OGPO: Offline Goal-conditioned Policy Optimization for Recoverable Vision-Language-Action Models

Xule Gao[1,2,3] **Xi Wang**[1,2,3] **Zehua Zang**[4] **Rui Wang**[1,2]
Changwen Zheng[1,2] Chuxiong Sun[1,2,†]

[1] National Key Laboratory of Space Integrated Information System
[2] Institute of Software, Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences
[4] Wangxuan Institute of Computer Technology, Peking University
gaoxule24@mails.ucas.ac.cn
[†] Corresponding author: chuxiong2016@iscas.ac.cn

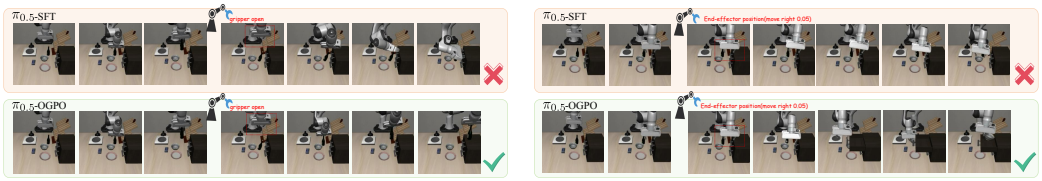
Abstract

Vision-language-action (VLA) models have shown strong promise in robotic manipulation, yet their post-training still predominantly relies on trajectory-level supervised imitation over robot demonstrations. While effective for learning feasible actions, this paradigm supervises policies to reproduce expert actions at each step, biasing them toward demonstration-specific action paths and providing limited guidance for recovering toward task-progress states once execution deviates from demonstrated trajectories. To address this limitation, we propose OGPO, an Offline Goal-conditioned Policy Optimization framework for recoverable VLA post-training. Instead of treating offline demonstrations merely as step-wise action labels, OGPO adaptively relabels trajectories with semantic key states and constructs goal-conditioned process rewards to optimize the policy for reaching these task-progress states. In this way, OGPO shifts VLA post-training from trajectory-level action imitation to key-state reachability learning, enabling policies to acquire a more robust ability to complete goal states from feasible off-demonstration states without requiring additional online interaction. Experiments on LIBERO and MetaWorld show that OGPO consistently improves VLA policy performance over standard supervised fine-tuning. Moreover, evaluations on LIBERO-Plus and our proposed LIBERO-ReAct, a mid-execution perturbation benchmark, further demonstrate that OGPO achieves stronger robustness and goal-directed recoverability under distribution shifts and recoverable off-demonstration states.

1 Introduction

Vision-language-action (VLA) policies have shown strong potential for language-conditioned robotic manipulation by scaling supervised action prediction on large-scale robot datasets [4, 50, 19]. Most existing VLA policies are adapted to downstream tasks through supervised fine-tuning (SFT), where the model is optimized to predict expert actions from visual observations and language instructions on demonstration trajectories [18, 39, 3, 2]. Although this paradigm is simple and stable, it treats demonstrations primarily as step-wise action labels for trajectory-level imitation. As a result, policies trained purely with SFT tend to reproduce demonstration-specific action paths and remain inherently bounded by the coverage and quality of demonstration data. As shown in Figure 1, when execution deviates from demonstrated trajectories, such policies may suffer from compounding errors in long-horizon tasks and lack explicit guidance for recovering toward task-progress states [31, 45].

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



(a) Put the wine bottle on top of the cabinet.

(b) Open the top of drawer and put the bowl inside.

Figure 1: We design two representative perturbation cases to evaluate the robustness and goal-directed recoverability of offline-trained VLA policies under deviations from demonstration trajectories. In case (a), the gripper is forcibly opened after grasping the object, while position perturbation is applied when the end effector approaches the drawer. The results show that $\pi_{0.5}$-SFT fails to complete the task, whereas $\pi_{0.5}$-OGPO successfully recovers toward the goal state and finishes the task, demonstrating stronger recoverability under execution-time deviations.

Recent studies have explored reinforcement learning (RL) to further improve pretrained VLA policies beyond supervised imitation, including iterative online RL fine-tuning [12], PPO/GRPO-style policy-gradient optimization [27, 22, 6], residual policy learning [43], and RL-based policy distillation [17]. By optimizing task-level rewards, RL can in principle encourage goal-directed behaviors that are not directly captured by step-wise action imitation. However, existing RL-based VLA post-training methods typically require costly environment rollouts and suffer from limited sample efficiency, making them difficult to scale for real-world robotic post-training.

In this work, we study whether VLA policies can acquire goal-directed recoverability purely from offline robot data, without requiring additional online interaction. We define this ability as recovering toward task-progress key states when execution deviates from demonstrated trajectories. Our key insight is that offline demonstrations should not be used merely as action sequences to imitate; they also contain implicit evidence about which task-progress states are reachable and which behaviors move the policy toward them. Based on this insight, we propose OGPO, an offline goal-conditioned policy optimization framework for recoverable VLA post-training. Specifically, we retain SFT as a lightweight action prior to preserve low-level action feasibility and language-action alignment, while introducing an offline goal-conditioned RL objective to learn key-state reachability from existing trajectories. To turn offline data into goal-reaching supervision, OGPO adaptively relabels trajectories with task-progress key states and constructs goal-conditioned process rewards to guide both policy optimization and value estimation. Instead of supervising the policy to imitate every action along the demonstration path, this objective trains the policy to reach task-progress states under goal-conditioned rewards. In this way, OGPO shifts VLA post-training from trajectory-level action imitation to key-state reachability learning, enabling policies to improve goal-directed recoverability under feasible off-demonstration deviations while avoiding the high interaction cost of online RL.

We evaluate OGPO on LIBERO [24] and MetaWorld [44], and further test robustness and recoverability on LIBERO-Plus [10] and our proposed LIBERO-ReAct benchmark with mid-execution perturbations. The results show that OGPO consistently improves over SFT baselines, especially on long-horizon and difficult tasks, while achieving stronger robustness and goal-directed recoverability under distribution shifts and execution-time deviations. Our contributions are threefold.

- First, we formulate VLA post-training from the perspective of goal-directed recoverability and identify the limitation of trajectory-level SFT in recovering toward task-progress states under execution deviations.
- Second, we propose OGPO, an offline goal-conditioned policy optimization framework that adaptively relabels fixed robot trajectories with task-progress key states and constructs goal-conditioned process rewards, allowing the policy to learn and optimize goal-directed behaviors beyond step-wise action imitation.
- Third, we conduct extensive experiments on standard manipulation, distribution-shift, and mid-execution perturbation benchmarks, demonstrating that OGPO consistently improves task performance, robustness, and goal-directed recoverability without requiring additional online interaction.

2 Related Works

VLA post-training aims to adapt pretrained vision-language-action policies to downstream robotic manipulation tasks while preserving their general action priors and improving task-specific performance. Existing approaches mainly follow two directions: SFT and RL.

2.1 Supervised Fine-tuning for VLA Policies

SFT has become a dominant post-training strategy for adapting pretrained VLA policies to downstream robotic tasks. In this paradigm, the model is fine-tuned on expert demonstration trajectories by minimizing a supervised action prediction loss conditioned on visual observations and language instructions. Existing VLA models instantiate this paradigm with different action representations and prediction objectives. OpenVLA [19] discretizes continuous robot actions into action tokens and optimizes the policy with a next-token prediction objective using cross-entropy loss. Octo [39] predicts continuous action chunks through a diffusion-based action head trained with a diffusion loss. π_0 [3] and $\pi_{0.5}$ [2] further adopt flow-matching objectives for continuous action-chunk prediction. Recent studies such as OpenVLA-OFT [18] improve SFT-based adaptation by introducing continuous action representations, action chunking, parallel decoding, and regression-based imitation objectives. Despite these advances, SFT remains inherently constrained by the coverage and quality of demonstration data: it primarily learns to reproduce expert actions along demonstrated trajectories, and thus provides limited support for generalizing to unseen states, handling execution perturbations, and mitigating long-horizon compounding errors [31].

2.2 Reinforcement Learning for VLA Post-training

Reinforcement learning (RL) has been extensively studied for sequential decision-making across diverse settings [28, 32, 13, 20, 38, 36, 35, 34, 37]. Recent works have explored RL as an additional post-training stage after SFT. By optimizing task-level rewards through online rollouts, RL allows VLA policies to move beyond the behavior distribution of expert demonstrations and potentially acquire more robust behaviors under distribution shift [25]. For example, iRe-VLA [12] alternates between online RL and supervised learning to improve policy performance while maintaining training stability. Other methods apply PPO- or GRPO-style policy-gradient optimization to VLA models [27, 22, 6], aiming to enhance long-horizon decision-making and task success through outcome-driven rewards. In addition, residual RL methods improve pretrained VLA policies by learning lightweight corrective policies around frozen generalist models [43], while RL-based policy distillation methods leverage VLA teachers or RL experts to obtain more specialized policies [17]. More recent studies further explore simulator- or world-model-based reinforcement fine-tuning to reduce the cost of direct real-world interaction [42, 23, 48, 16]. Although these methods demonstrate the promise of RL for improving VLA policies beyond SFT, they typically rely on online rollouts, real-world interaction, or simulator-based policy evaluation, making them costly and sample-inefficient for long-horizon robotic tasks.

Our distinction. In this work, we combine the benefits of SFT and RL in a fully offline VLA post-training framework. Unlike online or simulator-based RL methods, OGPO derives goal-reaching and failure-aware supervision directly from fixed offline trajectories. It also differs from hierarchical or subgoal-based VLA methods, such as CoT-VLA [47] and ThinkAct [14], which explicitly generate visual subgoals or latent plans during inference. Instead, OGPO uses relabeled local goals to construct process rewards for value learning and advantage-weighted policy optimization, without introducing additional inference-time modules. To the best of our knowledge, OGPO is the first framework that combines SFT regularization and goal-relabeled offline RL to improve VLA recoverability.

3 Method

In this section, we present OGPO, an offline goal-conditioned policy optimization framework for recoverable VLA models. OGPO operates in three stages. First, it relabels offline trajectories into key-state-centered goal-reaching segments, thereby exposing the task-progress structure hidden in demonstrations. Second, it converts segment-level completion labels into goal-conditioned process rewards, which provide dense learning signals for value estimation. Third, it performs offline

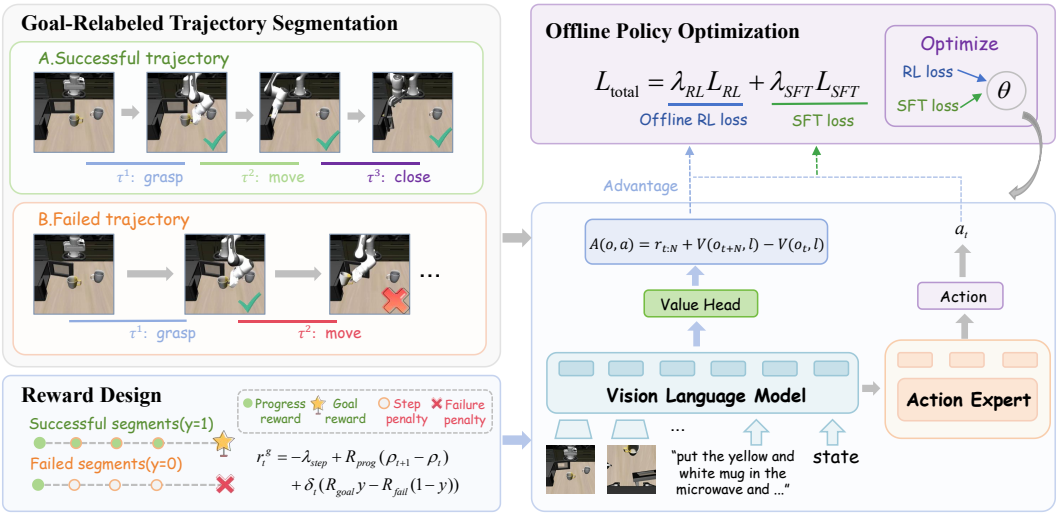


Figure 2: Offline Goal-conditioned Policy Optimization(OGPO) framework.

advantage-weighted policy optimization with SFT regularization, enabling the policy to improve toward goal-reaching behaviors while preserving feasible action priors.

3.1 Goal-Relabeled Trajectory Segmentation

To discover the task-progress structure embedded in offline trajectories, OGPO first segments each trajectory according to gripper state change. Given an expert demonstration trajectory τ:

$$\tau = \{\{(o_t, a_t)\}_{t=1}^{T}, l\} \tag{1}$$

where o_t and a_t denote the observation and action at timestep t, and l is a natural language instruction describing the task carried out in the demonstration. We assume that each trajectory τ is composed of a sequence of task-progress stages. The objective of trajectory segmentation is to split the trajectory into non-overlapping temporal segments $\{\tau^1, \tau^2, \ldots, \tau^M\}$, each corresponding to a task-progress stage. Following Chen et al. [8] and Zhu et al. [49], we realize such segmentation based on keyframes containing the change of gripper state. Despite its simplicity, gripper state change is a robust indicator of critical physical interaction, which can be viewed as a signal of the end of one stage and the beginning of the next.

After segmentation, we perform goal relabeling for each segment. Specifically, we first assign each segment τ^m a semantic local goal g^m based on the task instruction, gripper transitions, and observed state changes, as detailed in Appendix B.2. We then assign each segment a binary completion label $y^m = \mathbb{I}[\tau^m \text{ achieves } g^m]$, using different labeling procedures for successful and failed trajectories:

- For a successful trajectory, each valid segment is treated as completing its assigned local goal and is therefore labeled with $y^m = 1$. The segment-end observation is regarded as the corresponding achieved task-progress key state.
- For a failed trajectory, the task-level outcome does not directly indicate which local goals have been completed, as meaningful intermediate progress may occur before the eventual failure. We therefore introduce an offline VLM-based annotator to evaluate each segment and assign a binary completion label y^m. Details of the completion labeling are provided in Appendix B.3 and B.4.

After relabeling, we obtain a goal-relabeled offline dataset:

$$\widetilde{\mathcal{D}} = \{(o_t, l, a_t, g_t, y_t)\}. \tag{2}$$

Through this process, we obtain a diverse goal-relabeled dataset that contains both achieved and unfulfilled task-progress key states, providing a solid foundation for subsequent VLA policy optimization.

3.2 Designing Goal-Conditioned Process Rewards for Value Learning

Directly constructing a goal-conditioned policy on top of a pretrained VLA model would require architectural modifications and may interfere with the pretrained vision-language representations. To avoid this, we use goal-conditioned process rewards as training-time signals for offline policy optimization. This design preserves the original VLA policy structure and instruction-conditioned action interface, while still enabling the policy to acquire goal-directed behaviors by optimizing toward task-progress key states. Concretely, we construct goal-conditioned process rewards from the relabeled key states and completion labels. The relabeled goal g_t is used only to annotate rewards and define segment-level goal progress during training; it is not introduced as an additional input to either the policy or the value function.

For each relabeled segment $\tau^m = \{(o_t, l, a_t, g_t, y_t)\}_{t=s_m}^{e_m}$, where s_m and e_m denote the start and end time steps of the segment, we define a normalized progress variable:

$$\rho_t^m = y_t \cdot \frac{t - s_m}{e_m + 1 - s_m}, \qquad t \in [s_m, e_m + 1]. \tag{3}$$

When $y_t = 1$, ρ_t^m increases from 0 to 1 along the segment, providing dense progress reward for a completed goal. When $y_t = 0$, $\rho_t^m = 0$ for the entire segment, so failed segments do not receive progressive positive rewards.

For each transition $t \in [s_m, e_m]$, we use a segment-end indicator $\delta_t^m = \mathbb{I}[t = e_m]$, which equals 1 only at the final transition of the segment and 0 otherwise. The reward is then annotated as:

$$r_t = -\lambda_{\text{step}} + R^{\text{prog}} \left(\rho_{t+1}^m - \rho_t^m\right) + \delta_t^m \left(R_{\text{success}}^{\text{goal}} y_t - R_{\text{fail}}^{\text{goal}}(1 - y_t)\right). \tag{4}$$

The first term is a small step penalty that discourages redundant actions. The second term assigns dense positive credit to transitions inside achieved key-state segments. The last term provides a terminal bonus when the assigned key state is achieved and a terminal penalty when the key state remains unfulfilled. The reward-augmented offline dataset is written as:

$$\widetilde{\mathcal{D}} = \{(o_t, l, a_t, g_t, y_t, r_t)\}. \tag{5}$$

With the annotated rewards, we train a value function $V_\psi(o_t, l)$ to estimate the expected discounted return induced by the key-state relabeled rewards. For value learning, we use an n-step Bellman backup to construct the temporal-difference target:

$$\hat{V}_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V_{\bar{\psi}}(o_{t+n}, l), \tag{6}$$

where $V_{\bar{\psi}}$ denotes a target value network or a stop-gradient copy of the value model. In implementation, V_ψ is obtained by attaching a lightweight value head to the intermediate representation of the VLA backbone. The value function is optimized by minimizing the temporal-difference error:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(o_t, l) \sim \widetilde{\mathcal{D}}} \left[\left(V_\psi(o_t, l) - \hat{V}_t \right)^2 \right]. \tag{7}$$

Through this objective, states that are followed by achieved key-state progress are assigned higher values, while states in failed or incomplete segments receive lower values due to the absence of progress rewards and the terminal failure penalty.

3.3 Offline Goal-conditioned Policy Optimization

In this work, we seek a lightweight policy improvement objective that can be seamlessly integrated into the existing action modeling loss of a pretrained VLA policy. Concretely, we adopt an AWAC-style actor update [29], which casts policy improvement as advantage-weighted behavior regression.

We calculate the advantage value for each action a_t using an n-step return estimate induced by the goal-conditioned rewards:

$$A_t = \left(\sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V_\psi(o_{t+n}, l) \right) - V_\psi(o_t, l). \tag{8}$$

A positive advantage indicates that the observed action leads to a higher return than expected from the current state, while a small or negative advantage suggests lower contribution to future progress. Following AWAC [29], we convert the advantage into an exponential policy improvement weight:

$$\omega_t = \mathrm{sg}\left[\min\left(\exp\left(\frac{A_t}{\beta}\right), \omega_{\max}\right)\right], \tag{9}$$

where β controls the sharpness of advantage weighting, $\omega_{\max}$ prevents a small number of high-advantage samples from dominating the update, and $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation.

We then optimize the policy with an AWAC-style offline RL loss over the entire relabeled dataset $\widetilde{\mathcal{D}}$:

$$\mathcal{L}_{\mathrm{RL}}(\theta) = -\mathbb{E}_{\tilde{x}_t \sim \widetilde{\mathcal{D}}}\left[\omega_t \log \pi_\theta(a_t \mid o_t, l)\right]. \tag{10}$$

This objective biases policy improvement toward high-advantage actions in the relabeled dataset: actions that contribute to achieved key states receive larger weights, while actions with low progress return have limited influence on the policy update. As a result, the policy is encouraged not merely to imitate demonstrated actions, but to improve its ability to recover from intermediate states and continue progressing toward the intended goal states.

Furthermore, inspired by recent post-training paradigms in large language models (LLMs) that unify supervised and reinforcement objectives within a single optimization stage[11, 26, 46], we further retain SFT in the policy objective. Specifically, SFT is applied to successful demonstrations as a lightweight action prior, preserving low-level action feasibility and maintaining the language-action alignment inherited from the pretrained VLA model.

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{x_t \sim \mathcal{D}_{\mathrm{demo}}}\left[\log \pi_\theta(a_t \mid o_t, l)\right]. \tag{11}$$

The final policy optimization objective is:

$$\mathcal{L}_{\mathrm{total}}(\theta) = \lambda_{\mathrm{RL}}\mathcal{L}_{\mathrm{RL}}(\theta) + \lambda_{\mathrm{SFT}}\mathcal{L}_{\mathrm{SFT}}(\theta) \tag{12}$$

SFT and offline RL play complementary roles in this objective. SFT provides a coarse-grained behavioral prior, anchoring the policy to the feasible action distribution of expert demonstrations and stabilizing optimization. In contrast, offline RL provides fine-grained goal-directed optimization signals through goal-conditioned process rewards and advantage-weighted policy improvement, enabling the policy to deviate from demonstration-specific action paths when necessary and favor behaviors that better advance toward task-progress key states.

For notational simplicity, we represent the action-matching term in the above objectives using the likelihood-style form $\log \pi_\theta(a_t \mid o_t, l)$. This term serves as a general action-matching objective that encourages the policy output to align with the demonstrated action a_t. In practice, it is instantiated according to the action head of the underlying VLA model, such as action-token likelihood [19], continuous action regression [18], denoising prediction [9, 39], or flow-matching training objectives [3, 2].

4 Experiments

4.1 Experimental Setup

Benchmarks. To evaluate the effectiveness of our method, we conduct comprehensive experiments on two widely used robotic manipulation benchmarks, LIBERO [24] and MetaWorld [44]. To further assess robustness and performance under off-demonstration conditions, we design two additional evaluation settings. First, we use LIBERO-Plus [10] to test distribution shifts in camera viewpoint, robot initial state, visual noise, and object layout. Second, to evaluate dynamic recovery under execution-time disturbances, we introduce **LIBERO-ReAct**, a mid-execution perturbation benchmark built on LIBERO [24]. LIBERO-ReAct includes action noise, gripper-state perturbations, and end-effector position perturbations, and is designed to evaluate goal-directed recoverability during policy execution.

Backbones and Baselines. OGPO is built on the π_0 and $\pi_{0.5}$ backbones. We compare with their SFT-based few-shot and full-shot post-training variants to evaluate improvements over standard

Table 1: Performance comparison on the LIBERO benchmark. Policy input notation: T (third-view image), I (instruction), W (wrist image), P (proprioception). OGPO is built upon the few-shot SFT policy, with ↑ indicating performance gains over the few-shot baseline.

Model	Input	LIBERO				
		Spatial	Object	Goal	Long	Avg.
OpenVLA[19]	T+I	84.7	88.4	79.2	53.7	76.5
π_0-fast[30]	T+W+P+I	96.4	96.8	88.6	60.2	85.5
π_0[3]	T+W+P+I	96.8	98.8	95.8	85.2	94.2
SmolVLA[33]	T+W+P+I	93.0	94.0	91.0	77.0	88.8
WorldVLA[5]	T+I	85.6	89.0	82.6	59.0	79.1
NORA[15]	T+I	92.2	95.4	89.4	74.6	87.9
CoT-VLA[47]	T+I	87.5	91.6	87.6	69.0	81.1
UniVLA[40]	T+I	96.5	96.8	95.6	92.0	95.2
MolmoAct[21]	T+I	87.0	95.4	87.6	77.2	86.6
ThinkAct[14]	T+I	88.3	91.4	87.1	70.9	84.4
GROOT N1[1]	T+I	94.4	97.6	93.0	90.6	93.9
OpenVLA-OFT[18]	T+W+P+I	96.2	98.3	96.2	90.7	95.3
$\pi_{0.5}$-Full[2]	T+W+P+I	98.8	98.2	98.0	92.4	96.85
TGRPO[7]	T+I	90.4	92.2	81.0	59.2	80.7
VLA-RL[27]	T+I	90.2	91.8	82.2	59.8	81.0
SimpleVLA-RL[22]	T+I	98.2	98.7	98.8	91.7	96.9
$\pi_{0.5}$-PPO (RLinf)[6]	T+W+P+I	99.4	99.8	98.8	94.0	98.0
$\pi_{0.5}$-Few Shot	T+W+P+I	84.6	95.4	84.6	43.9	77.1
+ OGPO	T+W+P+I	99.2	98.6	97.6	93.2	97.1
		↑14.6	↑3.2	↑13.0	↑49.3	↑20.0

supervised imitation. We further report representative VLA models, including OpenVLA [19], SmolVLA [33], UniVLA [40], and OpenVLA-OFT [18], as broader references, and include PPO-based online RL baselines from RLinf [6] for comparison with online RL post-training. Overall, this baseline set provides a comprehensive and rigorous evaluation for assessing the effectiveness of our OGPO.

4.2 Main Results

Our evaluation begins with the overall performance on widely used manipulation benchmarks. Tables 1 and 2 present the results on LIBERO and MetaWorld-MT50. On LIBERO, OGPO and PPO are initialized from the same $\pi_{0.5}$ few-shot policy, obtained with limited downstream demonstrations. Starting from this backbone, OGPO improves the average success rate from 77.1% to 97.1%, outperforming the $\pi_{0.5}$ full-SFT baseline of 96.85% and achieving performance comparable to $\pi_{0.5}$+PPO, an online RL method requiring costly online rollout. The improvement is particularly significant on the long-horizon **LIBERO-Long** suite, where our method raises the success rate from 43.9% to 93.2%, also surpassing the full-SFT result of 92.4%, indicating its enhanced goal-reaching capability in long-horizon manipulation tasks. On MetaWorld-MT50, our method consistently outperforms SFT across different backbones, improving π_0 from 50.8% to 81.0% and approaching the online PPO-based result of 85.8%. The gains are especially large on more challenging tasks; for example, on the π_0 backbone, our method improves the success rate by 28.4% on Hard tasks and 53.6% on **Very Hard** tasks. Overall, these results validate that OGPO can effectively extract goal-reaching supervision from offline trajectories, enabling substantial policy improvement without online rollouts, particularly on long-horizon and difficult manipulation tasks.

Furthermore, to evaluate the robustness of OGPO and its ability to handle off-demonstration deviations, Table 3 reports performance under selected LIBERO-Plus perturbations and our proposed **LIBERO-ReAct**, a mid-execution perturbation benchmark built on LIBERO [24]. Among the perturbations provided by LIBERO-Plus, we select four representative types: camera viewpoint changes, robot initial-state variations, visual noise, and object layout changes, covering scene-, observation-, and initial-state distribution shifts. To further test dynamic recovery during policy

Table 2: Performance comparison on the MetaWorld-MT50 benchmark. Policy input notation: T (third-view image), P (proprioception), C (task conditioning). C denotes the task information used to distinguish MetaWorld tasks, and may be instantiated as a language instruction, task ID, or task embedding depending on the model.

Model	Input	MetaWorld-MT50				
		Easy	Medium	Hard	Very Hard	Avg.
Diffusion Policy[9]	T+C	23.1	10.7	1.9	6.1	10.5
TinyVLA[41]	T+P+C	77.6	21.5	11.4	15.8	31.6
SmolVLA[33]	T+P+C	87.1	51.8	70.0	64.0	68.2
π_0-PPO (RLinf)[6]	T+P+I	91.1	81.8	78.3	92.0	85.8
π_0-SFT	T+P+C	77.9	51.8	53.3	20.0	50.8
π_0-OGPO	T+P+C	89.2	79.6	81.7	73.6	81.0
		↑11.3	↑27.8	↑28.4	↑53.6	↑30.2
$\pi_{0.5}$-SFT	T+P+C	68.2	37.3	41.7	28.0	43.8
$\pi_{0.5}$-OGPO	T+P+C	82.1	55.8	71.7	59.0	67.2
		↑13.9	↑18.5	↑30.0	↑31.0	↑23.4

Table 3: Robustness Evaluation on the LIBERO-Plus Benchmark and Our Extended Perturbation Settings. The red-highlighted columns indicate the additional mid-episode perturbations introduced by us. All results are evaluated without perturbation-specific fine-tuning.

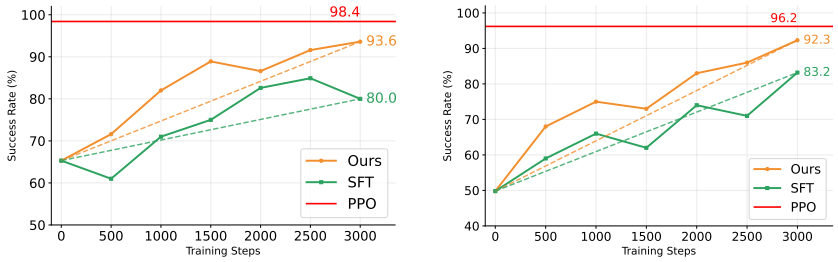
Model	Perturbation Dimensions							Total
	Camera	Robot-Init	Noise	Layout	Action	Gripper	End-effector	
π_0[3]	13.8	6.0	79.0	68.9	81.5	9.3	62.4	45.8
π_0+fast[30]	65.1	21.6	74.4	68.8	75.2	19.8	63.8	55.5
OpenVLA[19]	0.8	3.5	15.2	28.5	19.7	24.3	31.5	17.6
OpenVLA-OFT[18]	56.4	31.9	75.8	74.2	85.2	84.4	87.6	70.8
$\pi_{0.5}$-SFT[2]	62.8	69.8	69.5	85.4	86.6	76.8	86.0	76.7
$\pi_{0.5}$-OGPO	61.5	76.6	73.1	86.6	96.2	87.7	90.1	81.7
	↓1.3	↑6.8	↑3.6	↑1.2	↑9.6	↑10.9	↑4.1	↑5.0

execution, LIBERO-ReAct introduces three mid-episode action-space perturbations across the four LIBERO task suites, including Spatial, Object, Goal, and Long. For each task, we randomly select a 20-step interval during the episode and apply the perturbation after the policy predicts an action and before the action is executed in the environment:

- **Action noise**: we add uniform noise to the executed action command, with a range of $[-0.1, 0.1]$ for arm control and $[-0.5, 0.5]$ for gripper control.
- **Gripper-state perturbation**: we flip the gripper command to the opposite state, forcing an open command to close and a close command to open.
- **End-effector position perturbation**: we add uniform noise within $[-0.2, 0.2]$ to the predicted position motion command, simulating spatial deviations of the end effector.

Compared with the full-shot SFT baseline, our offline fine-tuned $\pi_{0.5}$ improves the overall robustness score from 76.7% to 81.7%. It achieves clear gains on LIBERO-Plus Robot-Init and Noise perturbations, and shows even larger improvements under LIBERO-ReAct disturbances. In particular, the success rate under gripper-state perturbation increases from 76.8% to 87.7%, suggesting that our method better recovers from erroneous grasping or releasing actions. These results indicate that our method improves both robustness to static distribution shifts and recovery from execution-time disturbances.

At last, to evaluate the scalability of OGPO, we conduct an additional set of experiments using π_0 as the backbone. As shown in Fig. 3, OGPO also achieves substantial improvements over standard SFT on LIBERO-Spatial and LIBERO-Goal under the same 3,000 training steps. Specifically, our method reaches 93.6% on LIBERO-Spatial and 92.3% on LIBERO-Goal, outperforming SFT by 13.6 and 9.1 points, respectively. These results further validate the effectiveness of OGPO, suggesting that the



(a) LIBERO-Spatial (b) LIBERO-Goal

Figure 3: Training efficiency comparison between Ours and SFT using the π_0 backbone.

Table 4: Ablation of the process-reward design. Results are success rates (%). SFT is included as a reference. Best results are in bold.

Method	LIBERO-Spatial	LIBERO-Goal	MetaWorld-Hard
SFT (reference)	80.0	83.2	53.3
Outcome-only reward	82.4	85.3	73.4
Without segmentation	71.8	75.2	69.5
Random segmentation	73.2	78.2	72.4
w/o λ_{step}	88.9	85.9	71.6
w/o R^{prog}	87.0	82.0	77.5
w/o R^{goal}	81.0	78.5	73.2
Full OGPO	**93.6**	**92.3**	**81.7**

proposed offline objective provides more effective learning signals than pure supervised imitation and leads to more efficient policy improvement.

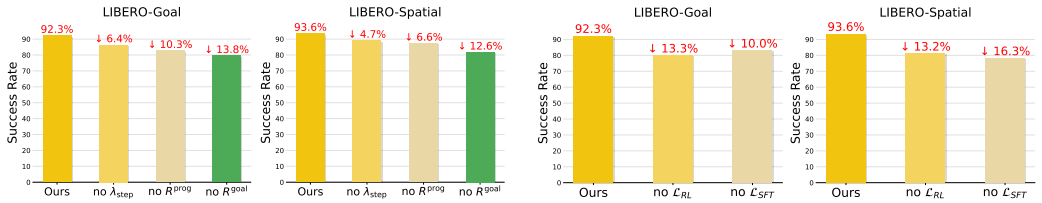
4.3 Ablation Study

To understand the contributions of different components in OGPO, we conduct ablation studies from three perspectives: reward design and trajectory segmentation, policy optimization objectives, and the utilization of failed trajectories.

Ablation on reward design. As shown in Fig. 4(a), removing R^{goal} causes the largest performance degradation among individual reward-term ablations on LIBERO-Spatial and LIBERO-Goal. We further examine the role of segment-level supervision in Table 4. Compared with trajectory-level outcome rewards, OGPO improves success rates by 11.2, 7.0, and 8.3 percentage points on LIBERO-Spatial, LIBERO-Goal, and MetaWorld-Hard, respectively. Removing or randomizing segment boundaries also substantially degrades performance, highlighting the importance of task-relevant segmentation. These findings suggest that meaningful segmentation and segment-level completion rewards are key contributors to OGPO's gains, providing informative supervision for value learning and advantage-weighted policy optimization. Detailed experimental settings and further analysis are provided in Appendix C.

Ablation on policy optimization objectives. As shown in Fig. 4(b), the full objective achieves the best performance on both LIBERO tasks. Removing offline RL reduces performance, demonstrating the importance of advantage-weighted policy improvement in emphasizing actions with higher goal-directed returns. Removing SFT also degrades performance, indicating that reward-guided optimization benefits from regularization toward feasible demonstrated behaviors. These results highlight their complementary roles: SFT preserves reliable action priors and stabilizes learning, while offline RL leverages segment-level rewards to favor actions that advance task progress.

Ablation on failed trajectories. We further investigate how failed trajectories affect policy optimization. As shown in Table 5, SFT slightly outperforms OGPO when trained exclusively on



(a) Reward design. (b) Optimization objectives.

Figure 4: Ablation studies using the π_0 backbone trained for 3,000 steps on LIBERO-Spatial and LIBERO-Goal.

Table 5: Effect of failed-trajectory ratio on the average success rate (%) across four LIBERO suites. Successful demonstrations are fixed, while failed trajectories are progressively added. Both methods use the same available trajectories at each ratio.

Failed/Successful Ratio	$\pi_{0.5}$-SFT	$\pi_{0.5}$-OGPO
0%	**96.85**	95.15
10%	90.65	**95.60**
23.6% (full dataset)	86.35	**97.10**

successful demonstrations. However, as the failed-to-successful ratio increases to 23.6%, SFT performance decreases from 96.85% to 86.35%, whereas OGPO improves from 95.15% to 97.10%. This contrast suggests that OGPO can better utilize mixed-quality offline data through segment-level completion supervision and advantage weighting, which emphasize successful intermediate progress while reducing the influence of low-return actions. Thus, failed trajectories can provide complementary learning signals beyond successful demonstrations, rather than serving merely as undesirable behaviors to imitate.

5 Conclusion and Limitations

In this work, we investigate how to leverage offline data to train a robust VLA model capable of recovering toward goal states after deviating from demonstration trajectories. Specifically, we propose OGPO, an offline goal-conditioned policy optimization framework for recoverable VLA post-training. Instead of treating demonstrations as step-wise action supervision, OGPO relabels offline trajectories into key-state-centered goal-reaching segments and transforms segment completion signals into goal-conditioned process rewards. These rewards facilitate value learning and advantage-weighted policy optimization. In addition, OGPO retains SFT to provide a coarse-grained behavioral prior that anchors the policy to the feasible action distribution induced by expert demonstrations. Overall, OGPO shifts VLA post-training from trajectory-level action imitation toward key-state reachability learning, thereby improving goal-directed recoverability without requiring online interaction. Extensive experiments demonstrate the effectiveness, robustness, and recovery capability of OGPO under off-demonstration scenarios. Despite the promising results, this work still has several limitations. First, the current gripper-state-based segmentation strategy may not generalize well to tasks whose progress cannot be naturally characterized by such events. Second, as a purely offline framework, OGPO is limited by the quality of the offline dataset, which may restrict its ability to recover from severe out-of-distribution deviations beyond the demonstrated trajectories. In future work, we plan to investigate more general and robust goal-state relabeling strategies, along with data generation and augmentation methods that can improve the diversity of offline data for recoverable policy learning.

Acknowledgments and Disclosure of Funding

Funding. This work was supported by the National Key Laboratory of Space Integrated Information System, Institute of Software, Chinese Academy of Sciences, under Project No. E5GF5909B1.

Competing Interests. The authors declare no competing interests.

References

[1] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[2] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023.

[5] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Worldvla: Towards autoregressive action world model, 2025.

[6] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2026.

[7] Ze Chen, Runliang Niu, He Kong, and Qi Wang. Tgrpo :fine-tuning vision-language-action model via trajectory-wise group relative policy optimization. *ArXiv*, abs/2506.08440, 2025.

[8] Zixuan Chen, Junhui Yin, Yangtao Chen, Jing Huo, Pinzhuo Tian, Jieqi Shi, Yiwen Hou, Yinchuan Li, and Yang Gao. Deco: Task decomposition and skill composition for zero-shot generalization in long-horizon 3d manipulation, 2026.

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[10] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models, 2025.

[11] Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. SRFT: A single-stage method with supervised and reinforcement fine-tuning for reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[12] Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15665–15672. IEEE, 2025.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[14] Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning, 2025.

[15] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. Nora: A small open-sourced generalist vision language action model for embodied tasks, 2025.

[16] Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, Yu Wang, Haoran Li, Chao Yu, and Dongbin Zhao. Wovr: World models as reliable simulators for post-training vla policies with rl, 2026.

[17] Tobias Jülg, Wolfram Burgard, and Florian Walter. Refined policy distillation: From vla generalists to rl experts. *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11677–11684, 2025.

[18] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025.

[19] 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 2025.

[20] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

[21] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025.

[22] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[23] Hengtao Li, Pengxiang Ding, Runze Suo, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. VLA-RFT: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators, 2026.

[24] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[25] Jijia Liu, Feng Gao, Bingwen Wei, Xin-Lei Chen, Qingmin Liao, Yi Wu, Chaoyang Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *ArXiv*, abs/2505.19789, 2025.

[26] Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. Uft: Unifying supervised and reinforcement fine-tuning. *arXiv preprint arXiv:2505.16984*, 2025.

[27] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[29] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[30] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models, 2025.

[31] 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.

[32] John D. Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[33] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andrés Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Rémi Cadène. Smolvla: A vision-language-action model for affordable and efficient robotics. *ArXiv*, abs/2506.01844, 2025.

[34] Chuxiong Sun, Peng He, Qirui Ji, Zehua Zang, Jiangmeng Li, Rui Wang, and Wei Wang. M2i2: Learning efficient multi-agent communication via masked state modeling and intention inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 25672–25681, 2026.

[35] Chuxiong Sun, Peng He, Rui Wang, and Changwen Zheng. Revisiting communication efficiency in multi-agent reinforcement learning from the dimensional analysis perspective. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '25, page 1977–1986, Richland, SC, 2025. International Foundation for Autonomous Agents and Multiagent Systems.

[36] Chuxiong Sun, Bo Wu, Rui Wang, Xiaohui Hu, Xiaoya Yang, and Cong Cong. Intrinsic motivated multi-agent communication. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1668–1670, 2021.

[37] Chuxiong Sun, Dunqi Yao, Rui Wang, Wenwen Qiang, Changwen Zheng, and Jiangmeng Li. Tmae: Learning targeted multi-agent exploration via causal inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 25682–25690, 2026.

[38] Chuxiong Sun, Zehua Zang, Jiabao Li, Jiangmeng Li, Xiao Xu, Rui Wang, and Changwen Zheng. T2mac: Targeted and trusted multi-agent communication through selective engagement and evidence-driven integration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15154–15163, 2024.

[39] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[40] Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026.

[41] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10:3988–3995, 2024.

[42] Junjin Xiao, Yandan Yang, Xinyuan Chang, Ronghan Chen, Feng Xiong, Mu Xu, Wei-Shi Zheng, and Qing Zhang. World-env: Leveraging world model as a virtual environment for VLA post-training, 2025.

[43] Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. In *The Fourteenth International Conference on Learning Representations*, 2026.

[44] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[45] Zehua Zang, Xi Wang, Fuchun Sun, Xiao Xu, Lixiang Liu, Jiahuan Zhou, and Jiangmeng Li. Test-time perturbation tuning with delayed feedback for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8110–8119, June 2026.

[46] Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. In *The Fourteenth International Conference on Learning Representations*, 2026.

[47] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025.

[48] Fangqi Zhu, YAN Zhengyang, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. WMPO: World model-based policy optimization for vision-language-action models. In *The Fourteenth International Conference on Learning Representations*, 2026.

[49] Yihang Zhu, Weiqing Wang, Shijie Wu, Ye Shi, and Jingya Wang. Learning semantic atomic skills for multi-task robotic manipulation. *arXiv preprint arXiv:2512.18368*, 2025.

[50] 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.

A Pseudo-code of OGPO

Algorithm 1 summarizes the training procedure of OGPO. The relabeled goals are only used to construct process rewards during training, and are not introduced as additional inputs to the policy or value function.

Algorithm 1 OGPO: Offline Goal-conditioned Policy Optimization

Require: Successful demonstrations $\mathcal{D}_{\text{demo}}$, failed trajectories $\mathcal{D}_{\text{fail}}$, pretrained VLA policy π_θ, value function V_ψ, VLM annotator
Ensure: Optimized policy π_θ

Initialize relabeled dataset $\widetilde{\mathcal{D}} \leftarrow \emptyset$
for each trajectory $\tau \in \mathcal{D}_{\text{demo}} \cup \mathcal{D}_{\text{fail}}$ **do**
    Segment τ into stages $\{\tau^1, \ldots, \tau^M\}$ using gripper-state changes
    for each segment τ^m **do**
        if $\tau \in \mathcal{D}_{\text{demo}}$ **then**
            Set the segment terminal state as goal g^m and label $y^m \leftarrow 1$
        else
            Assign a candidate local goal g^m and use the VLM annotator to infer its completion label y^m
        end if
        Add goal labels to transitions in τ^m
    end for
end for
for each relabeled segment τ^m **do**
    Compute progress variable ρ_t^m using Eq. 3
    Annotate process rewards r_t using Eq. 4
    Add reward-augmented transitions to $\widetilde{\mathcal{D}}$
end for
Value learning: Train V_ψ on $\widetilde{\mathcal{D}}$ with the n-step target in Eq. 6 by minimizing $\mathcal{L}_V$ in Eq. 7
Policy optimization:
while not converged **do**
    Sample a relabeled minibatch from $\widetilde{\mathcal{D}}$
    Compute advantages A_t and AWAC weights ω_t using the trained value function V_ψ
    Compute $\mathcal{L}_{\text{RL}}$ in Eq. 10 and $\mathcal{L}_{\text{SFT}}$ in Eq. 11
    Update π_θ by minimizing $\mathcal{L}_{\text{total}}$ in Eq. 12
end while
return π_θ

B Detailed Trajectory Segmentation and Goal Relabeling

This section provides the implementation details of our trajectory segmentation and goal relabeling procedure. The key idea is to exploit the gripper open-close pattern in manipulation trajectories. In many tabletop manipulation tasks, gripper state changes naturally correspond to important semantic transitions, such as approaching an object, grasping it, transporting it, and releasing it at the target location. Therefore, we use gripper state changes as temporal anchors to divide a long trajectory into several goal-level segments. After segmentation, each segment is assigned a semantic goal label and a binary completion label.

B.1 Gripper-Based Trajectory Segmentation

Given an offline trajectory

$$\tau = \{(o_t, l, a_t)\}_{t=1}^{T},$$

where o_t is the observation, l is the language instruction, and a_t is the action, we extract the gripper command from the action sequence:

$$c_t \in \{\text{open}, \text{close}\}.$$

We then identify gripper switching points as candidate segment boundaries:

$$\mathcal{B} = \{t \mid c_t \neq c_{t-1}\}.$$

To avoid noisy boundaries caused by short-term command oscillations, we only keep a switching point if the new gripper state is maintained for a short temporal window. The final boundary set is formed by adding the beginning and the end of the trajectory:

$$\bar{\mathcal{B}} = \{1\} \cup \mathcal{B} \cup \{T\}.$$

The trajectory is then divided into a sequence of segments:

$$\tau = \{\tau^1, \tau^2, \ldots, \tau^M\}, \qquad \tau^m = \{(o_t, l, a_t)\}_{t=s_m}^{e_m},$$

where s_m and e_m are two adjacent boundaries in $\bar{\mathcal{B}}$.

This gripper-based segmentation is simple but effective for manipulation tasks because most meaningful object interactions are synchronized with gripper events. For example, an open-to-close transition near an object usually indicates a grasping attempt, while a close-to-open transition near a target region usually indicates a placing or releasing attempt. The segment between these two events often corresponds to a transportation or alignment goal.

B.2 Goal Relabel

After obtaining the temporal segments, we assign a semantic goal g^m to each segment τ^m. The goal label is determined by the task instruction, the gripper transition type, and the visual state change within the segment. For example, for a task instruction such as "put the black bowl in the bottom drawer of the cabinet and close it", the trajectory can be decomposed into the following goals:

$$\text{reach} \rightarrow \text{grasp} \rightarrow \text{move} \rightarrow \text{lay} \rightarrow \text{close}.$$

In practice, we use the following rules to assign goal labels:

- If the segment ends with an open-to-close gripper transition near a target object, it is labeled as a grasping-related goal.
- If the gripper remains closed and the object pose changes with the end-effector, the segment is labeled as a moving or transporting goal.
- If the segment ends with a close-to-open gripper transition near the target region, it is labeled as a placing or releasing goal.
- For articulated-object tasks, if the gripper remains closed while the joint state or object part pose changes, the segment is labeled as an opening, pulling, pushing, or closing goal.

This procedure converts each original trajectory into a sequence of goal-conditioned segments:

$$\tau \rightarrow \{(\tau^1, g^1), (\tau^2, g^2), \ldots, (\tau^M, g^M)\}.$$

B.3 Completion Labeling

For each segment τ^m, we further assign a binary completion label:

$$y^m \in \{0, 1\},$$

where $y^m = 1$ indicates that the segment successfully completes its assigned goal g^m, and $y^m = 0$ indicates that the segment fails to complete the goal.

Successful trajectories. For a trajectory that successfully completes the full task, we label all its valid goal segments as successful:

$$y^m = 1, \qquad \forall m.$$

Failed trajectories. To better localize the failure stage and provide more accurate supervision for value learning, we use a VLM-based offline annotator to determine whether each segment in a failed trajectory completes its assigned goal. The VLM is asked to judge whether the visual state at the end of the segment satisfies the assigned goal. The resulting relabeled dataset is:

$$\widetilde{\mathcal{D}} = \{(o_t, l, a_t, g_t, y_t)\},$$

where g_t and y_t are inherited from the segment containing time step t.

VLM Annotation Prompt

Your task is to determine whether a trajectory segment completes a given candidate subgoal.
Task instruction: {instruction}
Candidate goal: {goal}
Segment observation: The following images show the first, middle, and last frames of a trajectory segment.
Question: Does this segment successfully complete the candidate goal?
Please answer with one of the following labels:
`SUCCESS`: the final frame satisfies the candidate goal.
`FAILURE`: the final frame does not satisfy the candidate goal.
Also provide a short reason.

Figure 5: Prompt template for VLM-based annotation of failed trajectory segments. We use Qwen2.5-VL-72B-Instruct as the VLM annotator.

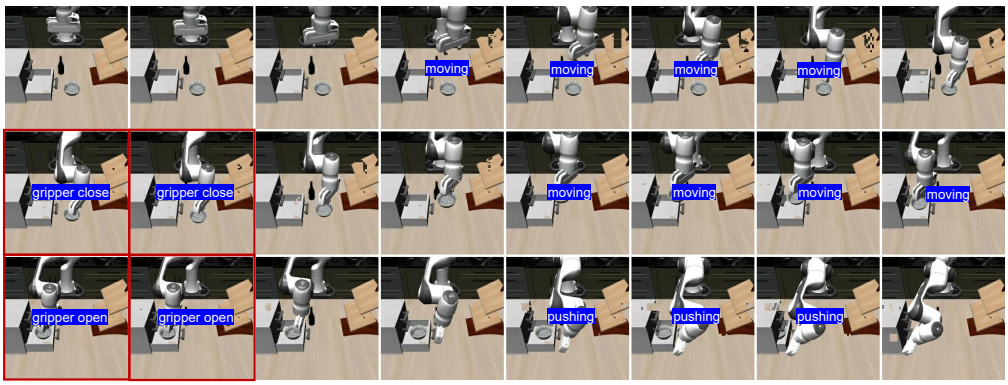


Figure 6: Annotated example of a successful trajectory.

B.4 VLM Annotation Prompt for Failed Segments

For failed trajectories, we use Qwen2.5-VL-72B-Instruct as the VLM annotator to evaluate whether each segment completes its candidate goal. As shown in Figure 5, the prompt provides the task instruction, candidate goal, and representative segment observations, and asks the annotator to output a completion label with a short explanation.

B.5 Annotated Cases

Figure 6 and Figure 7 show two annotated examples. For visualization clarity, we only show frames sampled every 10 time steps as a coarse illustration. The first case is a successful trajectory. After gripper-based segmentation, all goal segments are labeled as achieved. The second case is a failed trajectory. Although the full trajectory fails, the early segments still complete meaningful goals, while the later segment fails to complete the assigned goal. Our VLM-based annotation preserves the achieved segments and marks the failed segment as negative experience.

C Reward Design and Value Analysis

This section provides additional details and analyses of the process-reward design used by OGPO. We first restate the reward formulation for completeness, then examine the effects of its individual components and the trajectory segmentation strategy, and finally visualize the learned value estimates along representative trajectories.

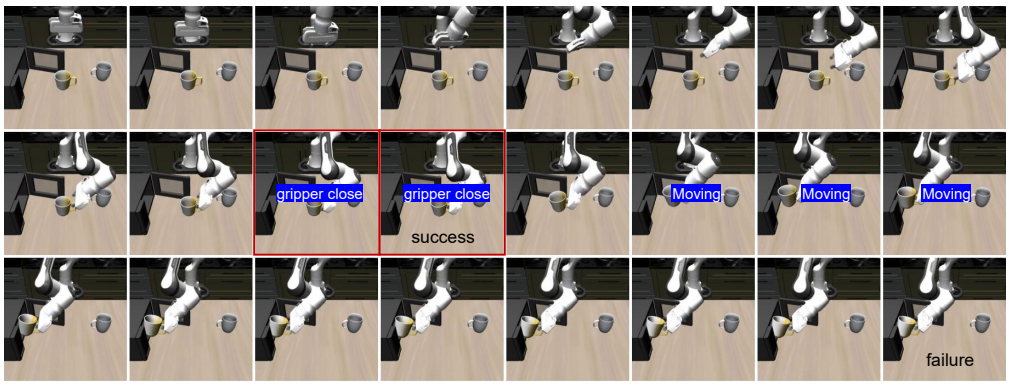


Figure 7: Annotated example of a failed trajectory.

C.1 Reward formulation.

For each relabeled segment τ^m, let

$$y^m = \mathbb{I}[\tau^m \text{ achieves } g^m]$$

indicate whether its assigned local goal is completed. As defined in Eq. 3, when $y^m = 1$, the end observation o_{e_m} represents the achieved task-progress key state corresponding to g^m, and ρ_t^m increases from 0 to 1 along the segment. When $y^m = 0$, ρ_t^m remains zero.

For each transition $t \in \{s_m, \ldots, e_m\}$, we define $\delta_t^m = \mathbb{I}[t = e_m]$ and annotate the process reward as

$$r_t(g^m) = -\lambda_{\text{step}} + R^{\text{prog}}\left(\rho_{t+1}^m - \rho_t^m\right) + \delta_t^m\left(R_{\text{success}}^{\text{goal}} y^m - R_{\text{fail}}^{\text{goal}}(1 - y^m)\right). \tag{13}$$

The reward is conditioned on g^m through the goal-specific completion label y^m, rather than by providing g^m as an additional input to the policy or value function. The step penalty discourages inefficient action sequences. For a completed segment, the progress term distributes positive credit across its transitions; for an uncompleted segment, this term remains zero. The final term assigns a segment-end bonus or penalty according to local goal completion.

Unless otherwise specified, we use $\lambda_{\text{step}} = 0.01$, $R^{\text{prog}} = 1.0$, $R_{\text{success}}^{\text{goal}} = 1.0$, and $R_{\text{fail}}^{\text{goal}} = 2.0$ in all experiments.

C.2 Reward-design ablations.

We evaluate both the individual reward terms and the trajectory structure used for reward construction. Specifically, we consider the following variants:

- **Outcome-only reward.** This variant replaces Eq. 13 with a single trajectory-level terminal reward, where $r_T = 1$ for a successful trajectory and $r_T = 0$ otherwise. All preceding transitions receive zero reward, and no local-goal labels or within-segment temporal credit are used.
- **Without segmentation.** This variant treats the entire trajectory as a single goal-reaching segment and applies the same reward formulation over the full trajectory. It therefore retains completion-gated temporal credit but does not model intermediate manipulation stages.
- **Random segmentation.** This variant replaces the gripper-based boundaries with randomly sampled boundaries while keeping the number of segments and all other reward settings unchanged.
- **Reward-term ablations.** These variants individually remove the step penalty λ_{step}, the temporal progress term R^{prog}, or the local goal-completion term R^{goal}, while retaining the full segmentation and optimization pipeline, as shown in Fig. 4(a).

All variants use the π_0 backbone and the same offline data and training configuration.

Table 6: Ablation of the process-reward design. Results are success rates (%). SFT is included as a reference. The best result in each setting is shown in bold.

Method	LIBERO-Spatial	LIBERO-Goal	MetaWorld-Hard
SFT (reference)	80.0	83.2	53.3
Outcome-only reward	82.4	85.3	73.4
Without segmentation	71.8	75.2	69.5
Random segmentation	73.2	78.2	72.4
w/o λ_{step}	88.9	85.9	71.6
w/o R^{prog}	87.0	82.0	77.5
w/o R^{goal}	81.0	78.5	73.2
Full OGPO	**93.6**	**92.3**	**81.7**

As shown in Table 6, full OGPO outperforms the outcome-only baseline by 11.2, 7.0, and 8.3 percentage points on LIBERO-Spatial, LIBERO-Goal, and MetaWorld-Hard, respectively. This shows that the improvement does not come merely from applying offline policy optimization with task-level outcomes; instead, local-goal completion rewards provide more fine-grained supervision for value learning and policy optimization.

Both the no-segmentation and random-segmentation variants substantially underperform full OGPO and fall below the outcome-only baseline. This suggests that missing or incorrect segment boundaries can introduce misleading local rewards and impair value learning, demonstrating the importance of task-relevant trajectory segmentation. Together with the reward-term ablations, these results show that the effectiveness of OGPO arises from the combination of meaningful segmentation, local-goal relabeling, and dense progress supervision.

C.3 Value curve analysis.

To examine how the annotated rewards shape value learning, we visualize $V_\psi(o_t, l)$ along representative execution trajectories. As shown in Fig. 8, the learned value curves exhibit clear local high-value regions along the execution trajectory. Interestingly, these high-value turning points are well aligned with semantically meaningful task-progress key states, such as approaching the target object, establishing a stable grasp, opening the drawer, or placing the object at the goal location. These patterns suggest that the segment-level completion rewards are propagated through value learning, allowing the value function to reflect intermediate task structure rather than responding only to the final task outcome. The resulting return estimates provide more localized signals for advantage-weighted offline policy optimization.

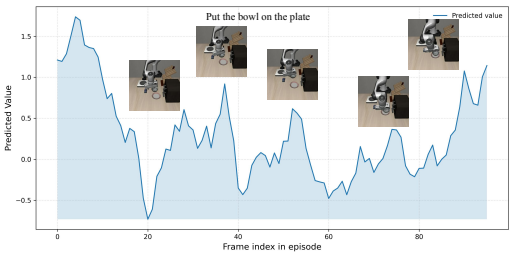
D Hyperparameter Analysis

In this section, we analyze the sensitivity of our method to the objective weights in Eq. 12. The final policy objective contains two terms: supervised fine-tuning on achieved goal segments and advantage-guided offline reinforcement fine-tuning. Their relative importance is controlled by λ_{SFT}, λ_{RL} respectively:

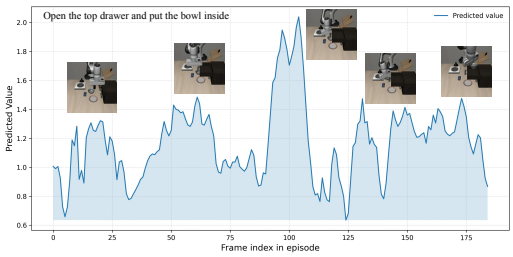
$$\mathcal{L}_{\text{total}}(\theta) = \lambda_{\text{SFT}}\mathcal{L}_{\text{SFT}}(\theta) + \lambda_{\text{RL}}\mathcal{L}_{\text{RL}}(\theta).$$

Unless otherwise specified, we use $\lambda_{\text{SFT}} = 0.4, \lambda_{\text{RL}} = 0.6$.

Discussion. The two terms play complementary roles. The SFT term provides a stable imitation anchor from achieved goal segments. The offline RL term further distinguishes actions according to their estimated long-term contribution to goal completion. Empirically, we find that using two components yields more stable and effective policy improvement than relying on any single term alone.



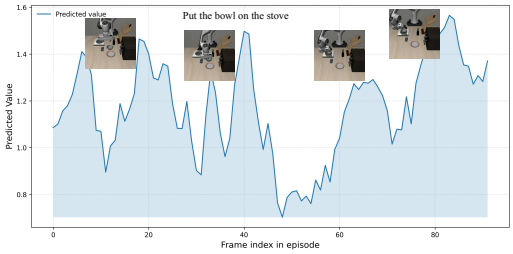
(a) Put the bowl on the plate.



(b) Open the drawer and put the bowl inside.



(c) Push the plate to the front of the stove.



(d) Put the bowl on the stove.

Figure 8: Visualization of the learned goal-directed value along execution trajectories. Local high-value frames are aligned with semantically meaningful task-progress key states, such as placing the bowl on the plate, opening the drawer, or moving the plate toward the stove. This qualitative evidence suggests that the learned value provides dense progress signals beyond final task success.

Table 7: Sensitivity analysis of the objective weights. The default setting is highlighted in gray.

λ_{SFT}	λ_{RL}	LIBERO-Goal
0.1	0.9	79.6
0.2	0.8	83.2
0.3	0.7	86.8
0.4	0.6	92.3
0.5	0.5	89.2
0.6	0.4	82.5
0.7	0.3	82.8

E Implementation Details

E.1 Dataset Composition

This section describes the collection and composition of the offline training data. For both LIBERO and MetaWorld-MT50, the successful trajectories are obtained from the official demonstration datasets. To collect failed trajectories, we roll out the corresponding few-shot SFT checkpoint on π_0 and retain unsuccessful episodes. We collect 10 failed trajectories for each task and process them using the same trajectory format as the successful demonstrations. The resulting data composition after preprocessing is summarized in Table 8.

Note: The failed trajectories are ordinary unsuccessful rollouts rather than trajectories specifically collected to demonstrate recovery behaviors. The LIBERO-ReAct benchmark is used exclusively for out of distribution(OOD) evaluation. Therefore, the recovery results on LIBERO-ReAct measure generalization to previously unseen execution-time perturbations rather than training on perturbation-specific recovery trajectories.

Table 8: Composition of the offline datasets used by OGPO. The ratio is calculated between successful and failed trajectories.

Task Suite	Successful	Failed	Ratio
LIBERO-Spatial	432	100	4.32:1
LIBERO-Object	454	100	4.54:1
LIBERO-Goal	428	100	4.28:1
LIBERO-Long	379	100	3.79:1
LIBERO Total	1,693	400	4.23:1
MetaWorld-Easy	1,400	280	5:1
MetaWorld-Medium	550	110	5:1
MetaWorld-Hard	300	60	5:1
MetaWorld-Very Hard	250	50	5:1
MetaWorld Total	2,500	500	5:1

Table 9: Value-head architectures. Each sequence includes the pooled input representation, hidden layers, and scalar output.

Backbone	MLP Dimensions
π_0	$1024 \to 512 \to 256 \to 128 \to 1$
$\pi_{0.5}$	$2048 \to 1024 \to 512 \to 256 \to 1$

E.2 Value-Head Architecture

The value function reuses the visual-language representations of the pretrained VLA backbone. Given the final-layer token representations $H_t = \{h_{t,i}\}_{i=1}^{N}$ for observation o_t and instruction l, we first apply masked mean pooling over the valid image-language tokens:

$$\bar{h}_t = \frac{\sum_{i=1}^{N} m_i h_{t,i}}{\sum_{i=1}^{N} m_i}, \tag{14}$$

where $m_i \in \{0, 1\}$ indicates whether token i is valid. The pooled representation is then passed through a multilayer perceptron that produces a scalar value estimate:

$$V_\psi(o_t, l) = \mathrm{MLP}_\psi(\bar{h}_t). \tag{15}$$

The value-head architectures used for the two VLA backbones are summarized in Table 9. Each hidden layer is followed by a ReLU activation, while the scalar output layer uses no activation.

During value learning, the pretrained VLA backbone is frozen and only the MLP value head is optimized using the n-step temporal-difference objective in Eq. 7. After training, the entire value model is kept fixed and used only to compute advantage weights for offline policy optimization. The value model is discarded after post-training and introduces no additional parameters or computation during policy inference.

E.3 Training Configuration

We implement our offline fine-tuning method within the RLinf [6] framework. For fair comparison, our implementation follows the same preprocessing, action normalization, backbone initialization, and optimization setup as the corresponding SFT baseline. The value model is trained before policy fine-tuning and is only used during offline training to compute advantage weights.

F Computational Cost Analysis

We compare the computational cost of our offline fine-tuning method with online RL-based post-training. Unlike online RL methods, our method does not require policy rollout collection during training. All reward annotation, value learning, and policy optimization are performed on the fixed

Table 10: Optimization hyperparameters.

Hyperparameter	**LIBERO**	**MetaWorld**
Optimizer	AdamW	AdamW
Policy learning rate	2.5×10^{-5}	2.5×10^{-5}
Value learning rate	1.55×10^{-4}	1.55×10^{-4}
Adam β_1	0.9	0.9
Adam β_2	0.95	0.95
Adam ϵ	1.0×10^{-8}	1.0×10^{-8}
Weight decay	1.0×10^{-10}	1.0×10^{-10}
Gradient clipping	1.0	1.0
LR schedule	Cosine warmup	Cosine warmup
Warmup steps	1,000	1,000
Policy Training Steps	30,000	10,000
LoRA	Yes	Yes
LoRA rank	32	32

Table 11: Value model training settings.

Setting	**LIBERO**	**MetaWorld**
Value loss	TD	TD
Value training epochs	5	5
Steps per epoch	600	1000
Input images	2	1
Value feature	After VLM	After VLM
Pooling mode	Mean token	Mean token
Freeze VLM	Yes	Yes
Strict checkpoint loading	Yes	Yes
Used at inference	No	No

offline dataset. The comparison shows that our offline fine-tuning method is substantially more computationally efficient than online RL-based training. More importantly, our method avoids online environment interaction, making it easier to scale to multiple task suites and safer for robotic manipulation settings where online rollout collection can be expensive or risky.

Table 13: Computational cost comparison on LIBERO. GRPO denotes the online RL training recipe in RLinf. The GRPO cost is reported for one LIBERO task suite, while our method is reported for all four LIBERO suites.

Method	**Online Rollouts**	**Hardware**	**Task Suites**	**Wall-clock Time**
RLinf-GRPO	Yes	8×A100	1 LIBERO suite	∼175 hours
Ours	No	4×RTX 4090 48GB	4 LIBERO suites	∼68 hours

Table 12: Offline objective hyperparameters.

Hyperparameter	**LIBERO**	**MetaWorld**
AWAC temperature β	1.0	1.0
Maximum advantage weight	20.0	20.0
Normalize advantage	Yes	Yes
Advantage clipping	3.0	3.0
External reward model	No	No
Online critic model	No	No
Online rollouts	No	No

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state that the paper proposes an offline goal-conditioned policy optimization framework for improving recoverability in VLA policies. The claims are supported by experiments on LIBERO, MetaWorld, LIBERO-Plus, and the proposed mid-execution perturbation benchmark.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses several limitations in the Conclusion and Limitations section, including the reliance on gripper-state changes for trajectory segmentation, possible noise in VLM-based goal annotation under ambiguous observations, and the dependence of the purely offline method on the coverage of the offline dataset.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results, theorems, or proofs. The main contribution is an algorithmic framework with empirical validation.

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: The paper describes the training data construction, task suites, backbone policy, optimization objectives, perturbation settings, evaluation protocols, and implementation details needed to reproduce the main experimental 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: We have provided open access to the data and code in supplemental material.

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: The paper specifies the benchmark suites, data usage, evaluation metrics, training objectives, hyperparameters, optimizer settings, and perturbation configurations used in the experiments.

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: [No]

   Justification: The paper reports average success rates over multiple tasks and trials, but does not report formal confidence intervals or statistical significance tests. We omit these due to the computational cost of repeated VLA policy training and evaluation.

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 paper provides the GPU type, number of GPUs, training time, and evaluation cost used for the main experiments in appendix.

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: The research is conducted on standard robotic manipulation benchmarks and does not involve human subjects, private data, or deceptive data collection. We follow the NeurIPS Code of Ethics.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The paper discusses the potential positive impact of improving the robustness and recoverability of robot policies, as well as possible risks associated with deploying more autonomous robotic systems in real-world environments.

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 paper does not release high-risk pretrained generative models, scraped datasets, or models intended for dual-use deployment. The experiments are conducted on simulated robotic manipulation benchmarks.

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: The paper cites the original sources of the benchmarks, datasets, model backbones, and codebases used in the experiments, including LIBERO, MetaWorld, LIBERO-Plus, and the corresponding VLA backbone models. Their licenses and usage terms are respected.

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces a mid-execution perturbation evaluation protocol and documents the perturbation types, timing, severity levels, and evaluation procedure. Additional details are provided in the appendix to support reuse.

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: The paper does not involve crowdsourcing experiments or research with human subjects.

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: The paper does not involve human subjects, so IRB approval or an equivalent review is not applicable.

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: We use Qwen2.5-VL-72B-Instruct as an offline vision-language annotator to assess local-goal completion in failed trajectory segments. The annotation results are used to construct goal-conditioned process rewards for policy optimization. The annotator is used only during offline data preparation and introduces no additional computation during policy inference. Further details are provided in Appendix B.4.