COOPERATIVE MULTI-AGENT VISION-LANGUAGE-ACTION MODELS VIA REINFORCED FINE TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study reinforcement learning (RL) methods for cooperative multi-agent Vision-Language-Action (VLA) models. This problem is challenging because VLAs are pretrained on large-scale single-agent data and therefore lack the fine-grained coordination skills required for inter-robot collaboration. Supervised fine-tuning (SFT) on multi-robot demonstrations partially bridges this gap, but its performance is bounded by the demonstration data and cannot improve from its own experience. We present a three-stage reinforced fine-tuning (RFT) pipeline for multi-agent VLAs. First, initialization-aware data collection sweeps over initial configurations and invokes human demonstrations only when the pretrained VLA repeatedly fails, yielding robustness to initialization shift with reduced human cost. Second, offline credit-filtered tuning assigns credit to individual agents and fine-tunes on per-agent trajectories with positive advantage rather than on entire joint rollouts. Third, we find existing online RL for VLAs are less effective for hard multi-agent tasks, which we attribute to noisy co-exploration and unstable updates. We instead use online latent-space fine tuning, which freeze the VLA and perform RL in its latent noise space. We evaluate our multi-agent VLA with both π_0 and $\pi_{0.5}$ backbones across 11 tasks in RoboTwin, RoboFactory and real-world manipulation with two Franka robots. Our multi-agent VLA improves the average success rate by $+23.1\%$, $+16.4\%$, and $+44\%$ on RoboTwin, RoboFactory, and real-world tasks, respectively. Code available at `https://anonymous.4open.science/r/mavla_rft-2BC0/`.

1 INTRODUCTION

Vision-language-action (VLA) models have rapidly emerged as a general interface between visual perception, language understanding, and low-level robotic control, mastering diverse single-agent tasks within a single policy (Team, 2024; Kim et al., 2024; Black et al., 2026; Intelligence, 2025a). However, real-world robot manipulation often require robots to collaborate with others, such as multi-arm assembly (Zhang et al., 2026b) and multi-robot household manipulation Chen et al. (2025b). Despite the advances in single-agent tasks, such performance do not transfer to multi-agent setting automatically, limiting the applicability of VLAs.

Extending VLAs to multi-agent setting is nontrivial. VLAs are typically pre-trained on web-scale single-agent data to acquire general-purpose manipulation skills. However, multi-agent cooperation requires fine-grained low-level coordination skills between robots, which is not covered in the single-agent pretraining dataset. Existing works in vision-language model (VLM) domain focus on decomposing multi-agent task to single-agent plans (Mandi et al., 2023; Liu et al., 2025b; Qin et al., 2025; Tan et al., 2025a; Chen et al., 2025b). However, their low-level execution either relies on rule-based policy or assigned to existing VLAs, overlooking the fine-grained low-level action details required for VLA cooperation. Recently, CHORUS (Doshi et al., 2026) demonstrate the feasibility of multi-VLA cooperation by supervised fine-tuning, yet its performance is inherently bounded by demonstration data and cannot further improve from its own experience.

In this paper, we present a three-stage recipe for reinforced fine-tuning of multi-agent VLAs, consisting of data collection, offline RL, and online RL, which together bridge the gap between single-agent and multi-agent VLAs. First, we propose initialization-aware data collection to ensure that the pretrained single-agent VLA possesses the basic skills required for multi-agent cooperation. The key

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

insight is that subsequent RL training is only possible if the pretrained VLA achieves a non-zero success rate across different environment initializations. For initializations where the VLA succeeds at least occasionally, we collect both successful and failed trajectories for further tuning. For initializations where the VLA never succeeds, we conclude that the single-agent VLA lacks the primitive skills required for that configuration, and we invoke human demonstrations for these cases only, thereby inducing acceptable human effort. Together, this process yields an offline dataset with broad coverage of initial conditions with non-zero success rate, at reasonable human cost.

The second stage uses offline RL to reinforce behaviors from successful trajectories and penalize failure cases. A fundamental challenge here is credit assignment across agents (Rashid et al., 2020). In a multi-agent rollout, overall task failure does not imply that every agent behaved poorly, since one agent may have executed its subtask flawlessly while another caused the failure. Consequently, fine-tuning on entire joint rollouts may credit individual agents improperly, either penalizing desirable individual behavior in failed rollouts or rewarding flawed individual behavior in successful ones. To address this, we propose offline credit-filtered tuning, which estimates a per-agent advantage capturing the contribution of each agent to the team outcome, and fine-tunes the VLA only on per-agent trajectory with positive advantage, ensuring that only desirable behaviors are reinforced.

The third stage further improves the VLA via online RL. We find that existing online RL methods for single-agent flow-matching VLAs (Chen et al., 2026) can be unstable in hard multi-agent tasks with low success rates, a failure we attribute to noisy co-exploration and unstable concurrent updates. First, these methods explore by adding independent noise in raw action space of each agent, which disrupt precise coordination and collapse the learning signal. Second, the intractable marginal likelihood of flow matching forces high-variance surrogate gradients over the denoising chain. While tolerable for a single agent, this variance is amplified by the non-stationarity of concurrent multi-agent updates. To overcome these issues, we instead perform online RL in the latent noise space of the flow-matching policy, using DSRL (Wagenmaker et al., 2025). Rather than modifying VLA weights, we freeze them and induce diverse behaviors by steering the low-dimensional latent noise via online RL, so exploration remains confined to in-distribution behaviors, which yields stable training and further improves performance through trial and error in regime with low success rates.

Together, the three steps gives the first reinforced fine tuning recipe for multi-agent VLAs. We evaluate our framework with both π_0 and $\pi_{0.5}$ backbones across 11 tasks in RoboTwin, RoboFactory and real-world tasks with two Franka robots, covering varying agent numbers from 2 to 4. Our multi-agent VLA improves the average success rate by $+23.1\%$, $+16.4\%$, and $+44\%$ on RoboTwin, RoboFactory, and real-world tasks, respectively. Ablations show each stage is essential for success and visualize its effectiveness against alternative methods, exploring the potential of RL methods in multi-agent VLAs and critical factors that affects its success.

Contributions. Our contributions are two-folded. First, we propose multi-agent VLA, a three-stage reinforced fine-tuning pipeline for multi-VLA cooperation, spanning data collection, offline fine tuning and online fine tuning. Second, our multi-agent VLA achieves consistent enhancement of VLA collaborative across 11 tasks in RoboTwin, RoboFactory and real-world Franka robots.

2 Related Work

Vision-Language-Action Models. VLA models provides a practical way of integraing visual perception, language understanding, and robotic control into a unified policy. Early systems such as RT-1 (Brohan et al., 2023b) and RT-2 (Brohan et al., 2023a) demonstrated large-scale multi-task learning and the transfer of web-scale semantic knowledge to physical actions, while Open X-Embodiment (Collaboration, 2025), Octo (Team, 2024), and OpenVLA (Kim et al., 2024) further advanced cross-embodiment pretraining and open generalist robot policies. Recent works increasingly adopt diffusion or flow-matching action decoders to model multimodal and temporally coherent action sequences, including Diffusion Policy (Chi et al., 2024), RDT-1B (Liu et al., 2025c), π_0 (Black et al., 2026), Diffusion-VLA (Wen et al., 2025a), DexVLA (Wen et al., 2025b), HybridVLA (Liu et al., 2025a), and DiT-A (Hou et al., 2025). Other models further explore efficient action tokenization (Pertsch et al., 2025), open-world generalization (Intelligence, 2025a), spatial reasoning (Lin et al., 2026), world modeling (Cen et al., 2025; Zhang et al., 2025b), and lightweight deployment (Shukor et al., 2025). However, VLAs are pretrained on single-agent datasets, and lack the skills required for multi-agent cooperation, which motivates our paper.

Reinforced Learning for VLAs. Reinforcement learning has recently been introduced to improve the capability of VLA policies beyond supervised imitation, including offline and online methods. Offline RL optimize policy from a fixed collection of demonstrations, typically by learning value functions and using them to guide policy optimization (Nakamoto et al., 2025; Chen et al., 2025c). Human demonstrations are also included to correct VLA from mistakes (Intelligence, 2025b). In contrast, online RL assumes continued interaction with environments and improve policy via trial-and-errors. The problem has been widely studied in VLAs with discrete action bins (Guo et al., 2025; Li et al., 2025; Lu et al., 2025; Tan et al., 2025b; Zhang et al., 2025a; Shu et al., 2025; Chen et al., 2025d). For flow matching and diffusion policy, the problem is more challenging since no tractable log-likelihood is available. (Xu et al., 2024; Luo et al., 2025a;b) explored this problem on small-model backbones, while later works extend it to π-series VLAs (Chen et al., 2026; Zhang et al., 2026a) by approximating action log-likelihood. Besides, DSRL (Wagenmaker et al., 2025) freezes the VLA policy and use RL to optimize the latent noise space, serving as an alternative to online RL methods. In our paper, we demonstrate existing works cannot be applied straightforwardly to multi-agent settings, and provide the first RL pipeline for multi-agent VLA.

Embodied Multi-Agent Collaboration. Embodied multi-agent collaboration has advanced rapidly with existing works broadly falling into two categories. The first focuses on high-level collaborative planning, where VLMs decompose a shared task into agent-specific subgoals and allocate them into different agents (Mandi et al., 2023; Kannan et al., 2024; Zhang et al., 2024), with further mechanisms such as execution feedback, embodiment awareness, and sequential evaluation to improve robust collaboration (Liu et al., 2025b; Chen et al., 2025a; Zhang et al., 2025c). The second line moves beyond symbolic planning toward action-level multi-robot execution. Benchmarks include RoboTwin (Chen et al., 2025b) for bimanual manipulation together with rule-based low-level execution. RoboFactory (Qin et al., 2025) extends the setting to cooperation with compositional constraints and more than two agents. Another line of work leverages VLMs for high-level task scheduling and coordination among multiple embodied agents (Tan et al., 2025a; Sun et al., 2025), with subsequent works studying sim-to-real transfer (Kang et al., 2026). However, these works rely on rule-based low-level control or simply assign the plan to VLAs, neglecting the fine-grained low-level skills required for multi-robot cooperation. The work most closely related to ours is CHORUS (Doshi et al., 2026), which achieves multi-VLA cooperation via supervised fine-tuning. However, how to further improve the capability of multi-agent VLAs through RL remains largely unexplored.

3 PRELIMINARIES

Problem Formulation. Under the literature of Multi-Agent Reinforcement Learning (MARL) (Rashid et al., 2018; Yu et al., 2022), we formulate our problem as a Markov game (Littman, 1994), defined by a tuple $\mathcal{G} = \langle \mathcal{N}, \Psi, \mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, \mathcal{P}, R, \gamma \rangle$. Here $\mathcal{N} = \{1, ..., N\}$ is the set of N agents, Ψ is space of language instructions, $\mathcal{S}$ is the state space, $\mathcal{A}^i$ is the individual action space, with $\mathcal{A} = \times_{i \in \mathcal{N}} \mathcal{A}^i$ the joint action space. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the state transition probability. $R : \mathcal{S} \times \mathcal{A} \times \Psi \to \mathbb{R}$ is the shared reward function, $\gamma \in [0, 1)$ is the discount factor.

At time $t = 0$, all agents receive a shared instruction $\psi \in \Psi$. At each timestep t, each agent observes $s_t \in \mathcal{S}$ and takes an action chunk $a_t^i \sim \pi^i(\cdot|s_t, \psi)$ drown from its policy, forming a joint action chunk $\mathbf{a}_t$. The joint policy is $\pi = \prod_{i=1}^N \pi^i$. After that, the environment proceeds to next state following the transition probability $\mathcal{P}(s_{t+1}|s_t, \mathbf{a}_t)$, and yields reward $r_t = R(s_t, \mathbf{a}_t, \psi)$ for each agent. For generality, we avoid task-specific rewards and instead adopt a general reward scheme, including a penalty of -100 at the end of an episode if the task fails, and a step-wise penalty of -1 at each timestep to encourage faster task completion. Let ρ_0 be the initial state distribution, the joint objective is $J(\pi) = \mathbb{E}_{s_0 \sim \rho_0}[\mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_t | s_0, \psi]]$. The corresponding V and Q function is $V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_t | s_0 = s, \psi]$ and $Q_\pi(s, \mathbf{a}) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_t | s_0 = s, \mathbf{a}_0 = \mathbf{a}, \psi]$, respectively.

Handling Partial Observability. In practice, each VLA may have local observation as input, rather than global state. The setting is called Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek & Amato, 2016). However, the process does not satisfy the Markov property when each agent only assess to partial observation, and is proven to be NEXP-complete (Bernstein et al., 2002). It is therefore standard in MARL to make theoretical analysis under full observability (Kuba et al., 2022; Zhong et al., 2024; Zhang et al., 2025c). In practice, we feed camera

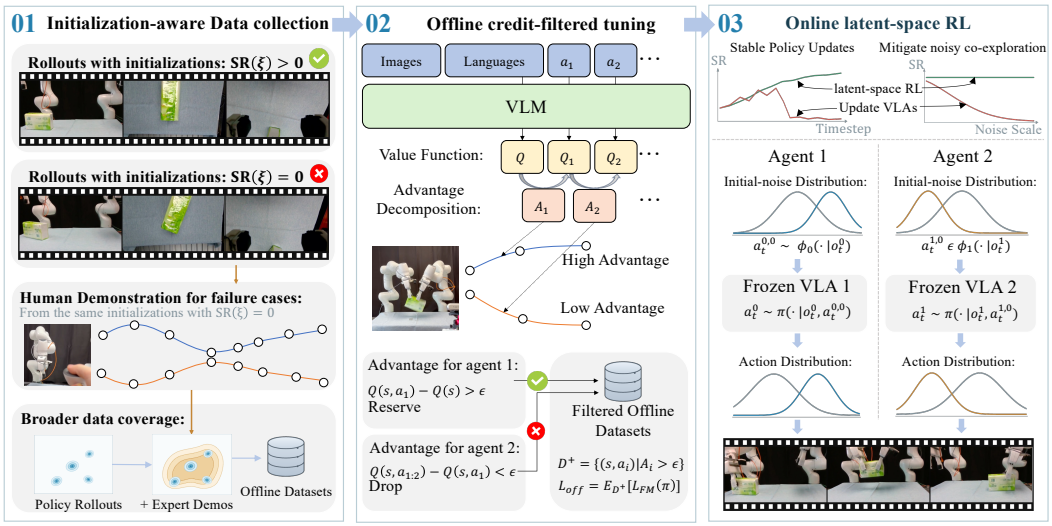


Figure 1: The three-stage recipe for reinforcement fine-tuning multi-agent VLAs. We first perform *Initialization-Aware Data Collection* to broaden coverage of environment conditions, followed by *Offline Credit-Filtered Tuning* to decompose agent-wise advantages and reinforce high-contribution behaviors. Finally, we extend DSRL to the multi-agent setting and perform *Online Latent-Space Reinforcement Learning* to enable further improvement through trial and error.

RGB observations to the VLA backbone directly, following common MARL implementations that use partial observations as policy input (Rashid et al., 2018; Yu et al., 2022).

Heterogeneous Agent Reinforcement Learning. In multi-agent reinforcement learning, a fundamental challenge is to evaluate the contribution of each agent to the joint return, known as *credit assignment* (Yang et al., 2025). Heterogeneous agent reinforcement learning (Kuba et al., 2022; Zhong et al., 2024) solves this by estimating the agent-wise advantage according to the *multi-agent advantage decomposition* lemma. Agents are then updated sequentially with non-negative advantage, yielding a *monotonic improvement guarantee* on the joint return at every on-policy update.

Formally, let $i_{1:m}$ denote an ordered subset $\{i_1, \ldots, i_m\} \in \mathcal{N}$, and let $-i_{1:m}$ be its complement. The multi-agent Q function is $Q_\pi^{i_{1:m}}\left(s, a^{i_{1:m}}\right) \triangleq \mathbb{E}_{\pi^{-i_{1:m}}}\left[Q_\pi\left(s, a^{i_{1:m}}, a^{-i_{1:m}}\right)\right]$. For disjoint sets $j_{1:k}$ and $i_{1:m}$, the multi-agent advantage function is $A_\pi^{i_{1:m}}\left(s, a^{j_{1:k}}, a^{i_{1:m}}\right) \triangleq Q_\pi^{j_{1:k}, i_{1:m}}\left(s, a^{j_{1:k}}, a^{i_{1:m}}\right) - Q_\pi^{j_{1:k}}\left(s, a^{j_{1:k}}\right)$. Then, we have the following multi-agent advantage decomposition lemma:

Lemma 3.1 (Multi-Agent Advantage Decomposition (Kuba et al., 2022)) *In any cooperative Markov games, given a joint policy π, for any state s and any agent subset $i_{1:m}$, the equation holds:*

$$A_\pi^{i_{1:m}}\left(s, a^{i_{1:m}}\right) = \sum_{j=1}^{m} A_\pi^{i_j}\left(s, a^{i_{1:j-1}}, a^{i_j}\right). \tag{1}$$

4 Method

In this section, we introduce a three-stage recipe for multi-agent reinforced fine tuning, including data collection for broad distribution coverage, offline fine-tuning to reinforce desired behaviors and online fine-tuning that tunes latent noise space in flow matching to improve from trials and errors. The overall process is illustrated in Fig. 1. The pseudo code is available in Appendix A.

4.1 Initialization-Aware Data Collection

A general-purpose VLA must succeed across diverse environment configurations, including object poses, robot poses, and task layouts. However, recent studies have shown that existing VLAs are brittle to variations in these initial conditions (Wang et al., 2025). Moreover, in challenging multi-agent settings, VLAs can achieve a zero success rate under certain initializations. Every policy rollout then receives the same negative reward, rendering reinforcement learning infeasible. We attribute such

failures to the lack of the primitive skills for these configurations, and propose an initialization-aware data collection scheme that calls human demonstrations only under such conditions.

Formally, let $\xi \in \Xi$ denote an environment initialization. For each ξ, we execute the pretrained policy for K episodes and estimate its empirical success rate $\mathrm{SR}(\xi)$. When $\mathrm{SR}(\xi) > 0$, reinforcement learning is feasible, and we retain these K rollouts containing both successful and failed trajectories for subsequent reinforced fine-tuning. In contrast, $\mathrm{SR}(\xi) = 0$ indicates that the pretrained policy lacks the primitive behavior required for this initialization, and that sparse-reward exploration alone is unlikely to bootstrap effective learning within a reasonable interaction budget. For these initializations only, we calls for human experts for successful demonstrations, giving a minimal set of successful behaviors that expands the support of the policy and enables reinforcement learning to operate in regions that would otherwise yield no positive signal.

Despite being a simple solution, our initialization-aware data collection gathers data with broad coverage of environment conditions, making subsequent reinforcement learning possible. It also strikes a balance between fully autonomous exploration and exhaustive human demonstration collection, invoking human effort only when necessary.

4.2 Offline Credit-Filtered Tuning

Given the mixed offline dataset produced by initialization-aware data collection, we apply offline RL to reinforce desired behaviors. A key obstacle in the multi-VLA setting is *credit assignment* (Rashid et al., 2018). Since task success is a joint outcome of all agents, an individual agent's contribution to a rollout is hard to assess. We therefore propose offline credit-filtered tuning, which quantifies each agent's contribution by its individual advantage and updates the policy only on transitions with positive advantage, thereby reinforcing beneficial behaviors while filtering out the rest.

Formally, given an offline dataset $\mathcal{D}_{\text{offline}}$, we estimate the agent-wise advantage as follows. Let $i_{1:N} = i_1, i_2, ...i_N$ an arbitrary fixed decision order of N agents, with $a^i_{1:m} = (a^{i_1}, ..., a^{i_m})$ for the first m actions, with $a^i_0 = \varnothing$. Given the discounted return $G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ of a trajectory in the dataset, we fit an agent-combined Q function by sequentially regressing onto the same return:

$$\min_Q \mathbb{E}_{\tau \in \mathcal{D}_{\text{offline}}} \sum_{m=0}^{N} \left[H(G_t, Q_\pi^{i_{1:m}}(s_t, \mathbf{a}_t^{i_{1:m}})) \right], \tag{2}$$

where the two boundary cases are the V function, $Q_\pi^{i_{1:0}}(s_t) = V_\pi(s_t)$, and the joint value function $Q_\pi^{i_{1:N}}(s_t, \mathbf{a}_t^{i_{1:N}}) = Q_\pi(s_t, \mathbf{a}_t)$. Every intermediate $Q_\pi^{i_{1:m}}$ corresponds to the value after revealing the action taken by first m agents, before the rest $N - m$ agents have acted.

The critic is implemented as a transformer that accepts a variable-length action input, so all $N + 1$ terms are learned by a shared network in one pass. Following the distributional value function design of $\pi_{0.6}$ (Intelligence, 2025b), which yields better empirical results, each Q function outputs a 201-bin distribution over returns, and H is the cross entropy between empirical return G_t and the predicted distribution. Given the learned agent-combined Q function, the advantage of each individual agent is obtained as a telescoping difference between consecutive orders:

$$A_\pi^{i_m}(s_t, \mathbf{a}_t^{i_{1:m-1}}, a_t^{i_m}) = Q_\pi^{i_{1:m}}\left(s_t, \mathbf{a}_t^{i_{1:m}}\right) - Q_\pi^{i_{1:m-1}}\left(s_t, \mathbf{a}_t^{i_{1:m-1}}\right), \quad m = 1, 2, ...N. \tag{3}$$

By the multi-agent advantage decomposition lemma (Kuba et al., 2022), the sum of individual advantage corresponds exactly to the joint advantage $A_\pi^{i_{1:N}}\left(s, a^{i_{1:m}}\right) = \sum_{j=1}^{N} A_\pi^{i_j}\left(s, a^{i_{1:j-1}}, a^{i_j}\right)$.

The estimated agent-wise advantage gives us a per-agent notion of desirable behavior. An action is worth reinforcing when it improves upon what that agent would have done on average under the behavioral policy π, *i.e.*, when $A_\pi^{i_m} > 0$. This results in an agent-wise advantage filtered dataset:

$$\mathcal{D}^+ = \left\{ \left(s_t, a_t^{i_m}\right) \in \mathcal{D}_{\text{offline}} \middle| \; A_\pi^{i_m}(s_t, \mathbf{a}_t^{i_{1:m-1}}, a_t^{i_m}) > 0 \right\}, \quad m = 1, 2, ...N. \tag{4}$$

Afterwards, for each individuals, we perform offline fine-tuning on $\mathcal{D}^+$ using the standard conditional flow-matching objective in π_0 (Black et al., 2026): $\mathcal{L}_{\text{offline}} = \mathbb{E}_{\mathcal{D}^+}\left[\mathcal{L}_{\text{FM}}(\pi)\right]$.

Proposition 4.1 (Monotonic Policy Improvement) *Let* $\pi^+ \in \arg\min_\pi \mathbb{E}_{\mathcal{D}^+}[\mathcal{L}_{\text{FM}}(\pi)]$. *Assume (1) the critic gives accurate advantages* $A_\pi^{i_m}$ *on the states visited by* π^+*, and* π^+ *operates in the*

support of $\mathcal{D}^+$, (2) an action kept by the filter has non-negative expected advantage when the preceding agents $i_{1:m-1}$ act according to π^+, then we have $J(\pi^+) - J(\pi) \geq 0$.

Proof sketch. By performance difference lemma (Kakade & Langford, 2002) and multi-agent advantage decomposition lemma, the improvement over expected return is defined as the expectation over agent-wise advantage. Since actions keep a non-negative advantage with preceding agents follows π^+, the expected return cannot get worse. The proof details are in Appendix B.

4.3 Online Latent-Space Reinforcement Learning

Offline RL can exceed demonstration quality by stitching suboptimal trajectories, but it cannot interact with the environment and thus cannot discover behaviors beyond the data support. As such, we further tune multi-agent VLAs via online RL. Extending online RL to flow-matching VLAs is nontrivial, since flow matching yields no tractable action likelihood and the log-probability must instead be approximated. Consequently, existing works mainly operate at small policy scales (Zhang et al., 2026a; Xu et al., 2024; Luo et al., 2025a), with π_{RL} (Chen et al., 2026) the only VLA-scale research with publicly available implementation (Yu et al., 2025).

However, while π_{RL} works well for simple multi-agent tasks, the success do not extend to harder settings. We hypothesize the reason to noisy co-exploration and unstable policy updates. First, the exploratory noises in online RL fine-tuning consists of unstructured Gaussian noise injected to each agent independently. Such uncoordinated noise disrupts the cooperative behavior of the offline-tuned policies and can decrease the success rate below its pre-RL level. Second, the episode reward is sparse and distributed across both time horizon and denoising steps, yielding high-variance gradient updates. While this variance is tolerable in the single-agent setting, it is exacerbated in the multi-agent setting by non-stationarity. From agent i's perspective, the policy changes of other agents are absorbed in environment dynamics, violating the stationarity assumption required by RL.

Guided by this diagnosis, we turn to online RL in the latent noise space of the flow-matching policy, and adopt DSRL (Wagenmaker et al., 2025) as our base algorithm. Formally, consider a policy $\pi(a|s)$, with agent index dropped for simplicity. π-series VLAs employ rectified flow (Liu, 2022), which is a linear ODE from noise $a^0 \sim \mathcal{N}(0, \mathbf{I})$ to the final action output a^1 during inference, with a linear interpolation $a_t^\tau = \tau a_t^1 + (1-\tau)a_t^0$, $\tau \in [0, 1]$. Instead of tuning the rectified flow, DSRL tunes the latent noise a^0. In place of sampling from Gaussian distribution $\mathcal{N}(0, \mathbf{I})$, they learn the latent noise from a policy $\phi(a^0|s)$ using online RL. This paradigm avoids tuning the weights of VLA and provides a lightweight solution since $\phi(a^0|s)$ can be a small network instead of large VLAs.

DSRL serves as a clean remedy against the problem of noisy co-exploration and unstable policy updates in multi-VLA. First, since the VLA is fixed, exploration is confined to the latent noise space, producing in-distribution, temporally coherent behaviors. Joint exploration thus preserves basic cooperation from offline fine tuning and results in non-degenerate success rate. Second, tuning in latent space avoids modeling the intractable marginal likelihood of flow matching, and constrains policies within the set of offline-tuned behaviors, lowering policy variance during update and thus bounding the non-stationarity during training.

5 Experiments

Experimental setting. We evaluate our multi-agent VLA on a total of 11 cooperation tasks. To evaluate closely synchronized bimanual actions, we consider four tasks in RoboTwin (Chen et al., 2025b), including *Lift Pot*, *Grab Roller*, *Handover Mic*, and *Put Bottles Dustbin*. To evaluate coordination of three or four agents, we consider four tasks in RoboFactory (Qin et al., 2025), including *Camera Alignment*, *Take Photo*, *Three Robots Stack Cubes*, and *Long Pipeline Delivery*. We train low-level end-to-end VLAs for these tasks. For real-world evaluation, we design three tasks that requires cooperation of two Franka robots, including *Handover*, *Lift*, and *Handover & Drop*.

Baselines. Due to the scarcity of researches in RL methods for multi-agent VLAs, we use CHORUS (Doshi et al., 2026) as a baseline for supervised fine-tuning, and implement our data collection, offline RL and online RL on its checkpoint. For online stage, we additionally compare our choice of DSRL with Flow-SDE and Flow-Noise (Chen et al., 2026), with results reported in ablations. Our backbones include π_0 (Black et al., 2026) and $\pi_{0.5}$ (Intelligence, 2025a).

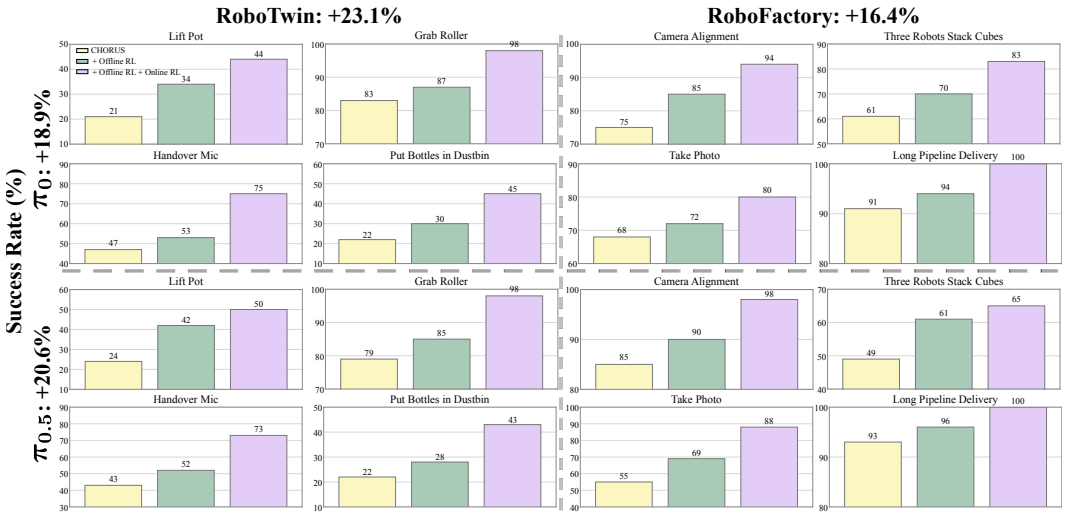


Figure 2: Success rate (%, ↑) under 8 different simulation tasks from RoboTwin and RoboFactory with backbone models π_0 and $\pi_{0.5}$.

Implementation Details. We adopt a decentralized execution setting similar to CHORUS (Doshi et al., 2026). Each VLA receives its local observations including wrist-camera RGB images and proprioceptive states. All agents additionally share a third-person RGB observation and the language instruction. For each simulation task, we collect 50 expert demonstrations for supervised fine-tuning and 30 demonstrations for real-world tasks. During the initialization-aware data collection stage, we evaluate 96 environment initialization configurations for each simulation task and 50 configurations for each real-world task, with $K = 5$ the number of rollouts for each configuration in simulation and $K = 1$ for real world. We evaluate each task using 100 rollouts in simulation and 50 rollouts in real-world. See additional implementation details in Appendix C.

5.1 Simulation Results

We evaluate our multi-agent VLA on 8 simulation tasks in two environments. We include our initialization-aware data collection automatically and report our offline and online RL stages as + *Offline RL* and + *Online RL*, respectively. As shown in Fig. 2, considering learning stages, our method improves an average of $+8.1\%$ after offline RL and $+19.7\%$ after online RL. As for backbones, our method improves an average of $+18.9\%$ on π_0 and $+20.6\%$ on $\pi_{0.5}$. As for environments, our method improves an average of $+23.1\%$ on bimanual RoboTwin tasks and $+16.4\%$ on RoboFactory with 3-4 agents. Together, the results show the effectiveness of our multi-agent VLA on both online and offline RL stages, different backbones, and varying agent numbers.

5.2 Ablations

In this section, we provide extensive ablation on each state of our Multi-Agent VLA with visualization of the effectiveness of each parts. We select $\pi_{0.5}$ as backbone and consider four harder tasks that can not reliably solved by current methods to provide insights on potential on further gains.

Ablation on initialization-aware data collection. We show the effectiveness of our initialization via offline RL on alternative data without human demonstrations (+*offline RL w/o init*). As shown in Table 1, adding human demonstrations provides better coverage, resulting in +5.7% success rate on average. Additionally, we show initialization benefits further RL. In Fig. 3, we report the success rate on 25 randomly sampled initializations. Our data collection results in broader success rate on different initializations via offline RL, and further enables broader success coverage in subsequent

Table 1: Ablation study on the initialization-aware data collection. We report success rate (%, ↑).

Task	CHORUS	+ Offline RL w/o IA	+Offline RL +Online RL w/o IA	+Offline RL	+Offline RL +Online RL
Lift Pot	24	30	46	42	**50**
Handover Mic	43	46	63	52	**73**
Three Robots Stack Cubes	49	57	58	61	**65**
Take Photo	55	68	79	69	**88**
Average	42.8	50.3	61.5	56.0	**69.0**

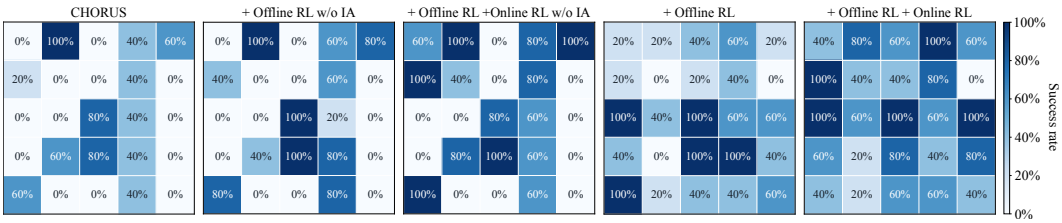
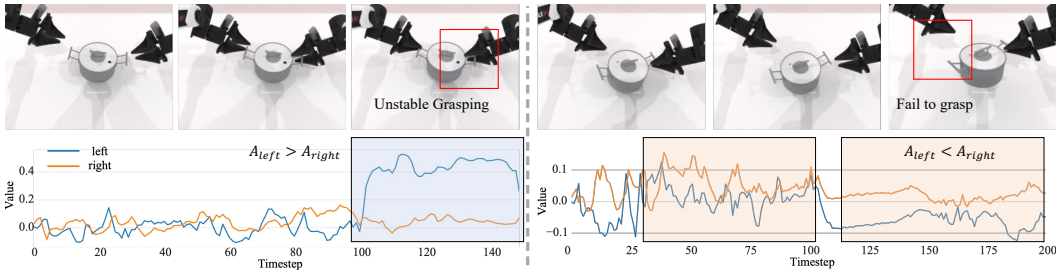


Figure 3: Visualization of success rates under different environment initializations. *Initialization-Aware Data Collection* identifies and compensates for challenging configurations in which agents struggle to acquire successful behaviors through autonomous exploration.



(a) The unstable grasp behavior of the right arm.

(b) The left arm fails to grasp the handle.

Figure 4: Visualization of agent-wise credit assignments and the behaviors of them.

online RL. The result states the importance of data coverage in multi-agent VLAs, and also points out the current limitation of VLAs in generalizing against various initializations and conditions.

Ablation on offline credit-filtered tuning. As shown in Fig. 2, offline RL provides an average +8.1% success rate. We also visualize the agent-wise credit assignment in Fig. 4. In case (a), the advantage of right arm is lower than the left arm, corresponding to the unstable grasp behavior of the right arm. In case (b), the advantage of left arm is lower than the right arm, and in the video the left arm fails to grasp the handle, leading to task failure, showing the capability to attribute failure to each agents.

Ablation on online latent-space reinforcement learning. As shown in Fig. 2, online RL is generally effective for multi-agent VLAs, improving +11.6% on the offline tuned checkpoint. We further compare DSRL with Flow-Noise and Flow-SDE (Chen et al., 2026) as alternative online RL methods. As shown in Table. 2, DSRL achieves the highest success rate, outperforming Flow-Noise by +7.5% and Flow-SDE by 8.9% on average. To understand why Flow-based RL offers limited performance, we offer evidences to support our claim that RL methods for single-agent VLA cannot be transferred directly to multi-agent VLAs due to noisy co-exploration and unstable policy updates (Section 4.3). To illustrate the effect of noisy co-exploration, we increase the exploratory noise of flow-based methods. As shown in Fig. 5, increasing the noise scale in Flow-SDE and Flow-Noise consistently reduces the success rate as policy explores. In contrast, DSRL operates in latent noise space initialized from $\mathcal{N}(0, I)$, which is identical to the original VLA setting and do not lose success rates. To illustrate the effect of unstable gradient updates, we illustrate learning dynamics of DSRL and flow-based methods in Fig. 6. We find flow-based noise can occasionally crash during training, while our DSRL improves performance in a stable way. Additionally, these experiments also calls for a multi-agent version of flow-matching algorithm for VLAs that handle the problem of noisy exploration and unstable gradients.

Table 2: Comparison of different online RL methods. We report success rate (%, ↑).

Task	Offline RL	+Flow-SDE	+Flow-Noise	+DSRL
Lift Pot	42	43	43	**50**
Handover Mic	52	56	62	**73**
Three Robots Stack Cubes	61	62	63	**65**
Take Photo	69	82	78	**88**
Average	56.0	60.1	61.5	**69.0**

5.3 Real-world Results

Finally, we evaluate our Multi-Agent VLA in real-world settings. We consider $\pi_{0.5}$ backbone with three manipulation tasks that requires coordinating two Franka robots. In *Handover*, one robot hands an object over to the other. In *Lift*, two robots jointly grasp the handles of a basket and lift it. In *Handover & Drop*, one robot picks up a piece of trash and hands it to the other robot, which

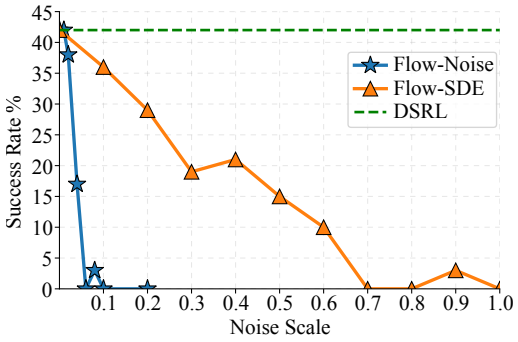


Figure 5: The success rate under different noise scales of method Flow-Noise and Flow-SDE. Flow-based RL shows noisy co-exploration.

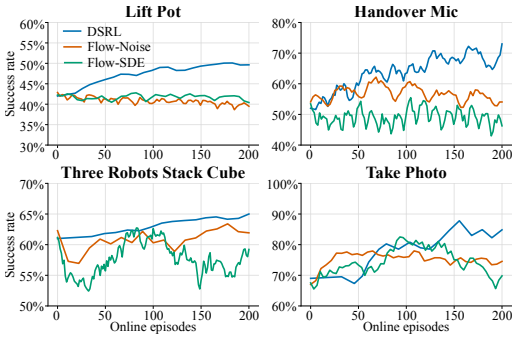
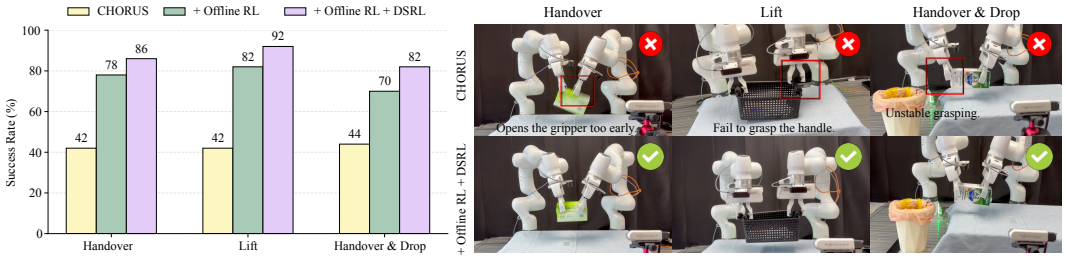


Figure 6: Learning dynamics of online RLs. Flow-based RL shows unstable policy update.



(a) Success Rate under 3 different tasks

(b) Overview of different collaborative tasks

Figure 7: The success rate (%, ↑) and the overview under 3 different collaborative tasks on 2 Franka robots. Our training recipe effectively enhances the collaborative performance of multi-agent VLAs.

subsequently places it into a dustbin. We use the same three-stages pipeline in simulation, but change the online RL stage to DSRL with 50 real-world rollouts as a proof-of-concept result.

The success rates can be seen in Fig. 7(a). Our multi-agent VLA stays consistently effective in real world, resulting in +34.0% success rate by offline RL and +10.0% by DSRL, with online RL improvements achieved by 50 real world rollouts only. We further provide a case study of our method and CHORUS in Fig. 7(b). The failure cases of CHORUS shows supervised fine tuning paradigm alone is insufficient to learn fine-grained VLA coordination. For *Handover*, the left robot opens its gripper before the right robot securely grasps the object, causing it to fall onto the table. For *Lift*, the right robot fails to securely grasp the basket handle before lifting. For *Handover & Drop*, the receiving robot fails to firmly grasp the trash handed over by the other robot, resulting in an unsuccessful handover. In contrast, our multi-agent VLA is free of these problems, exceeding the success rate of behavior cloning pipeline by large margins via reinforcing desired behaviors.

6 Conclusion

In this work, we propose a three-stage reinforced fine-tuning pipeline for cooperative multi-agent vision-language-action models. First, initialization-aware data provide better distribution coverage by sweeping over initial configurations and invokes human demonstrations with zero success rates. Second, offline credit-filtered tuning rewards good desired individual behavior by VLA-level credit assignment, which assigns credit to individual agents and fine-tunes on per-agent trajectories with positive advantage rather than on entire joint rollouts. Third, online latent-space fine tuning freeze the VLA and perform RL in its latent noise space to reduce the challenge of noisy co-exploration and unstable gradient updates in online methods for VLAs. Experiments with π_0 and $\pi_{0.5}$ across 11 simulated and real-world tasks demonstrate consistent improvements, achieving gains of 23.1%, 16.4%, and 44% on RoboTwin, RoboFactory, and real-world tasks, respectively.

AI USE STATEMENT

LLMs were employed for text polishing. The authors have thoroughly reviewed and validated all content presented in the paper.

ETHICS STATEMENT

Our work studies cooperation among multiple VLAs. The focus of this paper is to strengthen the ability of robot foundation models to coordinate with one another, thereby broadening their applicability to tasks such as collaborative assembly and multi-robot household assistance. To the best of our knowledge, our work raises no immediate ethical concerns. However, same as any works in robotics, deployment in real-world settings should be accompanied with appropriate safety safeguards and human oversight.

REPRODUCIBILITY STATEMENT

We provide implementation details in Appendix. C. Code is available at `https://anonymous.4open.science/r/mavla_rft-2BC0/`.

REFERENCES

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking, 2025. URL https://arxiv.org/abs/2512.05964.

Kevin Black, Noah Brown, Danny Driess, et al. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023a. URL https://arxiv.org/abs/2307.15818.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, et al. Rt-1: Robotics transformer for real-world control at scale, 2023b. URL https://arxiv.org/abs/2212.06817.

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. URL https://arxiv.org/abs/2506.21539.

Junting Chen, Checheng Yu, Xunzhe Zhou, Tianqi Xu, Yao Mu, Mengkang Hu, Wenqi Shao, Yikai Wang, Guohao Li, and Lin Shao. Emos: Embodiment-aware heterogeneous multi-robot operating system with llm agents, 2025a. URL https://arxiv.org/abs/2410.22662.

Kang Chen, Zhihao Liu, Tonghe Zhang, et al. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2026. URL https://arxiv.org/abs/2510.25889.

Tianxing Chen, Zanxin Chen, Baijun Chen, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025b. URL https://arxiv.org/abs/2506.18088.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy, 2025c. URL https://arxiv.org/abs/2502.05450.

Zengjue Chen, Runliang Niu, He Kong, Qi Wang, Qianli Xing, and Zipei Fan. Tgrpo :fine-tuning vision-language-action model via trajectory-wise group relative policy optimization, 2025d. URL https://arxiv.org/abs/2506.08440.

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, 2024. URL https://arxiv.org/abs/2303.04137.

Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models, 2025. URL https://arxiv.org/abs/2310.08864.

Ria Doshi, Tian Gao, Annie Chen, Chelsea Finn, and Jeannette Bohg. Chorus: Decentralized multi-embodiment collaboration with one vla policy, 2026. URL https://arxiv.org/abs/2606.12352.

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, 2025. URL https://arxiv.org/abs/2501.16664.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, and Yuntao Chen. Dita: Scaling diffusion transformer for generalist vision-language-action policy, 2025. URL https://arxiv.org/abs/2503.19757.

Physical Intelligence. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025a. URL https://arxiv.org/abs/2504.16054.

Physical Intelligence. $\pi^*_{0.6}$: a vla that learns from experience, 2025b. URL https://arxiv.org/abs/2511.14759.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pp. 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

Li Kang, Yutao Fan, Rui Li, Heng Zhou, et al. Coenv: Driving embodied multi-agent collaboration via compositional environment, 2026. URL https://arxiv.org/abs/2604.05484.

Shyam Sundar Kannan, Vishnunandan L. N. Venkatesh, and Byung-Cheol Min. Smart-llm: Smart multi-agent robot task planning using large language models, 2024. URL https://arxiv.org/abs/2309.10062.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, et al. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=EcGGFkNTxdJ.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, et al. Simplevla-rl: Scaling vla training via reinforcement learning, 2025. URL https://arxiv.org/abs/2509.09674.

Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning, 2026. URL https://arxiv.org/abs/2505.11917.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, pp. 157–163, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1558603352.

Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, Chengkai Hou, Mengdi Zhao, KC alex Zhou, Pheng-Ann Heng, and Shanghang Zhang. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model, 2025a. URL https://arxiv.org/abs/2503.10631.

Kehui Liu, Zixin Tang, Dong Wang, Zhigang Wang, Xuelong Li, and Bin Zhao. Coherent: Collaboration of heterogeneous multi-robot system with large language models, 2025b. URL https://arxiv.org/abs/2409.15146.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *ArXiv*, abs/2209.14577, 2022. URL https://api.semanticscholar.org/CorpusID:252596014.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation, 2025c. URL https://arxiv.org/abs/2410.07864.

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. URL https://arxiv.org/abs/2505.18719.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning, 2025a. URL https://arxiv.org/abs/2401.16013.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning, 2025b. URL https://arxiv.org/abs/2410.21845.

Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models, 2023. URL https://arxiv.org/abs/2307.04738.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance, 2025. URL https://arxiv.org/abs/2410.13816.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

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. URL https://arxiv.org/abs/2501.09747.

Yiran Qin, Li Kang, Xiufeng Song, Zhenfei Yin, Xiaohong Liu, Xihui Liu, Ruimao Zhang, and Lei Bai. Robofactory: Exploring embodied agent collaboration with compositional constraints, 2025. URL https://arxiv.org/abs/2503.16408.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Junyang Shu, Zhiwei Lin, and Yongtao Wang. Rftf: Reinforcement fine-tuning for embodied agents with temporal feedback, 2025. URL https://arxiv.org/abs/2505.19767.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

Nan Sun, Chengming Shi, and Dongyuwen. [AML]interactgen: Enhancing human-involved embodied task reasoning through LLM-based multi-agent collaboration, 2025. URL https://openreview.net/forum?id=WDdiCIkpxC.

Huajie Tan, Xiaoshuai Hao, Cheng Chi, Minglan Lin, Yaoxu Lyu, Mingyu Cao, Dong Liang, Zhuo Chen, Mengsi Lyu, Cheng Peng, Chenrui He, Yulong Ao, Yonghua Lin, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Roboos: A hierarchical embodied framework for cross-embodiment and multi-agent collaboration, 2025a. URL https://arxiv.org/abs/2505.03673.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models, 2025b. URL https://arxiv.org/abs/2505.17016.

Octo Model Team. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. Vlatest: Testing and evaluating vision-language-action models for robotic manipulation. *Proceedings of the ACM on Software Engineering*, 2(FSE):1615–1638, 2025. ISSN 2994-970X. doi: 10.1145/3729343. URL http://dx.doi.org/10.1145/3729343.

Junjie Wen, Minjie Zhu, Yichen Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Chengmeng Li, Xiaoyu Liu, Yaxin Peng, Chaomin Shen, and Feifei Feng. Diffusion-vla: Generalizable and interpretable robot foundation model via self-generated reasoning, 2025a. URL https://arxiv.org/abs/2412.03293.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control, 2025b. URL https://arxiv.org/abs/2502.05855.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. Rldg: Robotic generalist policy distillation via reinforcement learning, 2024. URL https://arxiv.org/abs/2412.09858.

Yaodong Yang, Chengdong Ma, Zihan Ding, Stephen McAleer, Chi Jin, Jun Wang, and Tuomas Sandholm. Game-theoretic multiagent reinforcement learning, 2025. URL https://arxiv.org/abs/2011.00583.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation, 2025. URL https://arxiv.org/abs/2509.15965.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models, 2024. URL https://arxiv.org/abs/2307.02485.

Hongyin Zhang, Zifeng Zhuang, Han Zhao, Pengxiang Ding, Hongchao Lu, and Donglin Wang. Reinbot: Amplifying robot visual-language manipulation with reinforcement learning, 2025a. URL https://arxiv.org/abs/2505.07395.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning, 2026a. URL https://arxiv.org/abs/2505.22094.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. Dreamvla: A vision-language-action model dreamed with comprehensive world knowledge, 2025b. URL https://arxiv.org/abs/2507.04447.

Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. Towards efficient llm grounding for embodied multi-agent collaboration, 2025c. URL https://arxiv.org/abs/2405.14314.

Zongzheng Zhang, Jingrui Pang, Zhuo Yang, Kun Li, et al. Dexora: Open-source vla for high-dof bimanual dexterity, 2026b. URL https://arxiv.org/abs/2605.18722.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024.

APPENDIX FOR "COOPERATIVE MULTI-AGENT VISION-LANGUAGE-ACTION MODELS VIA REINFORCED FINE TUNING"

A PSEUDO CODE FOR OUR POST-TRAINING PIPELINE.

Algorithm 1 summarizes our three-stage reinforced fine-tuning pipeline, consisting of initialization-aware data collection, offline credit-filtered tuning, and online latent-space reinforcement learning.

B PROOF OF PROPOSITION 4.1

We provide a theoretical justification for the monotonic improvement of our offline credit-filtered tuning. Let $\pi(\mathbf{a} \mid s) = \prod_{m=1}^{N} \pi^{i_m}(a^{i_m} \mid s)$ denote the behavioral policy that generates $\mathcal{D}_{\text{offline}}$, and let π^+ denote the policy obtained by fine-tuning on the positive-advantage dataset $\mathcal{D}^+$.

Our analysis relies on the following assumptions: (1) the critic gives accurate advantages $A_\pi^{i_m}$ on the states visited by π^+, and π^+ operates in the support of $\mathcal{D}^+$; (2) an action kept by the filter has non-negative expected advantage when the preceding agents $i_{1:m-1}$ act according to π^+.

Proposition B.1 (Monotonic Improvement of Credit-Filtered Tuning) *Under the assumptions above, the policy π^+ obtained by credit-filtered tuning satisfies*

$$J(\pi^+) - J(\pi) \geq 0. \tag{5}$$

By the performance-difference lemma,

$$J(\pi^+) - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi^+}, \mathbf{a} \sim \pi^+} \left[A_\pi(s, \mathbf{a}) \right]. \tag{6}$$

Using the multi-agent advantage decomposition lemma (Kuba et al., 2022), the joint advantage can be decomposed as

$$A_\pi(s, \mathbf{a}) = \sum_{m=1}^{N} A_\pi^{i_m} \left(s, \mathbf{a}^{i_{1:m-1}}, a^{i_m} \right). \tag{7}$$

Therefore,

$$J(\pi^+) - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi^+}} \sum_{m=1}^{N} \mathbb{E}_{\mathbf{a} \sim \pi^+} \left[A_\pi^{i_m}(s, \mathbf{a}^{i_{1:m-1}}, a^{i_m}) \right]. \tag{8}$$

By construction, credit-filtered tuning retains only actions with positive agent-wise advantage. Under the exact projection and the assumption that an action kept by the filter has non-negative expected advantage with $\mathbf{a}^{i_{1:m-1}} \sim \pi^{+, i_{1:m-1}}$, π^{+, i_m} is therefore supported only on actions satisfying

$$\mathbb{E}_{\mathbf{a} \sim \pi^+} [A_\pi^{i_m}(s, \mathbf{a}^{i_{1:m-1}}, a^{i_m})] \geq 0 \tag{9}$$

for states covered by $\mathcal{D}^+$. Hence, every agent-wise expectation in the above decomposition is non-negative. Since π^+ operates in the support of $\mathcal{D}^+$, summing over agents and integrating over the state-visitation distribution gives

$$J(\pi^+) - J(\pi) \geq 0, \tag{10}$$

which proves the proposition.

Remark. (1) The coverage requirement in Assumption 1 is consistent with our initialization-aware data collection, which supplements policy rollouts with human demonstrations for zero-success initializations, thereby broadening the coverage of successful behaviors and reducing the need for extrapolation. (2) Assumption 2 is plausible when fine-tuning induces moderate policy changes and retained actions have a positive advantage margin. Since fine-tuning starts from π and reinforces selected behaviors from the collected data, these behaviors may remain beneficial under the updated policies of preceding agents, provided that the resulting distribution shift is not large enough to reverse their expected advantages.

Algorithm 1 Reinforced Fine-Tuning for Multi-Agent VLAs

Require: Pretrained multi-agent VLA policies $\boldsymbol{\pi}$; environment initializations Ξ; number of rollouts K; advantage threshold ϵ latent policies $\boldsymbol{\phi} = \{\phi^i\}_{i=1}^N$

Ensure: Offline-tuned VLA policies $\boldsymbol{\pi}^+$ and latent-space policies $\boldsymbol{\phi} = \boldsymbol{\{\phi^i\}_{i=1}^N}$

$\mathcal{D}_{\text{offline}} \leftarrow \emptyset$

Stage I: Initialization-Aware Data Collection

for each initialization $\xi \in \Xi$ **do**
  Collect K rollouts $\mathcal{T}_\xi = \{\tau_\xi^1, \ldots, \tau_\xi^K\}$ using $\boldsymbol{\pi}$
  Estimate empirical success rate $\mathrm{SR}(\xi)$
  if $\mathrm{SR}(\xi) > 0$ **then**
    $\mathcal{D}_{\text{offline}} \leftarrow \mathcal{D}_{\text{offline}} \cup \mathcal{T}_\xi$
  else
    Collect human demonstrations $\mathcal{D}_{\text{human}}^\xi$ under initialization ξ
    $\mathcal{D}_{\text{offline}} \leftarrow \mathcal{D}_{\text{offline}} \cup \mathcal{D}_{\text{human}}^\xi$
  end if
end for

Stage II: Offline Credit-Filtered Tuning

Compute discounted returns G_t for trajectories in $\mathcal{D}_{\text{offline}}$
Train the agent-combined critic $Q_\pi^{i_{1:m}}$ for $m = 0, \ldots, N$ using Eq. 2
$\mathcal{D}^+ \leftarrow \emptyset$
for each transition $(s_t, \mathbf{a}_t)$ in $\mathcal{D}_{\text{offline}}$ **do**
  for $m = 1, \ldots, N$ **do**
    Compute agent-wise advantage using Eq. 3
    if $A_\pi^{i_m} > \epsilon$ **then**
      $\mathcal{D}^+ \leftarrow \mathcal{D}^+ \cup \{(s_t, a_t^{i_m})\}$
    end if
  end for
end for
Fine-tune $\boldsymbol{\pi}$ on $\mathcal{D}^+$ with the flow-matching objective: $\boldsymbol{\pi}^+ \leftarrow \arg\min_{\boldsymbol{\pi}} \mathbb{E}_{\mathcal{D}^+}[\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\pi})]$

Stage III: Online Latent-Space Reinforcement Learning

Freeze the offline-tuned VLA policies $\boldsymbol{\pi}^+$
while online RL has not converged **do**
  Reset the environment and observe $\{o_0^i\}_{i=1}^N$
  for $t = 0, \ldots, T-1$ **do**
    for each agent $i \in \mathcal{N}$ **do**
      Sample latent noise $a_t^{i,0} \sim \phi^i(\cdot \mid o_t^i)$
      Generate action chunk using the frozen VLA: $a_t^i \sim \pi^{+,i}(\cdot \mid o_t^i, a_t^{i,0})$
    end for
    Execute joint action $\mathbf{a}_t = (a_t^1, \ldots, a_t^N)$ and observe r_t and next observations
    Store $(\mathbf{o}_t, \mathbf{a}_t^0, r_t, \mathbf{o}_{t+1})$ in the online replay buffer
  end for
  Update $\boldsymbol{\phi} = \{\phi^i\}_{i=1}^N$ with DSRL using collected online experience
end while
return $\boldsymbol{\pi}^+, \boldsymbol{\phi}$

C Implementation Details

C.1 Decentralized execution

We adopt a decentralized execution setting similar to CHORUS (Doshi et al., 2026). Each VLA receives its local observations, including wrist-camera RGB images, proprioceptive states, and language instructions, and outputs actions only for the end effector of its corresponding robot. Different from CHORUS, where each agent operates solely on its local observations, we additionally provide all agents with a shared third-person RGB observation and the language instruction, since wrist-camera observations alone provide limited global context for effective multi-agent coordination. As for the language instruction, for RoboTwin (Chen et al., 2025b), we use the shared instruction provided by the RoboTwin itself. For RoboFactory and real-world tasks, we use the instruction as below:

Listing 1: Language provided to the agents in different tasks.

```
RoboFactory Simulation:

Camera Alignment:
Agent0: "Grasp one side of the camera, lift the camera and align the
    camera with the steak.",
Agent1: "Grasp one side of the camera, lift the camera and align the
    camera with the steak.",
Agent2: "Grasp the steak, lift the steak and stay still.".

Three Robots Stack Cubes:
Agent0: "Lift the blue cube, place the blue cube to the target and stay
    still.",
Agent1: "Lift the green cube, stay still and place the green cube on the
    blue one.",
Agent2: "Lift the red cube, stay still and place the red cube on the
    green one.".

Take Photo:
Agent0: "Grasp one camera handle, lift one camera handle, and stay still
    .",
Agent1: "Grasp one camera handle, lift one camera handle, and stay still
    .",
Agent2: "Grasp the steak, lift the steak align the camera, and stay still
    .",
Agent3: "After the steak align the camera, press the camera shutter.".

Long Pipeline Delivery:
Agent0: "Grasp the shoe and place it to the target.",
Agent1: "Grasp the shoe and pass it to the next robotic arm.",
Agent2: "Grasp the shoe and pass it to the next robotic arm.",
Agent3: "Grasp the shoe and pass it to the next robotic arm.".

Real-world tasks (shared instructions):
Handover:
Use the left arm to grasp the green-and-white tissue pack, hand it over
    to the right arm, and place it down on the table.

Lift:
Use both arms to grasp opposite sides of the black plastic mesh storage
    basket and lift it up.

Handover & Drop:
Use the right arm to grasp the empty a2 milk carton, hand it over to the
    left arm, and drop it into the white trash bin on the left.
```

Table 4: Training hyperparameters of DSRL.

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
Batch size	16	Actor lr	1×10^{-4}	Critic lr	3×10^{-4}
Training episodes	200	Temperature lr	3×10^{-4}	UTD ratio	1
Hidden dimensions	$(128, 128, 128)$	Latent dimension	50	Discount factor	0.999
Target update rate	0.005	Number of Q-functions	10	Critic reduction	Mean
CNN features	$(32, 32, 32, 32)$	CNN strides	$(2, 1, 1, 1)$	CNN padding	VALID
Encoder type	Small	Encoder normalization	Group	Bottleneck	True
Spatial softmax	True	Dropout rate	0.0	Target entropy	Auto
Action magnitude	1.0	Number of cameras	1	PopArt	False

C.2 Training details

For each simulation task, we collect 50 expert demonstrations for supervised fine-tuning and 30 demonstrations for real-world tasks. We trained the CHORUS models based on these demonstrations. During the initialization-aware data collection stage, we evaluate 96 environment initialization configurations for each simulation task, with $K = 5$ the number of rollouts for each configuration. While in real-world tasks, we evaluate 50 environment initialization configurations with $K = 1$.

During offline credit-filtered tuning, we use the collected rollouts to train the value function. We adopt $\gamma = 1$ to calculate the discounted retures. We use a sparse reward with a step penalty of -1. At the final timestep of each episode, the agent receives a reward of 0 if the task succeeds and a penalty of -1000 if the task fails. The architecture of value function largely follows that of π_0, where agent actions are encoded in the same manner as the noisy actions in π_0. We additionally introduce a value-expert head that discretizes the Monte Carlo return into 201 bins for value prediction. The value function is trained following the procedure of $\pi^*_{0.6}$ (Intelligence, 2025b). For the agent ordering $i_1, \ldots, i_{\mathcal{N}}$, we use a fixed order throughout training and evaluation. Specifically, we use the order 'left, right' for RoboTwin and real-world tasks, while for RoboFactory, we follow the agent ordering defined in the original dataset (Qin et al., 2025). We then compute the agent-wise advantages and filter out samples whose advantages fall below a threshold ϵ. We choose ϵ such that approximately 5% of the original data is retained for subsequent fine-tuning. We add all of the frames from human demonstrations to $\mathcal{D}^+$. As for the finetuning of the CHORUS model, offline-tuned VLA policies, and the value function, we follow the pipeline similar to π_0 (Black et al., 2026) and $\pi_{0.5}$ (Intelligence, 2025a), with the training settings shown in Tab. 3:

Table 3: Training settings of the VLA policies and value functions.

Setting	Value
Batch size	32
Training steps	30,000
VLM	Full-parameter
Action expert	Full-parameter
Chunk horizon	50 for simulation and 20 for real-world

For online latent-space reinforcement learning, we train for 200 exploration episodes and select the checkpoint with the best performance for evaluation in simulation. As for the real-world tasks, we change the online RL stage of DSRL to offline version with 50 real-world rollouts as proof-of-concept result. The parameters of DSRL are shown in Tab. 4. The parameters of Flow-Noise and Flow-SDE are shown in Tab. 5.

For evaluation, we evaluate each task using 100 rollouts in simulation and 50 rollouts in the real world. In simulation, we use an action chunk size of 50, with execution horizons of 50 and 20 for RoboTwin and RoboFactory, respectively. For real-world deployment, we employ Real-Time Chunking (RTC) (Black et al., 2025) for smooth and responsive VLA control, with an action chunk size of 50, an execution horizon of 8, and an overlap of 12 actions between consecutive action chunks.

Table 5: Training hyperparameters of Flow-SDE and Flow-Noise.

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
Training episodes	200	Advantage type	embodied_grpo	Train expert only	True
Top-k	50	Loss type	embodied_grpo	Reward type	chunk_level
Parallel Num	8	PPO clip high	0.2	Separate models	True
Discount factor	0.99	PPO clip low	0.2	Action chunks	50
Chunk steps	16	PPO clip c	3.0	Flow steps	10
Eval chunk steps	16	Actor lr	5×10^{-6}	Noise level	0.1 (Flow-SDE) 0.01 (Flow-Noise)
Normalize advantages	True	Micro-batch size	16	Joint log-prob	True
KL coefficient	0.0	Global batch size	128	Train temperature	1.0
Entropy bonus	0	Max grad norm	2.0	Eval temperature	0.6