 style latent world model, followed by a `Llama-3.2-11B-Vision-Instruct` module that generates text narrations for ranking. To guarantee fairness, this vision-language model is task-finetuned via LoRA directly on CoC's failure-augmented dataset. To maintain stable inference, its generation configuration is strictly bounded, limiting `imagined_steps` to 1 while dynamically synchronizing `num_history_images` to the target sequence length. Finally, for LPB, visual observations are resized to 224×224 resolution to match its latent dynamics model. The optimal trajectory is then selected by computing the nearest-neighbor distance to the expert latent manifold.

C Inference Latency Analysis

While the benchmark evaluations in the main text report the performance of *CoC-Normal* with full reasoning traces, real-world robotic deployments often impose strict real-time constraints. To evaluate the real-time feasibility and computational efficiency of our framework, we introduce *CoC-Fast* and compare CoC (Normal and Fast modes) against GPC, LPB, and the non-steered base policy. We select the long-horizon `setup_chess` task for latency analysis, as it requires dozens of continuous corrective steps, rigorously exposing the steady-state real-time capabilities of all evaluated frameworks.

Implementation of CoC-Fast. This mode optimizes inference latency via the prompt template below. Instead of forcing full CoT generation at every control step, it outputs a single word `'PASS'` if the base proposal is safe, or provides a direct correction for $t > 0$.

System Prompt Templates for CoC-Fast Evaluator

```
=== For Subsequent Steps (t > 0) ===
Task: {instruction}. Proposal: {base_proposal_list}.
Decision Task: Is this proposal safe and correct?
CRITICAL INSTRUCTION: If safe, output the exact word 'PASS' and NOTHING ELSE.
   If dangerous, output 'CORRECT' and provide the corrected JSON.
```

Quantitative Results and Analysis. The latency metrics in Table 6 reveal several critical insights into the system's runtime behavior: (1) **Dominance of Token Generation:** In *CoC-Normal* mode, the total pipeline latency reaches 12.6 seconds per step. The autoregressive text generation is the primary bottleneck, accounting for over 98% of the total latency ($\approx$12.5s), while the ViT pre-filling and base policy execution consume minimal time. (2) **Efficiency of Strategic Prompting:** By transitioning to *CoC-Fast* mode, the VLM steering latency drops drastically to 157.0 ms per step, achieving an $80\times$ speedup over the Normal mode. In this configuration, shortening the generated output substantially reduces inference latency. (3) **Base Policy Bottlenecks in Baselines:** The results reveal a hidden computational overhead in sample-and-rank paradigms. To obtain more steering success, LPB requires expanding the action proposals to $N = 64$, which inflates the base policy sampling latency to 338.3 ms. Similarly, GPC requires 8 action candidates, increasing this sampling latency to 193.6 ms. In contrast, CoC relies on a single action proposal, keeping the base policy latency lightweight (156.9 ms). (4) **Overall System Superiority:** *CoC-Fast* achieves the highest

Table 7: **Success-rate comparison of CoC-Normal and CoC-Fast.** Success rates (%) use the seven-task COLOSSEUM protocol and aggregation of Table 2. Condition-wise entries report the mean and standard deviation across applicable tasks. "Avg. Success" averages the 14 perturbed settings, excluding "No Perturbations". Boldface denotes better performance.

Model	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Texture	RO Texture	MO Size
CoC-Normal	71.6	71.8 ±31.1	67.6 ±37.1	**70.9** ±36.5	61.1 ±34.0	46.7 ±36.0	88.0 ±0.0	78.5 ±29.8
CoC-Fast	**73.0**	**73.1** ±37.6	**74.9** ±35.0	**70.9** ±39.9	**63.2** ±43.9	**52.0** ±40.0	**94.0** ±2.0	**80.0** ±31.3
Model	Light Color	RLBench Variations	Background Texture	Table Texture	Camera Pose	Distractor Objects	Table Color	RO Size
CoC-Normal	**68.4** ±32.9	**72.6** ±34.0	70.3 ±30.0	68.4 ±27.9	69.3 ±37.9	**71.2** ±27.6	**77.1** ±25.8	**92.0** ±0.0
CoC-Fast	66.9 ±38.9	69.1 ±40.3	**70.9** ±40.3	**70.9** ±42.4	**76.6** ±30.0	69.7 ±40.6	70.9 ±41.1	**92.0** ±0.0

control frequency (3.15 Hz) among all evaluated steering methods. While LPB features an extremely fast latent steering module (30.6 ms), its heavy action sampling drags the total frequency down to 2.71 Hz. GPC is further bottlenecked by both action sampling and diffusion rollouts (2.20 Hz). Overall, *CoC-Fast* provides the optimal balance between logical correction and real-time execution.

Success-Latency Trade-off. To examine the effect of shorter outputs on correction performance, we compare CoC-Fast with CoC-Normal using the same checkpoint trained with structured reasoning targets. Table 7 shows that CoC-Fast achieves an average success rate of 73.0%, compared with 71.6% for CoC-Normal, with gains and losses varying across perturbations. Together with the latency reduction in Table 6, these results show that shorter outputs preserve overall correction performance on this benchmark while substantially reducing inference time.

D Offline Evaluation of Predictive Reasoning

We evaluate *Error Detection*, *Progress Estimation*, and *Action Precision* using an offline protocol with reference targets generated by the same structured annotation pipeline used for SFT. These scores quantify agreement with task-relevant correction targets, while closed-loop task success remains our primary measure of practical steering effectiveness.

D.1 Experimental Setup and Protocol

Evaluated Models. CoC is compared against a diverse set of state-of-the-art reasoning models, including four zero-shot general VLMs (Qwen2.5-VL [44], Llama-3.2-Vision [45], Gemini 3.1 Pro [46], GPT-5.4 [47]), two domain-specific embodied VLMs (MomaGraph-R1 [30], VLM-3R [48]), and three inference-time policy steering methods (FOREWARN [12], LPB [11], and GPC [13]).

Dataset Construction. To objectively assess the false positive bias of various steering methods and VLMs, we construct a perfectly balanced validation set containing real compounding errors. We pool the failure-augmented training set and randomly sample exactly 100 positive instances that require correction and 100 negative instances that require no intervention. This 200-sample benchmark ensures that classification metrics are not skewed by class imbalance.

Offline Simulation of Propose-and-Rank Baselines. Unlike CoC, which autonomously synthesizes corrective actions, inference-time steering baselines require a diverse pool of candidates to score. To ensure an equitable offline comparison, we dynamically generate a candidate pool of 8 action proposals for each instance. For negative instances, candidate 0 is the ground-truth action, while the other candidates receive minor Gaussian noise. For positive instances, candidate 0 is injected with severe destructive noise to simulate a failing proposal, and candidate 1 is set to the perfect expert recovery action. An ideal evaluator should accurately distinguish the ground-truth or expert recovery action from noisy alternatives.

Metrics. We evaluate *Error Detection* via F1-Score, Precision, and Recall. For *Progress Estimation*, we report the RMSE and MAE between the predicted scalar and the ground-truth task progress

recorded in the failure-augmented dataset. For *Action Precision*, we compute the Euclidean distance (L_2) for spatial kinematics (translation and rotation) within the normalized continuous action space $[-1, 1]$, alongside binary error rates for gripper states.

D.2 Detailed Quantitative Analysis of Table 1

Error Detection: Selectivity over Sensitivity. As shown in Table 1, CoC achieves the highest F1-score of 0.851, with precision of 1.000 and recall of 0.740. The steering baselines do not output progress estimates and exhibit a false-positive bias. FOREWARN and GPC achieve recall of 1.000 with precision of 0.500, frequently flagging valid proposals for correction. CoC's more selective decisions are consistent with the value of combining geometric context and task progress to distinguish correction-required proposals from benign deviations.

Progress Estimation and Action Precision. CoC exhibits superior performance in Progress Estimation, achieving an RMSE of 0.037 and an MAE of 0.007. MomaGraph-R1 also uses scene graphs, but its F1-score of 0.085 shows that a structured state representation does not by itself ensure accurate correction decisions on this benchmark. CoC couples geometric states with progress-aware evaluation of action proposals, linking the representation to the correction objective. In terms of Action Precision, zero-shot general VLMs exhibit gripper-state error rates above 20%, indicating persistent errors in selecting the reference open or close command. These errors highlight the need to evaluate discrete gripper control alongside continuous pose prediction. Furthermore, while spatially aware models like VLM-3R possess strong 3D understanding, they output implicit feature representations. Without the strict kinematic correspondence enforced by CoC's anchor-grounded structures, VLM-3R falls significantly behind CoC in fine-grained spatial accuracy. CoC achieves state-of-the-art Translation L_2 error of 0.020. The slightly higher Rotation L_2 error compared to FOREWARN suggests that while CoC excels at determining *where* to move (translation) and *whether* to intervene (detection), fine-grained angular corrections remain a challenging task for discrete token-based LLM architectures, leaving room for future refinement.

D.3 Qualitative Case Studies

Figures 5 and 6 illustrate the two error types. In Scenario A, CoC preserves a valid action that resolves stagnation, whereas the illustrated baselines request unnecessary corrections. In Scenario B, CoC detects a misaligned proposal and generates a correction that avoids the illustrated collision.

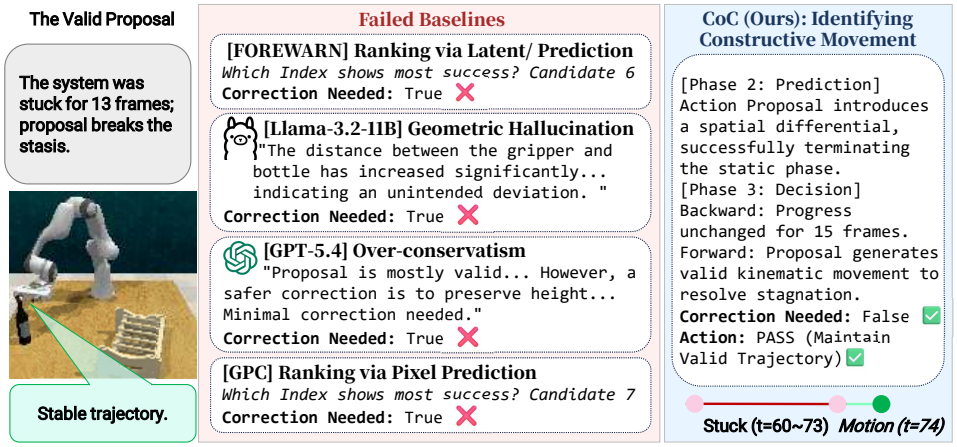


Figure 5: **Scenario A: false-positive corrections.** A valid proposal breaks a 13-frame stasis. FOREWARN, GPC, Llama-3.2, and GPT-5.4 incorrectly request correction, while CoC recognizes the constructive motion and returns `PASS`.

D.4 Prompt Templates for Evaluated Baselines

To ensure a fair and standardized offline evaluation, we enforce strict output formatting for all zero-shot general VLMs and domain-specific embodied VLMs.

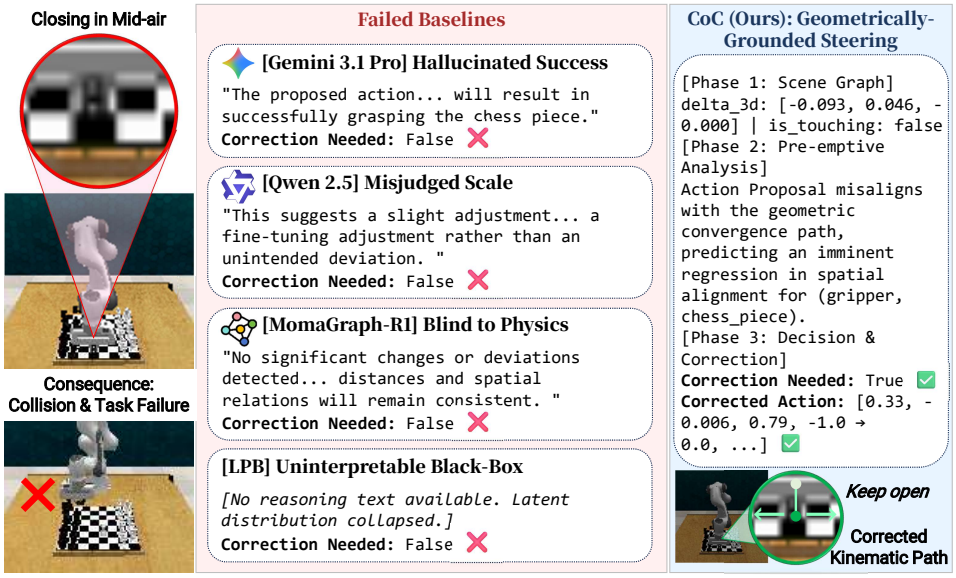


Figure 6: **Scenario B: missed corrections.** The proposal closes the gripper before correct alignment. The illustrated baselines accept it, while CoC detects the geometric mismatch and generates a corrective action.

During evaluation, these baseline models receive the standard textual state descriptions and multi-view images extracted directly from the evaluation dataset, both of which are appended with a task-specific structural instruction. This structural prompt explicitly guides the models to generate a CoT reasoning process followed by a parseable JSON block. This guarantees that their outputs can be systematically processed by our automated metrics engine without being penalized due to unstructured responses.

Zero-Shot Prompt for Error Detection and Correction (All Baselines)

```
Task instruction: {instruction}
[Time Step: {t}] {Observation text and multi-view images from dataset}
Action Proposal: {base_proposal_list}

Follow the format: [Thinking Process] then final JSON block. End your
response with a JSON block containing 'correction_needed' (bool) and
'corrected_action' (list of 8 floats).
```

Zero-Shot Prompt for Progress Estimation (All Baselines)

```
Task instruction: {instruction}
[Time Step: {t}] {Observation text and multi-view images from dataset}

Follow the format: [Thinking Process] then final JSON block. Based on the
visual observation, estimate the completion progress of the task. End your
response with a JSON block containing 'global_progress' (a float between
0.0 and 1.0).
```

E Details of the COLOSSEUM Evaluation

To comprehensively evaluate closed-loop correction capabilities and OOD robustness, we conduct extensive evaluations on the COLOSSEUM benchmark [43].

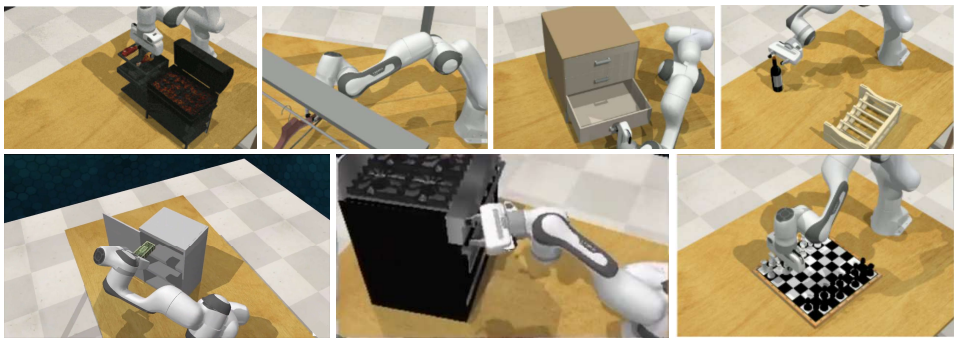


Figure 7: **Visualization of 7 COLOSSEUM tasks.**

E.1 Simulation Environments and Robot Setup

The simulation is powered by CoppeliaSim [52]. The agent controls a Franka Panda robot arm equipped with a parallel-jaw gripper. To capture visual observations, the environment is configured with four 128×128 RGB-D cameras. These cameras provide a comprehensive multi-view perspective, positioned strategically at the front, left shoulder, right shoulder, and the robot's wrist.

E.2 Task Definitions and Perturbation Settings

Evaluated Tasks. We select 7 representative long-horizon and precision-demanding manipulation tasks from the suite: *Setup Chess*, *Move Hanger*, *Place Wine at Rack Location*, *Put Money in Safe*, *Meat on Grill*, *Turn Oven On*, and *Open Drawer*. Visual snapshots of these tasks are provided in Figure 7.

14 OOD Perturbation Settings. We evaluate 14 OOD perturbation settings on COLOSSEUM, comprising 13 individual settings and one joint setting. These variations span physical and visual shifts, including changes in light color, manipulation object (MO) color, receiver object (RO) color, MO texture, RO texture, table color, table texture, background texture, MO size, and RO size. Furthermore, it incorporates camera pose shifts, RLBench variations, and introduces dynamic challenges such as the injection of random distractor objects. Note that the *All Perturbations* configuration applies the perturbations jointly as a separate evaluation condition. Its result is not an average over the individual settings, and it need not be the hardest condition for every method.

E.3 Baselines and Execution Details

Base Policy. We employ BridgeVLA [4] as the default uncorrected base policy to generate the initial action proposals for all steering frameworks. Specifically, the identical COLOSSEUM-finetuned model provided by the authors is deployed across all experiments to ensure controlled comparisons.

Inference-Time Steering Baselines. We compare CoC with state-of-the-art steering methods including FOREWARN [12], LPB [11], and GPC [13]. The training and inference implementations of these baselines are specified in Appendix B.3.

Execution Protocol. We evaluate each applicable task-condition pair in three independent runs, with 25 episodes per run. Results are reported for both the unperturbed condition (*No Perturbations*) and the generalization settings in Tables 8 through 14.

E.4 Complete Per-Task Results

Tables 8 through 14 provide the exhaustive success rates corresponding to the aggregated results presented in Table 2 of the main text. They detail the granular performance of the base policy, all baseline steering methods, and our proposed CoC across all 7 manipulation tasks under their applicable individual and joint perturbation settings.

Table 8: **Comparison with SOTA methods on the `meat_on_grill` task.** The "Avg. Success" column reports the average success rate (%) across 11 generalization settings.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Size	RO Size
BridgeVLA (Base)	NeurIPS (2025)	83.9	94.7 ±1.9	17.3 ±1.9	32.0 ±0.0	93.3 ±1.9	96.0 ±0.0	–
FOREWARN	RSS (2025)	67.5	97.3 ±1.9	**100.0** ±0.0	**97.3** ±1.9	**96.0** ±3.3	86.7 ±5.0	–
LPB	NeurIPS (2025)	84.6	**100.0** ±0.0	21.3 ±1.9	28.0 ±3.3	94.7 ±1.9	**100.0** ±0.0	–
GPC	RA-L (2026)	83.8	97.3 ±1.9	20.0 ±0.0	32.0 ±0.0	89.3 ±1.9	98.7 ±1.9	–
CoC w/o Geometry	–	66.4	94.7 ±1.9	**100.0** ±0.0	92.0 ±0.0	94.7 ±1.9	88.0 ±9.8	–
CoC w/o History	–	67.0	**100.0** ±0.0	**100.0** ±0.0	94.7 ±3.8	90.7 ±5.0	88.0 ±3.3	–
CoC w/o Prediction	–	33.1	40.0 ±3.3	40.0 ±16.3	40.0 ±0.0	40.0 ±0.0	36.0 ±6.5	–
CoC w/o Progress	–	68.7	92.0 ±0.0	92.0 ±3.3	92.0 ±3.3	92.0 ±3.3	76.0 ±0.0	–
CoC	–	**91.3**	96.0 ±0.0	88.0 ±9.8	96.0 ±0.0	88.0 ±0.0	88.0 ±9.8	–

Models	Reference	Light Color	Table Color	Table Texture	Background Texture	Distractor Objects	Camera Pose	RLBench Variations
BridgeVLA (Base)	NeurIPS (2025)	97.3 ±1.9	92.0 ±0.0	**97.3** ±1.9	**97.3** ±1.9	**100.0** ±0.0	**100.0** ±0.0	**100.0** ±0.0
FOREWARN	RSS (2025)	64.0 ±3.3	56.0 ±0.0	42.7 ±1.9	36.0 ±0.0	66.7 ±1.9	44.0 ±0.0	53.3 ±1.9
LPB	NeurIPS (2025)	98.7 ±1.9	**96.0** ±0.0	96.0 ±3.3	**97.3** ±1.9	**100.0** ±0.0	**100.0** ±0.0	98.7 ±1.9
GPC	RA-L (2026)	**100.0** ±0.0	92.0 ±0.0	96.0 ±3.3	96.0 ±3.3	**100.0** ±0.0	97.3 ±1.9	**100.0** ±0.0
CoC w/o Geometry	–	64.0 ±0.0	60.0 ±6.5	40.0 ±3.3	36.0 ±3.3	60.0 ±0.0	44.0 ±0.0	52.0 ±3.3
CoC w/o History	–	60.0 ±0.0	60.0 ±3.3	44.0 ±3.3	40.0 ±0.0	64.0 ±0.0	44.0 ±0.0	52.0 ±3.3
CoC w/o Prediction	–	36.0 ±9.8	28.0 ±9.8	12.0 ±3.3	20.0 ±6.5	24.0 ±3.3	40.0 ±0.0	48.0 ±0.0
CoC w/o Progress	–	64.0 ±3.3	72.0 ±6.5	56.0 ±9.8	40.0 ±6.5	68.0 ±3.3	48.0 ±3.3	56.0 ±6.5
CoC	–	**100.0** ±0.0	92.0 ±3.3	92.0 ±0.0	72.0 ±3.3	88.0 ±6.5	**100.0** ±0.0	**100.0** ±0.0

Table 9: **Comparison with SOTA methods on the `move_hanger` task.** The "Avg. Success" column reports the average success rate (%) across 10 generalization settings.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Size	RO Size
BridgeVLA (Base)	NeurIPS (2025)	37.9	40.0 ±0.0	1.3 ±1.9	24.0 ±3.3	40.0 ±0.0	–	–
FOREWARN	RSS (2025)	1.1	22.7 ±1.9	2.7 ±1.9	4.0 ±0.0	0.0 ±0.0	–	–
LPB	NeurIPS (2025)	38.8	**44.0** ±13.1	0.0 ±0.0	24.0 ±17.0	**48.0** ±0.0	–	–
GPC	RA-L (2026)	37.5	41.3 ±1.9	2.7 ±1.9	**28.0** ±16.3	38.7 ±8.2	–	–
CoC w/o Geometry	–	1.1	16.0 ±0.0	2.7 ±3.8	4.0 ±3.3	0.0 ±0.0	–	–
CoC w/o History	–	1.7	20.0 ±0.0	**8.0** ±3.3	1.3 ±1.9	0.0 ±0.0	–	–
CoC w/o Prediction	–	0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	–	–
CoC w/o Progress	–	1.2	24.0 ±6.5	**8.0** ±0.0	0.0 ±0.0	4.0 ±3.3	–	–
CoC	–	**39.2**	40.0 ±0.0	4.0 ±3.3	24.0 ±6.5	40.0 ±3.3	–	–

Models	Reference	Light Color	Table Color	Table Texture	Background Texture	Distractor Objects	Camera Pose	RLBench Variations
BridgeVLA (Base)	NeurIPS (2025)	**53.3** ±5.0	74.7 ±1.9	**48.0** ±3.3	40.0 ±0.0	37.3 ±3.8	**20.0** ±0.0	40.0 ±3.3
FOREWARN	RSS (2025)	4.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
LPB	NeurIPS (2025)	52.0 ±3.3	72.0 ±5.7	42.7 ±9.4	**44.0** ±0.0	45.3 ±10.5	16.0 ±0.0	**44.0** ±0.0
GPC	RA-L (2026)	50.7 ±3.8	73.3 ±7.5	45.3 ±3.8	34.7 ±7.0	52.0 ±0.0	12.0 ±0.0	37.3 ±1.9
CoC w/o Geometry	–	4.0 ±3.3	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
CoC w/o History	–	0.0 ±0.0	4.0 ±3.3	4.0 ±3.3	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
CoC w/o Prediction	–	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
CoC w/o Progress	–	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
CoC	–	52.0 ±3.3	**76.0** ±9.8	**48.0** ±3.3	40.0 ±0.0	**56.0** ±6.5	12.0 ±6.5	40.0 ±3.3

F Details of the Ablation Study

F.1 Inference-time Prompt Ablation Protocol

To assess the trained model's reliance on each reasoning module, we employ **inference-time prompt ablation**. To avoid introducing confounding optimization variables through retraining, we utilize identical Full CoC weights for all variants. We dynamically mask specific input features or bypass reasoning steps via precise prompt interventions during evaluation, measuring how these changes affect the trained model's performance. Retraining ablations with diverse supervision targets are reported separately in Section F.4.

Table 10: **Comparison with SOTA methods on the `open_drawer` task.** The "Avg. Success" column reports the average success rate (%) across 10 generalization settings.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Size	RO Size
BridgeVLA (Base)	NeurIPS (2025)	89.9	**96.0** ±0.0	50.7 ±3.8	97.3 ±1.9	–	**92.0** ±0.0	–
FOREWARN	RSS (2025)	42.1	**96.0** ±0.0	92.0 ±3.3	96.0 ±0.0	–	64.0 ±0.0	–
LPB	NeurIPS (2025)	86.8	85.3 ±3.8	54.7 ±1.9	93.3 ±1.9	–	89.3 ±1.9	–
GPC	RA-L (2026)	90.8	94.7 ±3.8	68.0 ±5.7	96.0 ±0.0	–	90.7 ±1.9	–
CoC w/o Geometry	–	43.1	94.7 ±1.9	92.0 ±3.3	86.7 ±5.0	–	76.0 ±3.3	–
CoC w/o History	–	41.9	92.0 ±3.3	94.7 ±1.9	92.0 ±3.3	–	72.0 ±9.8	–
CoC w/o Prediction	–	34.0	76.0 ±0.0	80.0 ±6.5	76.0 ±6.5	–	56.0 ±6.5	–
CoC w/o Progress	–	44.4	89.3 ±1.9	**96.0** ±3.3	96.0 ±0.0	–	72.0 ±6.5	–
CoC	–	**92.7**	86.7 ±1.9	92.0 ±3.3	**98.7** ±1.9	–	**92.0** ±0.0	–

Models	Reference	Light Color	Table Color	Table Texture	Background Texture	Distractor Objects	Camera Pose	RLBench Variations
BridgeVLA (Base)	NeurIPS (2025)	86.7 ±3.8	**93.3** ±1.9	**100.0** ±0.0	**100.0** ±0.0	**92.0** ±5.7	**92.0** ±6.5	94.7 ±1.9
FOREWARN	RSS (2025)	45.3 ±1.9	34.7 ±3.8	28.0 ±6.5	17.3 ±1.9	28.0 ±0.0	4.0 ±0.0	12.0 ±0.0
LPB	NeurIPS (2025)	80.0 ±3.3	90.7 ±1.9	98.7 ±1.9	96.0 ±3.3	82.7 ±5.0	**92.0** ±3.3	90.7 ±1.9
GPC	RA-L (2026)	**88.0** ±3.3	92.0 ±0.0	**100.0** ±0.0	**100.0** ±0.0	89.3 ±1.9	**92.0** ±3.3	92.0 ±0.0
CoC w/o Geometry	–	40.0 ±0.0	40.0 ±6.5	32.0 ±3.3	20.0 ±3.3	24.0 ±6.5	4.0 ±0.0	16.0 ±3.3
CoC w/o History	–	32.0 ±0.0	40.0 ±13.1	24.0 ±3.3	16.0 ±0.0	24.0 ±6.5	8.0 ±3.3	16.0 ±6.5
CoC w/o Prediction	–	28.0 ±3.3	20.0 ±3.3	32.0 ±0.0	20.0 ±9.8	16.0 ±3.3	4.0 ±0.0	8.0 ±0.0
CoC w/o Progress	–	52.0 ±6.5	36.0 ±6.5	28.0 ±6.5	20.0 ±6.5	24.0 ±6.5	4.0 ±3.3	16.0 ±3.3
CoC	–	**88.0** ±3.3	92.0 ±3.3	92.0 ±0.0	96.0 ±0.0	**92.0** ±3.3	88.0 ±6.5	**96.0** ±0.0

Table 11: **Comparison with SOTA methods on the `put_money_in_safe` task.** The "Avg. Success" column reports the average success rate (%) across 13 perturbations.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Texture	RO Texture	MO Size
BridgeVLA (Base)	NeurIPS (2025)	72.8	89.3 ±1.9	8.0 ±0.0	82.7 ±3.8	76.0 ±0.0	80.0 ±5.7	88.0 ±3.3	81.3 ±7.5
FOREWARN	RSS (2025)	69.4	**97.3** ±1.9	84.0 ±3.3	85.3 ±3.8	**96.0** ±0.0	**89.3** ±3.8	86.7 ±6.8	84.0 ±3.3
LPB	NeurIPS (2025)	72.8	89.3 ±5.0	4.0 ±3.3	82.7 ±5.0	68.0 ±6.5	73.3 ±7.5	**89.3** ±3.8	85.3 ±1.9
GPC	RA-L (2026)	72.2	84.0 ±3.3	1.3 ±1.9	81.3 ±3.8	77.3 ±5.0	78.7 ±5.0	88.0 ±6.5	85.3 ±5.0
CoC w/o Geometry	–	67.6	92.0 ±5.7	88.0 ±0.0	88.0 ±5.7	93.3 ±1.9	84.0 ±3.3	80.0 ±6.5	73.3 ±8.2
CoC w/o History	–	70.6	92.0 ±6.5	88.0 ±5.7	90.7 ±5.0	**96.0** ±5.7	82.7 ±7.5	84.0 ±6.5	84.0 ±6.5
CoC w/o Prediction	–	2.8	4.0 ±0.0	0.0 ±0.0	4.0 ±3.3	0.0 ±0.0	8.0 ±3.3	0.0 ±0.0	4.0 ±3.3
CoC w/o Progress	–	71.7	92.0 ±3.3	**96.0** ±0.0	**92.0** ±0.0	**96.0** ±3.3	88.0 ±0.0	84.0 ±3.3	88.0 ±6.5
CoC	–	**82.0**	88.0 ±0.0	86.7 ±1.9	88.0 ±3.3	82.7 ±13.2	82.7 ±1.9	88.0 ±0.0	**90.7** ±7.5

Models	Reference	Light Color	Table Color	Table Texture	Background Texture	Distractor Objects	Camera Pose	RLBench Variations	RO Size
BridgeVLA (Base)	NeurIPS (2025)	41.3 ±1.9	80.0 ±6.5	**85.3** ±1.9	82.7 ±3.8	78.7 ±3.8	82.7 ±3.8	80.0 ±5.7	–
FOREWARN	RSS (2025)	74.7 ±1.9	77.3 ±1.9	57.3 ±7.5	44.0 ±3.3	53.3 ±5.0	38.7 ±3.8	32.0 ±5.7	–
LPB	NeurIPS (2025)	40.0 ±3.3	**89.3** ±1.9	80.0 ±6.5	88.0 ±0.0	**86.7** ±3.8	76.0 ±3.3	**84.0** ±5.7	–
GPC	RA-L (2026)	34.7 ±1.9	80.0 ±13.1	78.7 ±6.8	81.3 ±1.9	84.0 ±6.5	**85.3** ±6.8	82.7 ±3.8	–
CoC w/o Geometry	–	64.0 ±0.0	68.0 ±3.3	60.0 ±0.0	44.0 ±6.5	60.0 ±9.8	40.0 ±3.3	36.0 ±9.8	–
CoC w/o History	–	72.0 ±3.3	76.0 ±3.3	64.0 ±0.0	44.0 ±6.5	68.0 ±13.1	40.0 ±6.5	28.0 ±0.0	–
CoC w/o Prediction	–	0.0 ±0.0	4.0 ±0.0	4.0 ±3.3	0.0 ±0.0	4.0 ±3.3	4.0 ±3.3	4.0 ±3.3	–
CoC w/o Progress	–	**80.0** ±0.0	76.0 ±3.3	56.0 ±6.5	44.0 ±6.5	44.0 ±0.0	44.0 ±6.5	44.0 ±13.1	–
CoC	–	58.7 ±18.9	80.0 ±3.3	72.0 ±3.3	**92.0** ±3.3	78.7 ±18.9	**85.3** ±1.9	80.0 ±0.0	–

F.2 Definitions of Ablated Variants and Prompts

We evaluate the full CoC model against four structurally modified variants, detailed below:

Variant 1: w/o History

Mechanism: Withholds the past observations, forcing the model to evaluate the proposal solely from the current observations.

Ablated Prompt ($t > 0$):
"You ONLY have access to the current single-frame observation. Given the Action Proposal: ..., perform a physical check based SOLELY on the current state: 1. Forward check: Predict ... (**Backward check is DISABLED since history is unavailable**)."

Table 12: **Comparison with SOTA methods on the `place_wine_at_rack_location` task.** The "Avg. Success" column reports the average success rate (%) across 13 perturbations.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	RO Texture	MO Size	RO Size
BridgeVLA (Base)	NeurIPS (2025)	80.3	92.0 ±3.3	8.0 ±0.0	78.7 ±1.9	89.3 ±3.8	88.0 ±3.3	88.0 ±5.7	82.7 ±5.0
FOREWARN	RSS (2025)	47.8	94.7 ±1.9	89.3 ±1.9	93.3 ±1.9	**98.7** ±1.9	86.7 ±1.9	80.0 ±3.3	70.7 ±1.9
LPB	NeurIPS (2025)	78.1	**100.0** ±0.0	6.7 ±1.9	81.3 ±3.8	84.0 ±6.5	82.7 ±8.2	**92.0** ±3.3	**92.0** ±3.3
GPC	RA-L (2026)	81.6	92.0 ±0.0	12.0 ±3.3	88.0 ±5.7	92.0 ±0.0	80.0 ±13.1	88.0 ±5.7	**92.0** ±6.5
CoC w/o Geometry	–	45.7	94.7 ±5.0	84.0 ±3.3	88.0 ±3.3	92.0 ±3.3	**93.3** ±6.8	69.3 ±1.9	72.0 ±3.3
CoC w/o History	–	49.0	96.0 ±3.3	82.7 ±10.5	92.0 ±5.7	92.0 ±6.5	86.7 ±1.9	88.0 ±6.5	80.0 ±6.5
CoC w/o Prediction	–	9.5	8.0 ±6.5	20.0 ±0.0	24.0 ±6.5	20.0 ±6.5	20.0 ±6.5	8.0 ±0.0	8.0 ±0.0
CoC w/o Progress	–	49.1	**100.0** ±0.0	78.7 ±3.8	**96.0** ±3.3	96.0 ±3.3	92.0 ±0.0	80.0 ±0.0	80.0 ±3.3
CoC	–	**89.8**	88.0 ±3.3	**92.0** ±3.3	93.3 ±3.8	90.7 ±1.9	88.0 ±6.5	**92.0** ±0.0	**92.0** ±3.3

Models	Reference	Light Color	Table Color	Table Texture	Background Texture	Distractor Objects	Camera Pose	RLBench Variations	MO Texture
BridgeVLA (Base)	NeurIPS (2025)	86.7 ±5.0	**88.0** ±0.0	**84.0** ±3.3	92.0 ±8.6	74.7 ±6.8	**92.0** ±3.3	92.0 ±3.3	–
FOREWARN	RSS (2025)	38.7 ±3.8	21.3 ±5.0	13.3 ±1.9	12.0 ±0.0	8.0 ±0.0	5.3 ±1.9	4.0 ±0.0	–
LPB	NeurIPS (2025)	70.7 ±6.8	78.7 ±3.8	73.3 ±3.8	86.7 ±5.0	**81.3** ±1.9	88.0 ±3.3	97.3 ±3.8	–
GPC	RA-L (2026)	89.3 ±9.4	**88.0** ±5.7	**84.0** ±9.8	85.3 ±3.8	76.0 ±6.5	89.3 ±3.8	97.3 ±3.8	–
CoC w/o Geometry	–	44.0 ±6.5	20.0 ±3.3	16.0 ±6.5	4.0 ±3.3	8.0 ±3.3	4.0 ±3.3	0.0 ±0.0	–
CoC w/o History	–	48.0 ±6.5	12.0 ±3.3	12.0 ±0.0	12.0 ±3.3	20.0 ±13.1	4.0 ±3.3	8.0 ±3.3	–
CoC w/o Prediction	–	12.0 ±9.8	4.0 ±3.3	0.0 ±0.0	0.0 ±0.0	8.0 ±3.3	0.0 ±0.0	0.0 ±0.0	–
CoC w/o Progress	–	48.0 ±13.1	24.0 ±3.3	16.0 ±0.0	4.0 ±3.3	12.0 ±3.3	4.0 ±3.3	8.0 ±3.3	–
CoC	–	**100.0** ±0.0	**88.0** ±0.0	72.0 ±3.3	**96.0** ±0.0	72.0 ±6.5	**92.0** ±3.3	**100.0** ±0.0	–

Table 13: **Comparison with SOTA methods on the `turn_oven_on` task.** The "Avg. Success" column reports the average success rate (%) across 10 generalization settings.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Size	RO Size
BridgeVLA (Base)	NeurIPS (2025)	91.9	94.7 ±1.9	88.0 ±0.0	94.7 ±1.9	–	82.7 ±5.0	–
FOREWARN	RSS (2025)	91.9	**100.0** ±0.0	96.0 ±0.0	92.0 ±3.3	–	96.0 ±5.7	–
LPB	NeurIPS (2025)	92.1	**100.0** ±0.0	90.7 ±3.8	89.3 ±6.8	–	82.7 ±1.9	–
GPC	RA-L (2026)	92.3	98.7 ±1.9	90.7 ±1.9	96.0 ±0.0	–	86.7 ±5.0	–
CoC w/o Geometry	–	**92.9**	**100.0** ±0.0	**100.0** ±0.0	**97.3** ±1.9	–	96.0 ±0.0	–
CoC w/o History	–	91.9	96.0 ±3.3	98.7 ±1.9	96.0 ±3.3	–	**100.0** ±0.0	–
CoC w/o Prediction	–	68.8	80.0 ±0.0	84.0 ±0.0	76.0 ±6.5	–	68.0 ±0.0	–
CoC w/o Progress	–	87.3	90.7 ±1.9	93.3 ±1.9	84.0 ±6.5	–	88.0 ±3.3	–
CoC	–	92.4	94.7 ±3.8	96.0 ±3.3	92.0 ±0.0	–	96.0 ±0.0	–

Models	Reference	Light Color	Table Color	Table Texture	Background Texture	Distractor Objects	Camera Pose	RLBench Variations
BridgeVLA (Base)	NeurIPS (2025)	90.7 ±5.0	96.0 ±3.3	97.3 ±1.9	94.7 ±1.9	90.7 ±1.9	92.0 ±6.5	92.0 ±3.3
FOREWARN	RSS (2025)	88.0 ±0.0	97.3 ±3.8	96.0 ±0.0	89.3 ±1.9	89.3 ±1.9	96.0 ±0.0	78.7 ±1.9
LPB	NeurIPS (2025)	88.0 ±9.8	96.0 ±3.3	92.0 ±3.3	92.0 ±6.5	96.0 ±3.3	98.7 ±1.9	**96.0** ±3.3
GPC	RA-L (2026)	86.7 ±10.5	93.3 ±5.0	92.0 ±3.3	**97.3** ±3.8	93.3 ±1.9	93.3 ±1.9	93.3 ±1.9
CoC w/o Geometry	–	**92.0** ±3.3	96.0 ±3.3	96.0 ±0.0	84.0 ±3.3	88.0 ±3.3	96.0 ±3.3	84.0 ±0.0
CoC w/o History	–	88.0 ±0.0	**100.0** ±0.0	**100.0** ±0.0	80.0 ±3.3	84.0 ±0.0	96.0 ±0.0	76.0 ±3.3
CoC w/o Prediction	–	56.0 ±0.0	76.0 ±9.8	76.0 ±0.0	60.0 ±9.8	76.0 ±6.5	68.0 ±3.3	48.0 ±3.3
CoC w/o Progress	–	84.0 ±3.3	92.0 ±0.0	88.0 ±3.3	88.0 ±6.5	88.0 ±6.5	92.0 ±0.0	76.0 ±3.3
CoC	–	80.0 ±6.5	96.0 ±3.3	92.0 ±0.0	84.0 ±3.3	**100.0** ±0.0	**100.0** ±0.0	88.0 ±6.5

Variant 2: w/o Geometry

Mechanism: Removes the continuous `geometric_attributes` from the scene graph, forcing reliance only on discrete topological relations.

Ablated Prompt (t = 0):
"... extract the initial scene graph WITHOUT precise geometric attributes. ..."
Ablated Prompt (t > 0):
"1. Backward check: Compare History vs. Current to identify physical stasis. **Do NOT quantify exact geometric distance changes**. 2. Forward check: Predict how the Current Scene Graph (spatial relations only, no geometry) and task progress will change."

Table 14: **Comparison with SOTA methods on the `setup_chess` task.** The "Avg. Success" column reports the average success rate (%) across 12 generalization settings.

Models	Reference	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Texture	MO Size	RO Size
BridgeVLA (Base)	NeurIPS (2025)	6.8	5.3 ±5.0	0.0 ±0.0	0.0 ±0.0	4.0 ±3.3	6.7 ±1.9	6.7 ±3.8	–
FOREWARN	RSS (2025)	6.9	**13.3** ±1.9	8.0 ±3.3	8.0 ±3.3	4.0 ±5.7	4.0 ±3.3	10.7 ±1.9	–
LPB	NeurIPS (2025)	8.8	10.7 ±3.8	0.0 ±0.0	0.0 ±0.0	8.0 ±6.5	5.3 ±1.9	13.3 ±1.9	–
GPC	RA-L (2026)	8.7	8.0 ±3.3	0.0 ±0.0	0.0 ±0.0	8.0 ±3.3	8.0 ±0.0	**20.0** ±5.7	–
CoC w/o Geometry	–	8.9	8.0 ±5.7	4.0 ±0.0	8.0 ±0.0	8.0 ±0.0	8.0 ±3.3	6.7 ±5.0	–
CoC w/o History	–	**9.2**	8.0 ±9.8	0.0 ±0.0	**9.3** ±1.9	**10.7** ±7.5	**10.7** ±1.9	16.0 ±9.8	–
CoC w/o Prediction	–	3.0	12.0 ±6.5	4.0 ±3.3	0.0 ±0.0	4.0 ±3.3	4.0 ±0.0	4.0 ±0.0	–
CoC w/o Progress	–	9.1	12.0 ±6.5	9.3 ±1.9	4.0 ±3.3	8.0 ±3.3	4.0 ±0.0	8.0 ±3.3	–
CoC	–	9.0	9.3 ±3.8	**14.7** ±1.9	4.0 ±3.3	4.0 ±0.0	**10.7** ±1.9	12.0 ±3.3	–

Models	Reference	Light Color	Table Color	Table Texture	Background Texture	Distractor Objects	Camera Pose	RLBench Variations	RO Texture
BridgeVLA (Base)	NeurIPS (2025)	6.7 ±1.9	14.7 ±5.0	6.7 ±3.8	10.7 ±1.9	6.7 ±1.9	6.7 ±1.9	12.0 ±6.5	–
FOREWARN	RSS (2025)	5.3 ±7.5	1.3 ±1.9	6.7 ±6.8	13.3 ±1.9	6.7 ±1.9	4.0 ±3.3	10.7 ±1.9	–
LPB	NeurIPS (2025)	8.0 ±3.3	12.0 ±0.0	**12.0** ±3.3	16.0 ±3.3	2.7 ±1.9	**12.0** ±0.0	**16.0** ±3.3	–
GPC	RA-L (2026)	**12.0** ±6.5	**16.0** ±0.0	4.0 ±3.3	**20.0** ±3.3	0.0 ±0.0	4.0 ±3.3	12.0 ±6.5	–
CoC w/o Geometry	–	8.0 ±0.0	**16.0** ±9.8	8.0 ±6.5	16.0 ±3.3	4.0 ±0.0	**12.0** ±9.8	8.0 ±3.3	–
CoC w/o History	–	**12.0** ±3.3	**16.0** ±6.5	8.0 ±3.3	8.0 ±0.0	**12.0** ±6.5	4.0 ±0.0	4.0 ±0.0	–
CoC w/o Prediction	–	4.0 ±0.0	0.0 ±0.0	4.0 ±3.3	4.0 ±3.3	8.0 ±6.5	0.0 ±0.0	0.0 ±0.0	–
CoC w/o Progress	–	**12.0** ±3.3	12.0 ±6.5	4.0 ±0.0	16.0 ±6.5	8.0 ±0.0	8.0 ±6.5	**16.0** ±6.5	–
CoC	–	0.0 ±0.0	**16.0** ±0.0	10.7 ±1.9	12.0 ±3.3	**12.0** ±6.5	8.0 ±6.5	4.0 ±0.0	–

Variant 3: w/o Prediction

Mechanism: Skips future state prediction, evaluating the proposal based solely on historical and current observations.

Ablated Prompt (t > 0):
"1. Backward check: Compare History vs. Current... 2. Forward check: **Skip future scene graph prediction**. Directly evaluate the structural soundness of the Action Proposal."

Variant 4: w/o Progress

Mechanism: Removes the progress metric, forcing the model to rely totally on local observations without global task awareness.

Ablated Prompt (t = 0):
"... Predict the resulting spatial changes if the Action Proposal ... is applied. Do NOT compute or evaluate global task progress."
Ablated Prompt (t > 0):
"1. Backward check: Compare History vs. Current... 2. Forward check: Predict ... **Skip task progress evaluation**. "

F.3 Analysis of Component Contributions

In the fixed-weight prompt ablations of Table 3, removing forward prediction reduces average success from 71.6% to 18.1%. Removing geometry, history, or progress evaluation yields 50.6%, 51.9%, or 52.1%, respectively. These results show that the trained corrector relies on all four components, with the largest performance loss caused by bypassing prediction.

F.4 Retraining Ablations

To complement the fixed-weight prompt ablations, we retrain six variants from the same MomaGraph-R1 initialization using the training trajectories and SFT hyperparameters of Full CoC. Each variant modifies the supervision targets before training.

- **Direct-SFT** retains the final correction-needed and corrected-action labels but removes scene-graph, progress, and dual-check reasoning targets.
- **P-free** removes task-specific progress rules and progress-derived labels. Correction labels instead use generic geometric anomaly checks, such as distance divergence and object drops.

Table 15: **Retraining ablations of structured supervision on COLOSSEUM.** For each setting, success is averaged over applicable tasks within each run. Entries report the mean and sample standard deviation (%) across three evaluation runs. "Avg. Success" is the unweighted mean over the 14 perturbed settings, excluding "No Perturbations".

Variant	**Avg. Success ↑**	No Perturbations	All Perturbations	MO Color	RO Color	MO Texture	RO Texture	MO Size
Direct-SFT	6.5	15.62 ±0.66	14.48 ±2.16	12.76 ±0.33	10.93 ±1.22	0.00 ±0.00	0.67 ±1.15	9.33 ±2.40
P-free	32.3	49.14 ±1.14	52.00 ±3.57	48.38 ±3.63	35.47 ±2.01	11.33 ±2.31	38.67 ±3.06	48.89 ±3.36
H-0	44.7	60.95 ±1.19	62.48 ±2.16	59.05 ±0.87	52.53 ±1.22	43.33 ±1.15	85.33 ±8.08	58.22 ±3.01
H-25	31.0	39.62 ±1.44	47.24 ±1.19	43.81 ±2.93	36.27 ±2.57	10.67 ±3.06	46.00 ±4.00	41.33 ±1.76
G-5	31.6	45.52 ±1.65	51.43 ±1.51	50.29 ±1.98	33.33 ±4.82	5.33 ±3.06	39.33 ±4.16	50.67 ±1.15
G-50	9.4	19.05 ±0.66	21.33 ±1.32	18.10 ±1.44	16.80 ±0.80	1.33 ±2.31	1.33 ±1.15	14.00 ±1.76

Variant	Light Color	RLBench Variations	Background Texture	Table Texture	Camera Pose	Distractor Objects	Table Color	RO Size
Direct-SFT	8.19 ±1.44	6.67 ±1.44	4.76 ±0.33	4.95 ±0.87	7.24 ±0.87	5.14 ±0.57	6.10 ±1.19	0.00 ±0.00
P-free	29.90 ±3.49	18.48 ±1.19	17.14 ±0.99	25.71 ±1.51	17.71 ±0.57	22.86 ±3.57	30.86 ±4.88	54.67 ±6.11
H-0	35.62 ±2.01	18.86 ±1.14	22.67 ±1.19	32.38 ±0.87	22.10 ±1.84	27.43 ±0.57	37.90 ±0.33	68.00 ±6.93
H-25	25.14 ±2.49	16.19 ±0.87	17.71 ±2.62	21.71 ±1.14	19.62 ±3.67	21.14 ±0.57	22.29 ±4.68	65.33 ±9.24
G-5	28.57 ±3.02	18.48 ±1.65	18.86 ±0.57	22.29 ±1.98	19.43 ±0.57	22.48 ±2.16	25.14 ±0.99	57.33 ±9.24
G-50	9.90 ±1.75	5.90 ±0.33	7.24 ±1.32	7.62 ±1.19	6.48 ±1.75	10.29 ±2.62	6.86 ±1.51	4.00 ±0.00

- **H-0** retains scene-graph, progress, future-state, and correction-needed targets but removes corrected-action targets on failure frames. **H-25** retains 25% of these heuristic corrected-action labels.
- **G-5** and **G-50** perturb the object poses used for annotation by 5 mm / 5° and 50 mm / 30°, respectively. All pose-derived scene graphs, progress labels, and correction labels are then recomputed.

Table 15 shows that Direct-SFT reaches an average success rate of 6.5%, compared with 71.6% for Full CoC in Table 3. Since Direct-SFT retains the final correction labels, this gap supports the contribution of intermediate structured supervision beyond direct action imitation. Replacing progress-based supervision with generic geometric anomaly targets in P-free also reduces success to 32.3%, indicating the importance of relating correction targets to task progress. The correction-label ablations show that label quantity alone does not explain performance. H-0 outperforms H-25 (44.7% versus 31.0%), although both remain below Full CoC. Increasing the geometric perturbation from G-5 to G-50 reduces success from 31.6% to 9.4%. Even the smaller perturbation degrades performance, showing that accurate geometric targets are important for training the corrector. Together, these results support the complete supervision scheme while identifying its sensitivity to altered or corrupted targets.

G Real-World Experiments

This section details the hardware setup, data collection protocols, and the specific settings of our real-world evaluations.

G.1 Hardware Setup

We deploy a tabletop manipulation system consisting of a dual-arm Agilex ALOHA robot [49]. As shown in Fig. 8, the experimental setup integrates two fixed cameras and a wrist-mounted camera to provide comprehensive visual observations.

In our experiments, we utilize a single arm of the ALOHA system. The low-level robot actions are defined in a 7-DoF continuous space, encompassing the 6-DoF end-effector pose (translation and rotation) and the 1-DoF gripper state. The physical control loop operates at a frequency of 10Hz.

G.2 Task Definitions and Data Collection Protocols

Four distinct real-world tasks are designed to evaluate the geometric grounding, generalizable error recovery, and long-horizon recovery abilities.

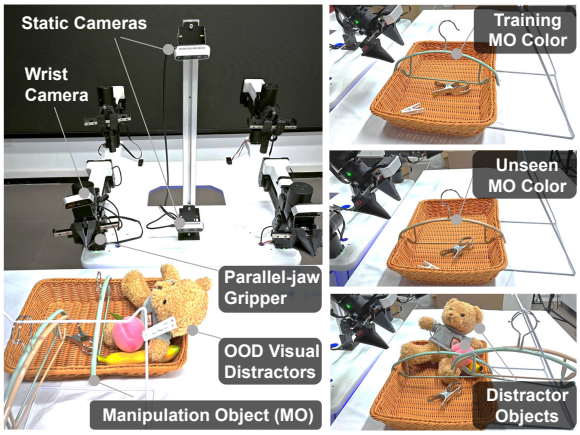


Figure 8: **(Left) Real-world hardware setup. (Right) Visualization of generalization settings.** The *Unseen Color* setting features OOD brown hangers, while the *Visual Distractors* setting features task-irrelevant objects (including a toy and fruits) cluttering the workspace.

- **Task 1: Move Hanger.** The robot must grasp a thin clothes hanger and accurately hook it onto a white metal rack. This task requires high geometric precision, where slight Z-axis deviations can cause physical collisions and task failure. As this task exactly parallels the simulated COLOSSEUM benchmark, it serves to robustly validate the sim-to-real transferability of our framework. *Success condition*: The hanger remains securely suspended on the rack without human intervention after the gripper fully releases.
- **Task 2: Move Hanger with Unseen MO Color.** We replace the green hanger used for training with a brown variant to evaluate zero-shot generalization under visual perturbations.
- **Task 3: Move Hanger with Distractor Objects.** We introduce OOD distractors by scattering colorful fruits and a toy around the rack, testing the steering method's resistance to severe visual feature changes.
- **Task 4: Pack into Box (Long-Horizon).** The robot must sequentially place four transparent cans into a compact cardboard box. This task is highly sensitive to myopic placements, since placing the first can in the center will leave insufficient physical space for the remaining cans, leading to unrecoverable dead-ends. *Success condition*: All four cans are entirely placed within the boundaries of the cardboard box without causing the box to tip over.

Data Collection and Base Policy Training: In line with the simulated experiments, we adopt a state-of-the-art imitation learning architecture, BridgeVLA [4], as the base policy. Using the ALOHA leader-follower teleoperation system, we collected exactly 50 expert demonstrations specifically for the *Move Hanger* and *Pack into Box* tasks in clean, distractor-free environments. A single BridgeVLA agent is fine-tuned from the official COLOSSEUM pre-trained weights in a multi-task setting exclusively on these 2 base tasks.

During inference, the base policy outputs step-wise action proposals, which are subsequently evaluated and refined by CoC and other evaluated steering baselines. CoC is fine-tuned on demonstrations and failure rollouts from Tasks 1 and 4. Offline annotations are generated using ALOHA forward kinematics and calibrated multi-view perception with SAM 3 [53]. Tasks 2 and 3 are evaluated without additional task-specific CoC fine-tuning.

G.3 Evaluation Protocols and Generalization Settings

For each of the four real-world tasks, every evaluated method undergoes $N = 20$ independent trials. To prevent policies from memorizing fixed trajectories, initial conditions are randomized. Specifically, the robot's initial end-effector pose is perturbed by translations of $\Delta_{x,y} \in [-5\text{ cm}, 5\text{ cm}]$ and yaw rotations of $\Delta_\theta \in [-15°, 15°]$. Simultaneously, the target objects are randomly initialized within a predefined 30 cm $\times$ 30 cm workspace.

Table 16: **Real-robot comparison with unsteered BridgeVLA.** Each method is evaluated in 20 trials per task under the same protocol. Entries report successful trials / total trials, with success rates in parentheses. Overall aggregates all 80 trials per method.

Method	Task 1 Move Hanger	Task 2 Unseen MO Color	Task 3 Visual Distractors	Task 4 Pack into Box	Overall
BridgeVLA	14/20 (70.0%)	10/20 (50.0%)	9/20 (45.0%)	5/20 (25.0%)	38/80 (47.5%)
CoC	**19/20 (95.0%)**	**17/20 (85.0%)**	**16/20 (80.0%)**	**17/20 (85.0%)**	**69/80 (86.25%)**

A trial is strictly recorded as a *Success* only if all sub-actions are completed and the final goal state is structurally stable. Protective stops triggered by severe physical collisions or timeouts (exceeding 900 physical execution steps) are recorded as *Failures*.

To evaluate the system's robustness and zero-shot generalization to out-of-distribution (OOD) visual variations, we introduce two challenging settings within the *Move Hanger* task. While the base policy is trained on the foundational *Move Hanger* task, these settings expose the model to visual conditions significantly deviating from the training manifold.

- *Unseen Object Colors (Visual Domain Shift):* The base training data exclusively contains green hangers. During testing, we substitute them with highly saturated brown variants. This evaluates the model's ability to maintain geometric grounding despite significant shifts in color distribution and appearance.
- *Visual Distractors (OOD Noise):* We scatter complex, task-irrelevant objects (*e.g.*, colorful stuffed fruits and a toy bear) cluttering the manipulation workspace. This serves as severe visual noise, designed to induce high feature reconstruction errors in predictive steering baselines that rely on rigid pixel- or latent-space matching.

G.4 Supplementary Video Guide

We strongly encourage readers to view the supplementary videos, which provide dynamic visualizations of the CoC framework in real-world steering. The videos are structured as follows:

- `real_task_move_hanger.mp4` | **Task 1:** Demonstrates CoC executing Z-axis spatial refinements to prevent clipping, contrasted with the baseline's fatal collision.
- `real_task_move_hanger_with_unseen_MO_color.mp4` | **Task 2:** Highlights CoC's robustness against unseen brown hangers.
- `real_task_move_hanger_with_distractor.mp4` | **Task 3:** Highlights CoC's robustness against severe visual distractors. Note how the baseline's corrupted unstructured prediction leads to continuous false-positive action rejections, ultimately resulting in a timeout failure.
- `real_task_pack_in_box.mp4` | **Task 4:** Showcases that in contrast to progress-blind baselines, CoC avoids a myopic placement (*i.e.*, a false negative) by aligning the first can to the corner, thereby preserving capacity for subsequent steps.

G.5 Unsteered BridgeVLA Baseline

To quantify the contribution of steering on hardware, we compare CoC with unsteered BridgeVLA under the four-task protocol in Section G.3. Both methods use the same frozen base policy and observation modalities.

Table 16 shows higher success with CoC on all four tasks, with the largest gain on Pack into Box (25.0% to 85.0%). This task requires early placements to preserve space for later objects, so the improvement is consistent with the value of evaluating actions against the final task goal. Overall, CoC increases completed trials from 38 to 69 out of 80, supporting the practical benefit of correcting base-policy proposals during execution.

Table 17: Summary of third-party assets.

Asset	Category	Source	License	URL
COLOSSEUM	Dataset	[43]	MIT	https://huggingface.co/datasets/colosseum/colosseum-challenge
BridgeVLA	Base Policy	[4]	Apache-2.0	https://github.com/BridgeVLA/BridgeVLA
Momagraph-R1	Pre-trained VLM	[30]	MIT	https://huggingface.co/cheryyunl/MomaGraph-R1
GPC	Baseline	[13]	Unspecified	https://github.com/han20192019/gpc_code
FOREWARN	Baseline	[12]	Unspecified	https://github.com/CMU-IntentLab/Forewarn
LPB	Baseline	[11]	Unspecified	https://github.com/zhanyisun/lpb
Qwen2.5-VL-7B-Instruct	General VLM	[44]	Apache-2.0	https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct
Llama-3.2-11B-Vision-Instruct	General VLM	[45]	llama3.2	https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct
Gemini 3.1 Pro	General VLM	[46]	Proprietary	https://ai.google.dev/
GPT-5.4	General VLM	[47]	Proprietary	https://openai.com/api/
VLM-3R	Embodied VLM	[48]	Apache-2.0 / CC BY-NC-SA	https://github.com/VITA-Group/VLM-3R
SAM 3	Segmentation model	[53]	SAM License	https://github.com/facebookresearch/sam3

H Licenses for Existing Assets

Table 17 summarizes the attribution and license information for all third-party assets used in this work. All assets are used in accordance with their respective original licenses and terms of use.

Terms of Use Compliance. All listed assets, including datasets, codes and models, were utilized exclusively for non-commercial academic benchmarking and robotic reasoning tasks. Specifically for VLM-3R, we strictly comply with the CC BY-NC-SA 4.0 restrictions imposed by its dependencies, ensuring no commercial deployment or usage that violates original restricted use cases.

I Broader Impacts

The proposed Chain-of-Correction framework fundamentally aims to improve the reliability, safety, and error-recovery capabilities of autonomous robotic systems.

Potential Positive Impacts. By enabling inference-time policy correction without relying on human interventions, this work reduces the cost and technical barriers of deploying robust robots in unstructured or dynamic environments. It promotes reliable embodied AI systems by enabling recovery from unexpected physical anomalies, ultimately facilitating more efficient human-robot collaboration in tasks ranging from healthcare logistics to assistive robotics.

Potential Negative Impacts and Mitigations. Despite its intended safety benefits, any residual errors in safety-critical deployments could lead to unexpected physical outcomes. Therefore, we emphasize that our framework should not be deployed in isolation. It must be coupled with hardware-level safety constraints, collision-avoidance overrides, and human oversight.

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 comprehensive evaluations in Section 4 (across a novel offline predictive reasoning protocol, the COLOSSEUM benchmark, and real-robot tasks) directly substantiate our claims, demonstrating that CoC achieves superior fine-grained error detection and robust error recovery against state-of-the-art baselines.

   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: The limitations are explicitly discussed in the conclusion section 5.

   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: This paper does not contribute 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: Details regarding SFT dataset construction are provided in Section 3 and Appendix A. Experiment setups are described in Section 4, with full reproducibility details provided in Appendices B through G. Source code and evaluation scripts are included in the supplementary materials.

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: Expert demonstration data is sourced from the public dataset [43], and failure rollouts can be generated by executing the COLOSSEUM-finetuned checkpoint from [4]. The supplementary material provides the complete source code, including scripts for SFT dataset construction, model fine-tuning, and reproducing the main simulation evaluations.

   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: General experimental setups are outlined in Section 4. Specific training and test details, including data splits and a comprehensive hyperparameter table detailing the optimizer and learning rates, are explicitly provided in Appendices A and B.

   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: For offline predictive reasoning and latency analysis, we report mean metrics and standard deviations across three independent evaluation runs. COLOSSEUM results are additionally summarized across applicable tasks (Tables 2 and 3) and across applicable perturbation settings within each task (Figure 3). For real-robot deployments, variance metrics are omitted due to intensive physical resource constraints, as is standard practice in robotics research.

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: The full details of compute requirements are provided in Appendix B.

   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: We confirm that all research conducted and reported in this work strictly conforms to 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 discuss the broader impacts of the paper in Appendix I.

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: The models and datasets in this work are specialized for robotic kinematic reasoning and manipulation tasks within controlled simulation and real-world environments. They do not possess capabilities for generating disinformation, malicious content, or other risks associated with general-purpose generative models. Therefore, they pose no significant risk for misuse.

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: All third-party assets used in this work are properly cited. Detailed information regarding their original licenses, terms of use, and repository URLs is consolidated in Appendix H.

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: Detailed documentation for our introduced offline evaluation protocol and data collection pipeline is provided in Sections 3, 4, and Appendices A, D. Furthermore, an anonymized codebase is provided in the supplementary material, with its implementation details covered in the aforementioned appendices.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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: This paper does not involve crowdsourcing nor research with human subjects.

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: [Yes]

Justification: A VLM serves as the core predictive reasoning engine of our Chain-of-Correction framework. Its implementation details, including input formulation and supervised fine-tuning, are explicitly described in Section 3 and Appendices A and B.

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.