SEPARATE ADAPTATION, UNIDIRECTIONAL COORDINATION: LEARNING BIMANUAL MANIPULATION WITH A SHARED VLA BACKBONE

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning bimanual manipulation from limited demonstrations requires both reusable single-arm skills and effective cross-arm coordination. Monolithic policies learn these jointly from costly bimanual data, while compositional methods reuse single-arm vision–language–action (VLA) models but leave the choice and use of exchanged information less explicitly modeled. We propose *Separate Adaptation, Unidirectional Coordination* (SAUC), which separates arm-specific adaptation from cross-arm interaction. Two independent low-rank adapters specialize a shared frozen single-arm VLA backbone. A lightweight State–Intent coordinator then uses the Follower's context to select and modulate the Leader's contextual state and action-readout intent before updating the Follower's intent representation. On six RoboTwin 2.0 tasks, SAUC trained with 50 clean demonstrations per task achieves absolute gains of 5.00% and 5.33% in mean success over the strongest evaluated monolithic pretrained policy under Easy and Hard, respectively. Compared with TwinVLA, it achieves absolute gains of 13.33% and 14.00% while using about $1/6$ as many trainable parameters. Ablations support separate adaptation and explicit coordination, and real-world experiments show higher success than TwinVLA on all three evaluated tasks. We find empirically that a shared single-arm action prior improves bimanual policy performance through tailored arm-specific adaptation and selective information exchange.

1 INTRODUCTION

Bimanual manipulation is a fundamental capability for robots performing parallel or coordinated two-arm tasks in complex environments (Krebs & Asfour, 2022). Monolithic policies map language and observations to joint bimanual actions (Black et al., 2025; Liu et al., 2025), implicitly learning the two arms' joint action structure from bimanual demonstrations. Their performance therefore depends strongly on data scale, yet high-quality bimanual demonstrations are costlier to collect and harder to scale than single-arm data, creating a bottleneck for further improvement (Song et al., 2026). Large-scale robot datasets provide a foundation for learning reusable manipulation skills (Open X-Embodiment Collaboration, 2024; Walke et al., 2023; Khazatsky et al., 2024). Many bimanual behaviors, however, build on existing single-arm action priors: a handover still relies on picking and moving, alongside cross-arm coordination. This motivates compositional approaches that reuse pretrained single-arm policies rather than relearn the entire perception-to-action mapping. By directing limited bimanual demonstrations toward learning cross-arm interactions, these methods improve the data efficiency of bimanual policy learning.

Existing compositional bimanual policies reuse pretrained single-arm skills in different ways. AnyBimanual transfers unimanual capabilities to bimanual tasks by scheduling pretrained single-arm skills (Lu et al., 2025), whereas TwinVLA directly reuses two pretrained SingleVLA branches and enables cross-arm information exchange through dense joint attention across intermediate layers (Im et al., 2026) (Figure 1a). However, existing compositional approaches mainly focus on establishing connections between the two single-arm policies, without clearly separating arm-specific task adaptation from cross-arm coordination. Two key aspects of coordination also remain less explicitly modeled: *what information should be exchanged between the arms, and how should that information influence the action representation of the other arm?*

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.

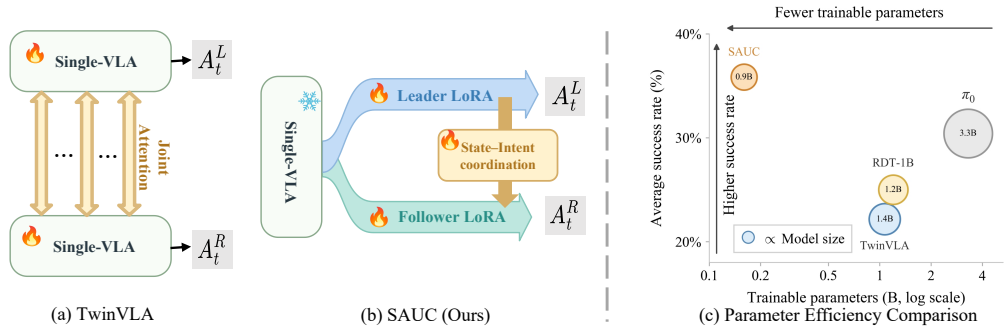


Figure 1: **Architecture and parameter efficiency.** (a) TwinVLA; (b) SAUC with a shared frozen backbone and separate adapters; (c) mean success across six tasks and Easy/Hard settings versus trainable parameters. Bubble sizes indicate total parameters; flames and snowflakes mark trainable and frozen components.

To address this challenge, we propose *Separate Adaptation, Unidirectional Coordination* (SAUC). As illustrated in Figure 1b, SAUC reuses a shared frozen single-arm VLA backbone and equips each arm with an independent Low-Rank Adaptation (LoRA) module (Hu et al., 2021), separating arm-specific adaptation from cross-arm coordination. The two branches take asymmetric roles in information flow: the *Leader* sends information, and the *Follower* uses it to adjust its action representation. After arm-local computation, the State–Intent coordinator uses the Follower context to select between the Leader's contextualized state and intent representations. It then models the Leader–Follower relation to modulate the selected information and fuses the resulting message into the Follower's action readout. The Leader retains its local decision path, while the Follower coordinates its action using the received information together with its own adapted action prior.

Extensive experiments on six tasks in RoboTwin 2.0 (Chen et al., 2025) show that SAUC effectively preserves pretrained single-arm action priors while learning cross-arm coordination from limited bimanual demonstrations. Among the compared pretrained policies, SAUC achieves the highest mean success across both settings with the fewest trainable parameters. As shown in Figure 1(c), it uses only about $1/6$ of TwinVLA's trainable parameters while improving the mean success rate by absolute margins of 13.33% and 14.00% under the Easy and Hard settings, respectively. Compared with monolithic policies pretrained on large-scale bimanual demonstrations, SAUC requires only 50 bimanual demonstrations per task and achieves absolute gains of 5.00% and 5.33% in the two settings, respectively. Real-world experiments further demonstrate improvements on both single-arm and bimanual manipulation tasks.

In summary, this work makes the following contributions:

- We propose SAUC, a compositional bimanual framework that decomposes bimanual policy learning into arm-specific adaptation and cross-arm coordination. A shared frozen single-arm VLA provides reusable action priors, which are specialized by independent arm adapters and subsequently coordinated through a lightweight coordinator.
- We design a lightweight coordinator that uses the Follower context to select and modulate the Leader's state and intent representations before fusing it into the Follower's intent representation, enabling explicit cross-arm coordination with negligible parameter overhead.
- On six RoboTwin 2.0 tasks and real-world bimanual tasks, SAUC uses only about $1/6$ of the trainable parameters of the state-of-the-art compositional policy while outperforming it by 13.33% and 14.00% under the Easy and Hard settings, respectively. With only 50 bimanual demonstrations per task, SAUC also outperforms monolithic pretrained policies by 5.00% and 5.33%.

2 Related Work

Single-Arm Policy Foundations. Robot policies learn from demonstrations and transfer web-pretrained knowledge to actions (Brohan et al., 2022; 2023), with multimodal task condition-

ing (Driess et al., 2023; Jiang et al., 2022) and pretrained visual representations (Zhai et al., 2023; Oquab et al., 2023). Complementary approaches model action sequences (Shafiullah et al., 2022), use language-mediated action hierarchies (Belkhale et al., 2024), or exploit 3D scene structure (Shridhar et al., 2022; Goyal et al., 2023; Ze et al., 2024). SAUC focuses on adapting a pretrained single-arm policy and coordinating the resulting arm-specific branches.

Monolithic Bimanual Policies. Monolithic policies predict actions for both arms through a joint policy, learning manipulation skills and cross-arm dependencies together. ACT and Mobile ALOHA learn from bimanual demonstrations using action chunks (Zhao et al., 2023; Fu et al., 2024). Diffusion-based action generation (Chi et al., 2024; Ho et al., 2020) and Transformer denoisers (Peebles & Xie, 2022) also underpin pretrained policies: RDT-1B uses a diffusion transformer and a unified action representation to learn from heterogeneous robot datasets (Liu et al., 2025), while π_0 combines a pretrained vision–language model with flow matching and trains on data from single-arm, dual-arm, and mobile robots (Black et al., 2025). These approaches advance joint action prediction, but leave the acquisition of bimanual behavior within a unified policy. We instead build on an existing single-arm action prior and learn arm-specific adaptation and cross-arm coordination from limited bimanual demonstrations.

Compositional Bimanual Policies. Compositional approaches aim to reuse pretrained single-arm capabilities and use limited bimanual supervision to learn cross-arm coordination. AnyBimanual coordinates pretrained unimanual policies through a skill manager and a visual aligner (Lu et al., 2025), EnergyAction composes unimanual energy functions with spatial and temporal coordination constraints (Song et al., 2026), while TwinVLA connects two pretrained SingleVLA branches through intermediate joint attention (Im et al., 2026), with its analysis further showing that dense interaction can dilute shared information across layers. Despite their different composition mechanisms, these methods leave *what information should be exchanged and how it should affect the other arm's action representation* less explicitly modeled. In contrast, SAUC explicitly decouples arm-specific adaptation from cross-arm coordination. Independent adapters specialize a shared frozen single-arm VLA backbone for each arm, while cross-arm interaction is introduced only after arm-local contextual computation through a lightweight coordinator, enabling bimanual coordination to be learned directly over existing single-arm VLA representations.

Cross-Arm Coordination Mechanisms. Existing bimanual methods differ in how they assign arm roles and in the information used to couple the two action policies. Stabilize to Act and VoxAct-B organize bimanual manipulation around acting and stabilizing roles (Grannen et al., 2023; Liu et al., 2024). InterACT uses hierarchical attention and decoder synchronization, conditioning each arm on predictions from the other arm to refine its actions (Lee et al., 2025). DIF instead decouples the two controllers and selectively scales and aligns partner-state features according to the receiver's observation and state (Jiang et al., 2025). However, its cross-arm interaction is largely restricted to state features and does not explicitly model action-level intent, limiting its ability to directly capture what the partner arm is about to do. SAUC extends the exchanged information from state features to action-level intent. The Follower context adaptively selects between the Leader's contextual state and action-readout intent, while the Leader–Follower relation further modulates the selected information before a unidirectional message updates the Follower's action readout. Thus, our Leader–Follower roles specify the direction of information flow rather than predefined acting or stabilizing behaviors.

3 METHOD

3.1 PRELIMINARIES

Single-arm action prior. We build on the pretrained SingleVLA released with TwinVLA (Im et al., 2026). SingleVLA couples a vision–language backbone with a continuous action head and is pretrained on approximately 800 hours of single-arm demonstrations from an Open X-Embodiment subset (Open X-Embodiment Collaboration, 2024). TwinVLA aligns heterogeneous demonstrations through absolute end-effector control and frequency matching, providing a transferable perception-to-action prior. We reuse these pretrained components to initialize bimanual learning. For an indi-

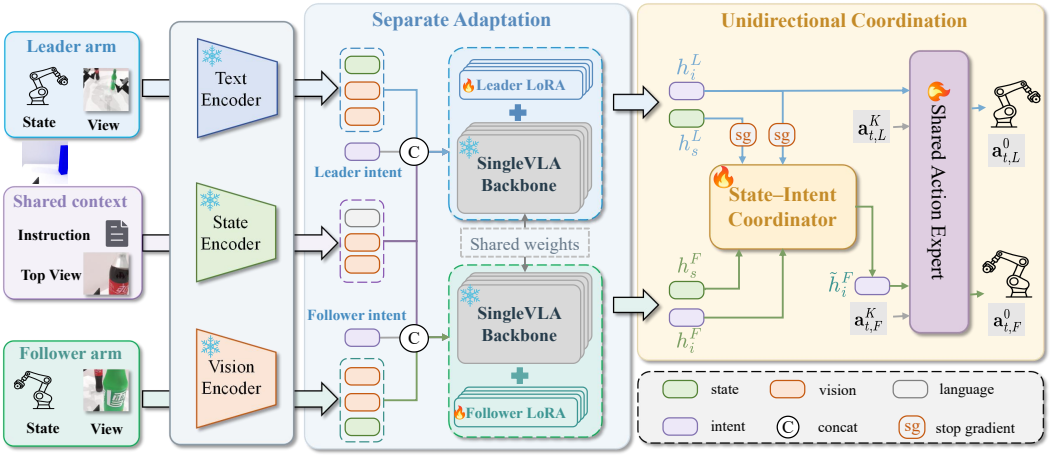


Figure 2: **Architecture of SAUC.** Two arm-specific branches share a frozen SingleVLA backbone and use independent LoRA adapters for separate adaptation. A unidirectional State–Intent coordinator uses stop-gradient Leader representations to update only the Follower's intent representation; the original Leader intent representation and the coordinated Follower intent representation are then decoded by a shared action expert.

vidual arm a, the inherited policy maps its observation o_t^a at timestep t to an action chunk A_t^a:

$$A_t^a = \pi_{\text{single}}(o_t^a)\,. \tag{1}$$

The policy π_{single} denotes the action-generation mapping, with sampling noise omitted for brevity. A learnable readout token summarizes the observation for a conditional-flow action head. This single-arm interface provides the starting point for learning a bimanual policy.

Compositional Bimanual Policy. We learn bimanual manipulation from paired observation–action demonstrations. The two arms $a \in \{\text{left}, \text{right}\}$ pursue the same language-specified task. At timestep t, each receives common context $c_t = (\ell, I_t^c)$, comprising an instruction and main-view image, and private observations $p_t^a = (I_t^a, s_t^a)$, comprising its wrist view and proprioception. The arm-local observation is $o_t^a = (c_t, p_t^a)$. Each arm predicts an action chunk $A_t^a = [a_t^a, \ldots, a_{t+H-1}^a]^\top$, where H is the action chunk length. A bimanual policy maps the two arms' observations to their respective action chunks:

$$(A_t^{\text{left}}, A_t^{\text{right}}) = \pi_{\text{bimanual}}\left(o_t^{\text{left}}, o_t^{\text{right}}\right), \tag{2}$$

where $\pi_{\text{bimanual}} = \text{Compose}(\pi_{\text{left}}; \psi; \pi_{\text{right}})$ denotes the compositional bimanual policy. The operator Compose combines the left arm's single-arm policy π_{left} and the right arm's single-arm policy π_{right} through a coordination mechanism with learnable parameters ψ.

3.2 Overall Architecture

Figure 2 shows two arm-specific branches sharing a pretrained single-arm VLA backbone. The Leader supplies coordination information; the Follower combines it with its own context. A shared action head decodes the original Leader intent and coordinated Follower intent into action chunks.

Separate Adaptation. Although both arms share a pretrained single-arm backbone as their action prior, their local states and observations differ. Each arm therefore needs to learn specialized knowledge on top of the shared prior. We equip each arm with an independent LoRA adapter (Hu et al., 2021), giving the backbones of the Leader and Follower policies the effective parameters

$$\theta_L = \theta_{\text{share}} + \Delta\theta_L, \qquad \theta_F = \theta_{\text{share}} + \Delta\theta_F. \tag{3}$$

The parameters θ_L and θ_F denote the effective backbone parameters of π_L and π_F. The shared pretrained parameters θ_{share} remain frozen, while $\Delta\theta_L$ and $\Delta\theta_F$ are the arm-specific weight updates

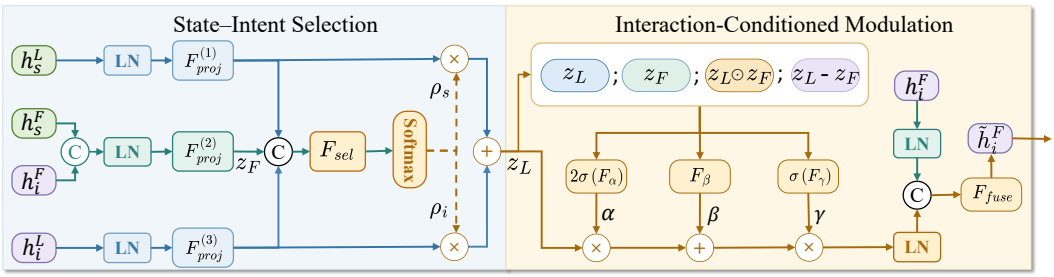


Figure 3: **State–Intent coordinator in SAUC.** Left: the Follower context determines the relative contributions of the Leader's state and intent representations, producing a selected message z_L. Right: the Leader–Follower interaction is used to predict channel-wise scales α, shifts β, and a scalar gate γ to modulate the message, which is then fused with the Follower intent representation.

induced by their trainable LoRA adapters. After multiple attention layers (Vaswani et al., 2017), the two branches produce each arm's state and intent representations:

$$(h_s^L, h_i^L) = \pi_L(o_t^L), \qquad (h_s^F, h_i^F) = \pi_F(o_t^F), \tag{4}$$

where π_L and π_F denote the Leader and Follower branches, respectively. For each arm, $h_s^a \in \mathbb{R}^D$ represents its current state in the visual and language context, while $h_i^a \in \mathbb{R}^D$ represents its action intent, where $D = 896$ is the backbone's hidden dimension. These representations provide the subsequent coordinator with state and intent information from both arms.

Unidirectional Coordination. The branches encode their observations independently. Bidirectional interaction in TwinVLA (Im et al., 2026) and DIF (Jiang et al., 2025) couples their representations and may introduce cross-arm interference. SAUC uses unidirectional exchange to preserve the Leader's local decision path and avoid reciprocal coupling.

The Leader–Follower assignment remains fixed within each run. The coordinator C_ψ selects and modulates the Leader's state and intent using the Follower context:

$$\widetilde{h}_i^F = C_\psi\big(\mathrm{sg}(h_s^L), \mathrm{sg}(h_i^L), h_s^F, h_i^F\big)\,, \tag{5}$$

Here, $\widetilde{h}_i^F$ is the coordinated Follower intent, and sg blocks the Follower loss from updating the Leader through the message path. The shared action expert decodes the original Leader intent and coordinated Follower intent:

$$A_t^L = \mathrm{ActionExpert}_\phi\big(a_{t,L}^K, h_i^L, s_t^L\big)\,, \qquad A_t^F = \mathrm{ActionExpert}_\phi\Big(a_{t,F}^K, \widetilde{h}_i^F, s_t^F\Big)\,, \tag{6}$$

where A_t^L and A_t^F denote the action chunks generated for the Leader and Follower arms, respectively.

3.3 State–Intent Coordinator

State–Intent Selection. The Leader's state describes its current situation, whereas its intent summarizes information for the next action chunk. Their relevance depends on the Follower context. We project all three sources into a common message space (Figure 3, left):

$$z_s = F_{\mathrm{proj}}^{(1)}(\mathrm{LN}(\mathrm{sg}(h_s^L))), \quad z_i = F_{\mathrm{proj}}^{(3)}(\mathrm{LN}(\mathrm{sg}(h_i^L))), \quad z_F = F_{\mathrm{proj}}^{(2)}(\mathrm{LN}([h_s^F\,; h_i^F])). \tag{7}$$

All projected features satisfy $z_s, z_i, z_F \in \mathbb{R}^d$, with $d = 128$; LN denotes layer normalization. Their concatenation determines the state and intent weights:

$$(\rho_s, \rho_i) = \mathrm{softmax}(F_{\mathrm{sel}}([z_s\,; z_i\,; z_F]))\,, \tag{8}$$

where ρ_s and ρ_i are the weights assigned to the Leader's state and intent representations, respectively. We use these weights to fuse the two projected representations into z_L:

$$z_L = \rho_s z_s + \rho_i z_i. \tag{9}$$

The context-dependent mixture z_L specifies *what information to coordinate*; interaction-conditioned modulation then determines how it updates the Follower intent.

Interaction-Conditioned Modulation. The same Leader message may require different responses under different Follower states. We represent their interaction using the message, Follower context, and their element-wise product and difference (Figure 3, right):

$$u = [z_L \,;\, z_F \,;\, (z_L \odot z_F) \,;\, (z_L - z_F)] \in \mathbb{R}^{4d}. \tag{10}$$

The operator $\odot$ denotes element-wise multiplication. Three independent affine heads then map the interaction representation to a scalar gating factor, channel-wise scaling coefficients, and channel-wise shifts, respectively:

$$\gamma = \sigma(F_\gamma(u)), \qquad \alpha = 2\sigma(F_\alpha(u)), \qquad \beta = F_\beta(u). \tag{11}$$

Here, σ is the element-wise sigmoid, $\gamma \in (0, 1)$ is a scalar factor, $\alpha \in (0, 2)^d$ gives channel-wise scales, and $\beta \in \mathbb{R}^d$ gives shifts without a sigmoid. These interaction-conditioned factors produce the modulated message:

$$m = \gamma\,(\alpha \odot z_L + \beta). \tag{12}$$

To combine the Follower's local information with the message $m \in \mathbb{R}^d$, we normalize each separately, concatenate them, and project back to the intent space:

$$\widetilde{h}_i^F = F_{\text{fuse}}\big([\text{LN}(h_i^F) \,;\, \text{LN}(m)]\big)\,, \tag{13}$$

The resulting $\widetilde{h}_i^F$ combines local intent with selected and modulated Leader information.

3.4 Training Objective

We train both arms with a conditional flow-matching objective (Im et al., 2026; Lipman et al., 2022). For arm $a \in \{L, F\}$, the shared action expert predicts a velocity from the noisy action chunk $A_t^{a,\tau}$, conditioned on its intent representation $\widetilde{h}_{i,t}^a$ and proprioception s_t^a. The per-arm loss is

$$\mathcal{L}_a = \mathbb{E}_{\mathcal{D},\tau,\epsilon^a}\left[\frac{1}{Hd_a}\left\|v_\phi\Big(A_t^{a,\tau}, \tau; \widetilde{h}_{i,t}^a, s_t^a\Big) - (\epsilon^a - A_t^a)\right\|_{\text{F}}^2\right], \tag{14}$$

where ϵ^a is Gaussian noise, Hd_a is the number of action coordinates in a chunk, and $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm. The conditions are the original Leader intent, $\widetilde{h}_{i,t}^L = h_{i,t}^L$, and the coordinated Follower intent, $\widetilde{h}_{i,t}^F$. We average the two arm losses:

$$\mathcal{L} = \frac{1}{2}\left(\mathcal{L}_L + \mathcal{L}_F\right). \tag{15}$$

4 Experiments

4.1 Experimental setup

Benchmark. We conduct extensive evaluations on six tasks from the RoboTwin 2.0 benchmark (Chen et al., 2025), covering four bimanual and two single-arm tasks. We report success rates under the Easy and Hard settings and average them equally across tasks. The benchmark reports 100 evaluation episodes per condition. The tasks assess complementary capabilities, including parallel object manipulation, inter-arm transfer, sequential cooperation, and single-arm manipulation. For fair comparison, SAUC and the reported baselines use 50 clean demonstrations per task for training. We initialize SAUC from the pretrained SingleVLA checkpoint released with TwinVLA (Im et al., 2026). Detailed task descriptions and their evaluation focus are provided in Appendix B.1. Further training and implementation details are provided in the appendix.

Baselines. We evaluate SAUC against five manipulation policies spanning monolithic and compositional approaches. Diffusion Policy (DP) (Chi et al., 2024) learns joint action sequences through conditional denoising, while ACT (Zhao et al., 2023) predicts joint action chunks with a Transformer; both learn from task-specific demonstrations without large-scale robot pretraining. RDT-1B (Liu et al., 2025) is a monolithic diffusion Transformer pretrained on robotic data (∼2,400h

Table 1: **Six-task comparison on RoboTwin 2.0.** Success rates (%) under Easy and Hard settings. Gray cells show task averages; bold and underlined values mark the best and second-best results, including ties. Task selection and descriptions are given in Appendix B.

Method	Overall		Bimanual Manipulation								Single-arm Manipulation			
	Average		Pick Dual Bottles		Place Dual Shoes		Handover Block		Place Object Basket		Open Laptop		Move Can Pot	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
Monolithic Bimanual Policies														
DP	24.17%	0.00%	24%	0%	8%	0%	10%	0%	15%	0%	49%	0%	39%	0%
ACT	29.17%	0.67%	31%	0%	9%	0%	42%	0%	15%	0%	56%	0%	22%	4%
RDT-1B	34.67%	15.33%	42%	13%	4%	**4%**	45%	**14%**	33%	**17%**	59%	32%	25%	12%
π_0	46.00%	14.83%	57%	12%	15%	0%	45%	8%	16%	2%	**85%**	46%	**58%**	21%
Compositional Bimanual Policies														
TwinVLA	37.67%	6.67%	18%	12%	18%	3%	17%	0%	**48%**	3%	80%	17%	45%	5%
SAUC (ours)	**51.00%**	**20.67%**	**60%**	**20%**	**19%**	3%	**58%**	9%	36%	8%	78%	**52%**	55%	**32%**

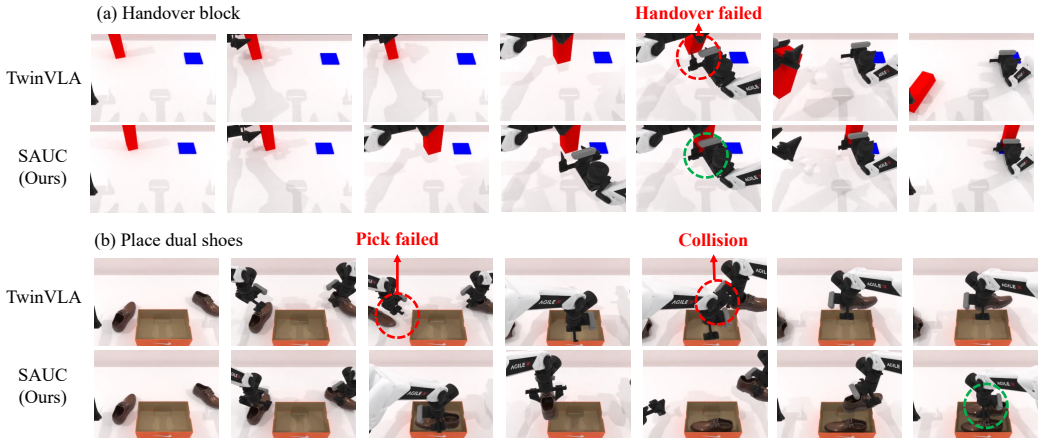


Figure 4: **Visualization in RoboTwin 2.0.** The top two rows show Handover Block and the bottom two rows show Place Dual Shoes, with rollouts progressing from left to right. For each task, TwinVLA is shown above and SAUC below. Red dashed circles highlight TwinVLA failures, while green dashed circles mark key moments of successful coordination by SAUC.

data) including bimanual demonstrations. π_0 (Black et al., 2025) combines a pretrained vision–language backbone with a flow-matching action expert and uses over 10,000h of cross-embodiment robot pretraining data. Our primary compositional baseline, TwinVLA (Im et al., 2026), duplicates a single-arm backbone pretrained on single-arm demonstrations (∼800h data) and coordinates its two branches through intermediate joint attention while sharing the vision encoder and action head. Building on this baseline, we explore more effective bimanual architectures and coordination mechanisms.

4.2 Simulation Results

(i) Overall performance. SAUC achieves the highest average success rate across the six tasks, reaching 51.00% on Easy and 20.67% on Hard (Table 1). Compared with the strongest monolithic baseline in each setting, SAUC improves performance by 5.00% over π_0 on Easy and by 5.33% over RDT-1B on Hard. Relative to TwinVLA, the improvements are 13.33% and 14.00%, respectively. Meanwhile, SAUC contains only 0.16B trainable parameters, compared with 0.94B for TwinVLA, i.e., roughly one sixth as many. Figure 1c summarizes this comparison using the equal-weight mean of the Easy and Hard task averages. DP and ACT perform worse than the pretrained

policies in both settings, further highlighting the benefit of reusing pretrained action priors when task-specific demonstrations are limited. **(ii) Bimanual manipulation.** On the four bimanual tasks, SAUC achieves the best Easy performance on Pick Dual Bottles (60%), Place Dual Shoes (19%), and Handover Block (58%), and also obtains the highest Hard success on Pick Dual Bottles at 20%, compared with 12% for TwinVLA and 13% for RDT-1B. On the more temporally dependent Handover Block task, SAUC reaches 9% on Hard, outperforming TwinVLA (0%) and π_0 (8%), while remaining below RDT-1B (14%). These results show that the overall gains extend across multiple forms of bimanual interaction. **(iii) Single-arm manipulation.** SAUC achieves the highest Hard success on both Open Laptop and Move Can Pot, reaching 52% and 32%, respectively. These results exceed TwinVLA by 35% and 27%, and π_0 by 6% and 11%, respectively. Although π_0 performs better on both tasks under Easy, SAUC retains competitive single-arm manipulation performance while supporting cross-arm coordination.

Qualitative Results. Figure 4 compares rollout sequences on two bimanual tasks. In Handover Block, TwinVLA fails during object transfer, whereas SAUC successfully completes the handover and places the block at the target location. In Place Dual Shoes, TwinVLA fails to maintain a stable grasp with the left arm and encounters a collision during placement. In contrast, SAUC waits until the partner arm leaves the workspace before placing the shoe into the box, successfully completing the sequentially dependent coordination. These examples qualitatively illustrate the cross-arm object transfer and coordinated placement behaviors captured by the two tasks.

4.3 Ablation study

Table 2: **Design ablation on six RoboTwin 2.0 tasks.** Success rates (%) with separate adaptation (SA) and State–Intent coordination (SIC). Gray cells show task averages; bold and underlined values mark the best and second-best results, including ties.

SA	SIC	Overall		Bimanual Manipulation								Single-arm Manipulation			
		Average		Pick Dual Bottles		Place Dual Shoes		Handover Block		Place Object Basket		Open Laptop		Move Can Pot	
		Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
✗	✗	39.17%	13.17%	29%	12%	13%	1%	21%	0%	**40%**	3%	68%	40%	**64%**	23%
✓	✗	39.50%	17.17%	38%	12%	**22%**	0%	22%	0%	38%	7%	**79%**	51%	38%	**33%**
✓	✓	**51.00%**	**20.67%**	**60%**	**20%**	19%	**3%**	**58%**	**9%**	36%	**8%**	78%	**52%**	55%	32%

Design Ablation. Table 2 evaluates the effects of Separate Adaptation (SA) and the State–Intent Coordinator (SIC) across all six tasks. Starting from a shared-LoRA baseline, introducing SA increases the average success rate from 39.17% to 39.50% on Easy and from 13.17% to 17.17% on Hard. Adding SIC on top of SA further improves the averages to 51.00% and 20.67%, respectively. The larger gain from SA under the Hard setting suggests that arm-specific specialization becomes increasingly important under more challenging conditions. The additional improvement brought by SIC further indicates that, after specialization, explicit information exchange between the two arm-specific branches remains beneficial. The clearest task-level gain from SIC is on Handover Block, where Easy/Hard success rises from 22%/0% to 58%/9%. We therefore examine the architecture and coordination mechanism on this task below.

Backbone Ablation. On Handover Block (Table 3), Shared-2L64, which combines one pretrained backbone with two arm-specific rank-64 LoRA adapters, achieves the highest Easy/Hard success rates (58%/9%), outperforming Independent (separate backbones; 35%/7%) and both shared-adapter variants, Shared-L64 and Shared-L128 (ranks 64 and 128, respectively), with the decline to 16%/0% for Shared-L128 suggesting that preserving a reusable single-arm prior while allocating separate adaptation capacity to each arm better balances general knowledge and arm-specific specialization than increasing shared adaptation capacity (see Appendix C for architecture definitions and further analysis).

Table 3: **Backbone design.**

Backbone	Easy	Hard
Independent	35%	7%
Shared-L64	40%	3%
Shared-L128	16%	0%
Shared-2L64	**58%**	**9%**

Table 4: **Coordination method.**

Method	Easy	Hard
None	22%	0%
SIM (DIF)	23%	2%
SIC w/o detach	31%	5%
SIC w/ detach	**58%**	**9%**

Table 5: **Message content.**

Message	Easy	Hard
Intent-only	7%	2%
State-only	45%	**10%**
Equal-weight	39%	4%
Adaptive (SIS)	**58%**	9%

How to Coordinate? Table 4 compares no explicit coordination (None), DIF's Selective Interaction Module (SIM) (Jiang et al., 2025), and the State–Intent Coordinator (SIC) with and without source detachment. Under the Easy/Hard settings, None, SIM, and SIC without detachment achieve 22%/0%, 23%/2%, and 31%/5%, respectively, while applying stop-gradient further improves SIC to 58%/9%. These results show that establishing a cross-arm communication pathway alone is insufficient; preserving Leader-to-Follower forward information flow while blocking backward gradient interference creates a clearer optimization boundary and leads to more effective cross-arm coordination.

What to Communicate? Table 5 compares four message compositions: state-only, intent-only, equal-weight, and adaptive State–Intent Selection (SIS). State-only exchange achieves Easy/Hard success rates of 45%/10%, substantially outperforming intent-only exchange at 7%/2% and indicating the value of contextual state information in this task. Fixed equal-weight fusion achieves 39%/4%, whereas adaptive SIS reaches 58%/9%, yielding absolute improvements of 19% and 5%, respectively. Adaptive selection achieves the highest success rate on Easy, while state-only exchange retains an absolute 1% advantage on Hard. Thus, when combining state and intent information, learned adaptive selection provides better overall performance than fixed equal-weight mixing.

Real-world Evaluation. We further compare SAUC with TwinVLA on three real-world manipulation tasks (Table 6). Pick and Place Mouse (R1) is a single-arm task, while Wipe Blackboard (R2) and Lint Roller Remove Dirt (R3) require bimanual coordination. Across 20 trials per task, SAUC consistently achieves higher success rates than TwinVLA on all three tasks, with average success rates of 66.67% and 35.00%, respectively. These results further show that the advantages of SAUC observed in simulation extend to both single-arm and bimanual manipulation in the real world. Hardware specifications are provided in Appendix D.

Table 6: **Real-world experiment.**

Method	Average	R1	R2	R3
TwinVLA	35%	7/20	8/20	6/20
SAUC (ours)	**66.67%**	**17/20**	**14/20**	**9/20**

5 Conclusion

We studied how to effectively reuse a pretrained single-arm VLA for bimanual manipulation when bimanual demonstrations are limited. We introduced SAUC, which separates arm-specific adaptation from cross-arm coordination: a shared single-arm VLA provides reusable action priors, independent adapters specialize them for the two arms, and a unidirectional State–Intent coordinator selectively transfers Leader information to the Follower. This design preserves arm-specific decision paths while introducing cross-arm interaction only where coordination is needed. By avoiding dense coupling throughout the backbone, SAUC keeps specialization and coordination as distinct learning processes. With only 50 bimanual demonstrations per task, SAUC achieves the highest average success in the six-task RoboTwin 2.0 comparison while using only about $1/6$ of TwinVLA's trainable parameters, and the real-world experiments further confirm its effectiveness on both single-arm and bimanual manipulation. Our ablations show that arm specialization and explicit coordination provide complementary gains, and that the content, direction, and gradient boundary of cross-arm communication all matter for effective coordination. Overall, these results show that combining reusable single-arm action priors with lightweight, structured cross-arm interaction provides a data- and parameter-efficient approach to bimanual policy learning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with coding and debugging under our guidance, provide feedback on methodological and experimental details, and support result interpretation. We developed the core research idea ourselves and led the method design, experimental design, and execution. We did not use AI for autonomous end-to-end research.

Additionally, we used generative AI tools for literature review and manuscript editing. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

The experiments include simulated and real-world manipulation tasks. Safe physical deployment requires collision checking, workspace limits, emergency stopping, and supervised validation. Releases must respect the licenses of the pretrained model, benchmark, and software dependencies.

REPRODUCIBILITY STATEMENT

Section 3 specifies the model and gradient boundaries. Appendix A gives the available configuration, and Table 2 reports per-task ablation results. The accompanying manuscript materials retain the numerical records and their provenance. Run-level configuration exports, accepted evaluation seeds, and an anonymized implementation archive remain to be assembled for a reproducibility release.

REFERENCES

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024. URL https://arxiv.org/abs/2403.01823.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusi, 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. In *Proceedings of Robotics: Science and Systems*, 2025. URL https://www.roboticsproceedings.org/rss21/p010.pdf.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

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

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, et al. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023. URL https://arxiv.org/abs/2303.03378.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024. URL https://arxiv.org/abs/2401.02117.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3D object manipulation. *arXiv preprint arXiv:2306.14896*, 2023. URL https://arxiv.org/abs/2306.14896.

Jennifer Grannen, Yilin Wu, Brandon Vu, and Dorsa Sadigh. Stabilize to act: Learning to coordinate for bimanual manipulation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 563–576. PMLR, 2023. URL https://proceedings.mlr.press/v229/grannen23a.html.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020. URL https://arxiv.org/abs/2006.11239.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Hokyun Im, Euijin Jeong, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. TwinVLA: Data-efficient bimanual manipulation with twin single-arm vision-language-action models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jG9W6nAwVz.

Jian-Jian Jiang, Xiao-Ming Wu, Yi-Xiang He, Ling-An Zeng, Yi-Lin Wei, Dandan Zhang, and Wei-Shi Zheng. Rethinking bimanual robotic manipulation: Learning with decoupled interaction framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12427–12437, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Jiang_Rethinking_Bimanual_Robotic_Manipulation_Learning_with_Decoupled_Interaction_Framework_ICCV_2025_paper.html.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022. URL https://arxiv.org/abs/2210.03094.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024. URL https://arxiv.org/abs/2403.12945.

Franziska Krebs and Tamim Asfour. A bimanual manipulation taxonomy. *IEEE Robotics and Automation Letters*, 7(4):11031–11038, 2022. doi: 10.1109/LRA.2022.3196158. URL https://publikationen.bibliothek.kit.edu/1000150129.

Andrew Choong-Won Lee, Ian Chuang, Ling-Yuan Chen, and Iman Soltani. InterACT: Inter-dependency aware action chunking with hierarchical attention transformers for bimanual manipulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 1730–1743. PMLR, 2025. URL https://proceedings.mlr.press/v270/lee25a.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

I-Chun Arthur Liu, Sicheng He, Daniel Seita, and Gaurav Sukhatme. VoxAct-B: Voxel-based acting and stabilizing policy for bimanual manipulation. *arXiv preprint arXiv:2407.04152*, 2024. URL https://arxiv.org/abs/2407.04152.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/49f80e4d2471ad4f2edf4f5f1ab62339-Paper-Conference.pdf.

Guanxing Lu, Tengbo Yu, Haoyuan Deng, Season Si Chen, Yansong Tang, and Ziwei Wang. AnyBimanual: Transferring unimanual policy for general bimanual manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13662–13672, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Lu_AnyBimanual_Transferring_Unimanual_Policy_for_General_Bimanual_Manipulation_ICCV_2025_paper.html.

Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation*, 2024. URL https://rpl.cs.utexas.edu/publications/2024/05/13/rtx-icra24-rtx/.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. URL https://arxiv.org/abs/2304.07193.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022. URL https://arxiv.org/abs/2212.09748.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022. URL https://arxiv.org/abs/2206.11251.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022. URL https://arxiv.org/abs/2209.05451.

Mingchen Song, Xiang Deng, Jie Wei, Dongmei Jiang, Liqiang Nie, and Weili Guan. EnergyAction: Unimanual to bimanual composition with energy-based models. *arXiv preprint arXiv:2603.20236*, 2026. URL https://arxiv.org/abs/2603.20236.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. URL https://arxiv.org/abs/1706.03762.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. *arXiv preprint arXiv:2308.12952*, 2023. URL https://arxiv.org/abs/2308.12952.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. *arXiv preprint arXiv:2403.03954*, 2024. URL https://arxiv.org/abs/2403.03954.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*, 2023. URL https://arxiv.org/abs/2303.15343.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

A Implementation Details

Training setup. The visual encoder and base VLM remain frozen; we jointly train two LoRA adapters with rank $r = 64$ and scaling parameter $\eta = 128$, the State–Intent coordinator, and the shared action head. The main view is encoded once and its embeddings are reused in two adapter-specific Transformer forwards. Each branch yields state and intent representations of dimension $D = 896$. Each task is trained on two RTX 3090 GPUs with a per-device batch size of four. We use a learning rate of 2×10^{-4} with warmup followed by cosine decay, and predict action chunks of length $H = 20$.

Table 7: Model dimensions and recorded default training configuration for SAUC. Run-specific settings should be resolved from the corresponding checkpoint and evaluation manifests.

Setting	Value
Starting model components	SingleVLA from `jellyho/TwinVLA`
Backbone sharing	One base VLM; two adapter-specific forwards
Backbone hidden size / depth	896 / 24 layers
LoRA rank / alpha / dropout	64 / 128 / 0.01
Vision encoder and base VLM	Frozen
Trainable arm parameters	Both LoRAs throughout training
Action head	Shared and trainable
Coordinator insertion	Final readout only
Default direction	Left Leader $\rightarrow$ right Follower
Sender message	Contextual state + readout, detached
Message dimension	128
Relation-vector dimension	512
Fusion input / hidden / output	1024 / 896 / 896
Selector initialization	$(0.5, 0.5)$
Scalar / channel-scale / shift initialization	0.5 / $\mathbf{1}$ / $\mathbf{0}$
Auxiliary intent or stage supervision	None
Residual or identity-preserving fusion	None
Action prediction horizon	20
Training demonstrations	50 Clean per task
Recorded seed / run / endpoint	42 / run0 / 50,000 updates
GPUs / per-device batch	2 / 4
Effective batch (default, no accumulation)	8
Linearly scaled base learning rate	2×10^{-4}
Learning-rate schedule	Inherited warmup + cosine decay
Loss weighting	$(\mathcal{L}_L + \mathcal{L}_F)/2$

Forward-pass recipe. Encode the main view once and reuse its embeddings. Activate the Leader adapter and run the native arm-local sequence. Retain the original readout for its action loss and detach only the copies supplied to the coordinator. Activate the Follower adapter and run its native sequence. Project the summaries, select and modulate the message, and fuse the Follower readout as specified in Section 3.3. Decode the original Leader readout and fused Follower readout with the shared action head. During training, backpropagate the average of the two native action losses. At inference, the message uses observed inputs only and the predicted chunks are supplied to the existing closed-loop evaluator.

Initialization and optimization. The selector starts at $\rho_s = \rho_i = 0.5$, and the modulation heads at $\gamma = 0.5$, $\alpha = \mathbf{1}$, and $\beta = \mathbf{0}$; the fusion MLP uses standard initialization. Both LoRA adapters, the coordinator, and the shared action expert are optimized jointly from the pretrained SingleVLA components, while the vision encoder and base VLM remain frozen. The Leader loss trains its adapter, and the Follower loss trains its adapter and coordinator; both update the shared action expert.

A.1 Conditional Flow Matching

We use conditional flow matching (Lipman et al., 2022) to generate continuous action chunks, following the action head of SingleVLA (Im et al., 2026). For arm $a \in \{L, F\}$, the velocity field is conditioned on the intent representation $\widetilde{h}^a_{i,t}$ and proprioception s^a_t, both fixed during sampling. The Leader uses its original intent representation, whereas the Follower uses the coordinated readout.

Probability path and training target. Let $\mathcal{D}$ denote the observation–action demonstration distribution and $A^a_t \in \mathbb{R}^{H\times d_a}$ a demonstrated action chunk. Independently sample Gaussian noise ϵ^a with standard-normal entries and flow time $\tau \sim p_\tau$. The implementation uses $b \sim \mathrm{Beta}(1.5, 1.0)$ and $\tau = 0.001 + 0.999b$, which reweights the training times along the linear path

$$A^{a,\tau}_t = (1-\tau)A^a_t + \tau\epsilon^a, \qquad \tau \in [0,1]. \tag{16}$$

Under this convention, $\tau = 0$ is the data endpoint and $\tau = 1$ is the Gaussian endpoint. The conditional target velocity is the derivative of the interpolation,

$$u^a_t(A^a_t, \epsilon^a) = \frac{\mathrm{d}A^{a,\tau}_t}{\mathrm{d}\tau} = \epsilon^a - A^a_t. \tag{17}$$

Equation 14 regresses v_ϕ onto this target with the squared Frobenius norm averaged over the Hd_a action coordinates; Equation 15 averages the two arm losses. The time convention is equivalent to a noise-to-data parameterization under $s = 1 - \tau$, for which the target velocity becomes $A^a_t - \epsilon^a$. Reversing time therefore requires reversing the velocity sign as well.

ODE sampling. Starting from Gaussian noise $X^a(1)$, we solve the conditional ordinary differential equation

$$\frac{\mathrm{d}X^a(\tau)}{\mathrm{d}\tau} = v_\phi\Big(X^a(\tau), \tau; \widetilde{h}^a_{i,t}, s^a_t\Big) \tag{18}$$

from $\tau = 1$ to $\tau = 0$. For K sampling steps, let $\tau_k = k/K$ and $X^a_k \approx X^a(\tau_k)$. The implementation uses explicit Euler updates

$$X^a_{k-1} = X^a_k - \frac{1}{K} v_\phi\Big(X^a_k, \tau_k; \widetilde{h}^a_{i,t}, s^a_t\Big), \qquad k = K, \dots, 1. \tag{19}$$

The negative step size follows from integrating toward decreasing flow time, and X^a_0 is the generated action chunk. Figure 2 denotes X^a_K and X^a_0 by $a^K_{t,a}$ and $a^0_{t,a}$, respectively. The $\mathrm{ActionExpert}_\phi$ mapping in Equation 6 summarizes this sampling procedure. Control time t, continuous flow time τ, and discrete solver index k are distinct.

A.2 Communication and Optimization Properties

Parameter and coordination efficiency. The trainable parameters comprise the two arm-specific LoRA adapters, the State–Intent coordinator, and the shared action head, while the pretrained backbone remains frozen:

$$P_{\mathrm{train}} = P_{\Delta_L} + P_{\Delta_F} + P_{\mathrm{coord}} + P_{\mathrm{head}}. \tag{20}$$

Here, each P denotes the number of parameters in the indicated component. The measured trainable parameter counts are 0.16B for SAUC and 0.94B for TwinVLA, corresponding to approximately one sixth as many parameters, or an 83% reduction. The coordinator contributes approximately 2.32M parameters and exchanges a 128-dimensional message computed from a fixed set of state and intent summaries. Both its parameter count and its per-message computation are independent of the number of visual tokens.

Feature reuse and cached adapter dispatch. The main-view embedding is computed once per observation and reused by both arm branches. With E_{vis} denoting the frozen visual encoder and F_{θ,Δ_a} the backbone with arm-specific adapter Δ_a, the computation can be written as

$$e^c_t = E_{\mathrm{vis}}(I^c_t), \qquad (h^a_{s,t}, h^a_{i,t}) = F_{\theta,\Delta_a}(\ell, e^c_t, p^a_t), \quad a \in \{L, F\}. \tag{21}$$

Here, F_{θ,Δ_a} includes private-input embedding and summary extraction. Reusing e^c_t avoids encoding the same main-view image twice. Adapter dispatch caches references to the LoRA-bearing modules:

after a one-time traversal of N model modules, each switch visits only the M cached LoRA modules, reducing module-dispatch work from $O(N)$ to $O(M)$. Both adapters remain trainable.

Cross-arm information is exchanged once at the final readout, before iterative action sampling. The Follower therefore requires the Leader's summaries, rather than its sampled action trajectory:

$$\widetilde{h}^F_{i,t} = \mathrm{SIC}(h^L_{s,t}, h^L_{i,t}, h^F_{s,t}, h^F_{i,t}), \qquad \widetilde{h}^L_{i,t} = h^L_{i,t}. \tag{22}$$

These conditions are retained throughout all flow steps. The current implementation runs the two adapter-specific encodings sequentially, constructs the coordinated readout, and then invokes the shared action head for each arm. Thus, Follower encoding does not wait for completion of Leader action sampling. Once the conditions are available, the two sampling problems have no stepwise cross-arm dependency and can be batched.

B Task Selection and Descriptions

Task selection. We select four bimanual manipulation tasks from RoboTwin 2.0 (Chen et al., 2025) to evaluate cross-arm coordination. We additionally include two single-arm tasks to assess whether adapting the policy for bimanual manipulation preserves its pretrained single-arm capabilities.

B.1 Task Descriptions and Evaluation Focus

The six tasks in Table 1 follow the RoboTwin 2.0 task definitions (Chen et al., 2025). We group them by whether task completion requires both arms or a single active arm. Their manipulation requirements motivate the evaluation focus described below.

Pick Dual Bottles. Each arm grasps a separate bottle and lifts it toward its designated target region. Success requires both bottles to reach the target regions above the required height. This task evaluates parallel object manipulation and the ability to complete two arm-local goals within one episode.

Place Dual Shoes. The arms grasp separate shoes and alternately place them at the corresponding target positions in a shoe box. Both shoes must satisfy the target position and orientation requirements and be released. This task requires temporal coordination: one arm must wait for the partner arm to clear the shared placement region before placing its shoe. It therefore evaluates sequentially dependent placement, including collision avoidance and the timing of alternating arm actions.

Handover Block. One arm picks up a block and transfers it to the other arm, which then places it on a target platform. This task requires asynchronous coordination between arms with distinct roles and action phases. At the handover, the donating arm must accurately perceive the receiving arm's state and time its release to the receiver's secure grasp. Successful completion therefore depends on partner-state awareness, precise grasp–release timing, and subsequent placement by the receiving arm.

Place Object Basket. One arm places an object in a basket, after which the other arm grasps and lifts the basket with the object inside. Successful completion requires the basket and contained object to be lifted together. This task evaluates sequential cooperation between arms with distinct, dependent roles.

Open Laptop. A single arm engages the laptop lid and opens it through its articulated motion. Success depends on the lid opening sufficiently while the end effector remains near the lid's interaction point. This task evaluates single-arm control of an articulated object.

Move Can Pot. A single arm grasps a can and places it beside a pot. Success depends on the can's position relative to the pot, its orientation, and release by the gripper. This task evaluates object-relative placement using one arm.

C Backbone Design Ablation

This appendix details the four configurations in Table 3 and their results on Handover Block. The comparison examines whether the pretrained backbone is shared and whether adaptation parameters are shared across arms. In the shared configurations, L64 and L128 denote LoRA adapters with ranks 64 and 128, respectively; 2L64 denotes two separate rank-64 adapters, one per arm.

C.1 Architecture Descriptions

Independent. Each arm uses a separate backbone instance, maintaining its own branch parameters rather than reusing a single backbone across arms. This configuration provides the reference for independent arm branches. It achieves 35% success on Easy and 7% on Hard.

Shared-L64. Both arms reuse one frozen pretrained backbone and one shared rank-64 LoRA adapter. The inputs remain arm-specific, but both branches use the same adaptation parameters, so their training signals update a common adapter. Shared-L64 achieves 40%/3% Easy/Hard success.

Shared-L128. This configuration retains the shared backbone and shared adapter structure while increasing the adapter rank from 64 to 128. The higher rank expands the shared adaptation capacity but does not introduce arm-specific parameters. Its Easy/Hard success rates are 16%/0%.

Shared-2L64. Both arms reuse the same frozen backbone, while each has an independent rank-64 LoRA adapter. The pretrained weights are shared, while the learned updates can specialize to each arm's observations and role. This is the backbone parameterization used by SAUC and achieves the highest success in Table 3: 58% on Easy and 9% on Hard.

C.2 Analysis of Capacity Allocation

Shared capacity versus arm-specific capacity. Shared-2L64 achieves absolute gains of 18% on Easy and 6% on Hard over Shared-L64. In contrast, increasing the shared adapter rank from 64 to 128 reduces success by absolute margins of 24% and 3%, respectively. The results therefore favor allocating adaptation capacity to individual arms over enlarging one shared adapter. For example, on Handover Block, the donating and receiving arms must learn different grasping, transfer, and placement roles; separate updates allow these roles to specialize without requiring both branches to use the same parameter update.

Parameter allocation at equal LoRA capacity. For a projection with input and output dimensions d_{in} and d_{out}, a rank-r LoRA adapter contains $r(d_{\text{in}} + d_{\text{out}})$ parameters. Consequently, one rank-128 adapter and two rank-64 adapters have the same number of LoRA parameters when applied to the same target projections:

$$128(d_{\text{in}} + d_{\text{out}}) = 2 \times 64(d_{\text{in}} + d_{\text{out}}). \tag{23}$$

Their difference lies in parameter allocation: Shared-L128 learns one common update, whereas Shared-2L64 learns two independently parameterized updates. The 58%/9% versus 16%/0% results support arm-specific allocation rather than shared rank alone as the more effective design in this comparison.

Reusing the pretrained prior. Shared-2L64 also exceeds Independent by absolute margins of 23% on Easy and 2% on Hard. Together, these comparisons suggest that separate backbone instances are unnecessary for effective arm specialization on this task. Sharing the pretrained backbone retains a common single-arm prior, while separate adapters provide distinct adaptation paths for the two roles. The results support this balance of prior reuse and specialization.

D REAL-WORLD HARDWARE PLATFORM

Table 8 summarizes the dual-arm Piper platform used in the real-world experiments.

Table 8: Hardware specifications of our dual-arm Piper platform.

Component	**Specification**
Platform	Tabletop dual-arm manipulation system
Robotic Arms	2× AgileX Piper
Arm DOF	$2 \times 6 = 12$ revolute joints
Grippers	Two parallel-jaw grippers, one per arm
Total Action Dimensions	$2 \times (6 \text{ joints} + 1 \text{ gripper}) = 14$
External Camera	One Orbbec Gemini providing a top view
Wrist Cameras	Two Orbbec Gemini cameras, one per wrist
RGB Input Resolution	640×480 pixels per camera
Robot Communication	CAN bus at 1 Mbit/s

E ADDITIONAL ROLLOUT VISUALIZATIONS

Figures 5–10 show additional rollout examples in simulation and on the physical platform. Each figure contains two separate rollouts, with frames progressing from left to right.

E.1 SIMULATION ROLLOUTS

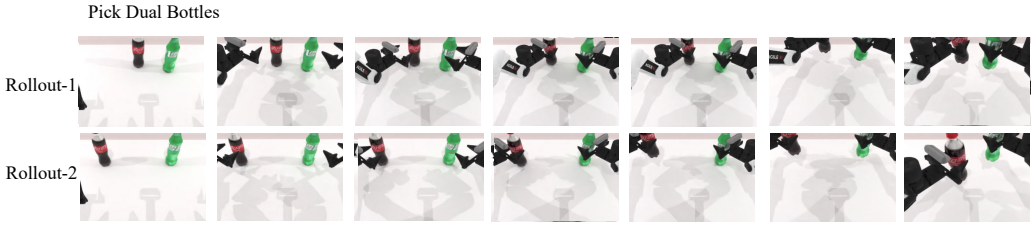


Figure 5: **Pick Dual Bottles in RoboTwin 2.0.** Two rollouts show each arm grasping and lifting a separate bottle.

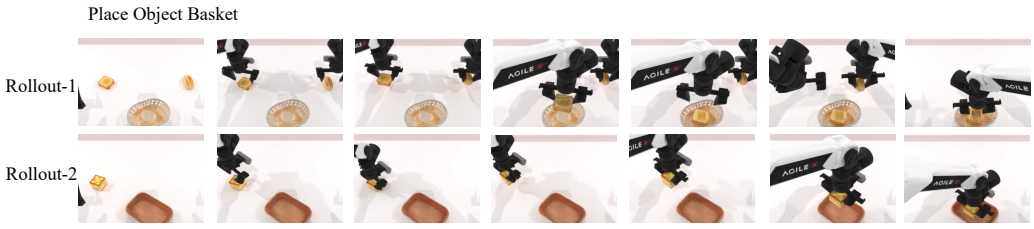


Figure 6: **Place Object Basket in RoboTwin 2.0.** Two rollouts illustrate object pickup, transport, and placement into a basket.

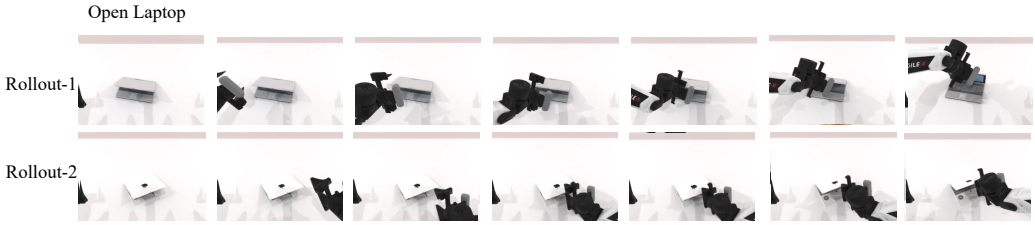


Figure 7: **Open Laptop in RoboTwin 2.0.** Two rollouts show a single arm engaging the laptop lid and opening it.

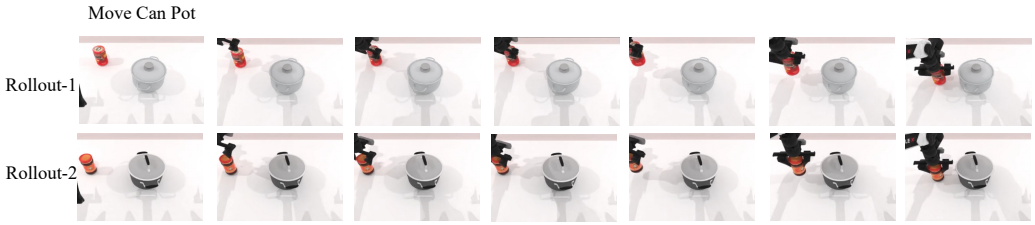


Figure 8: **Move Can Pot in RoboTwin 2.0.** Two rollouts show a single arm grasping a can and placing it beside a pot.

E.2 Real-World Rollouts

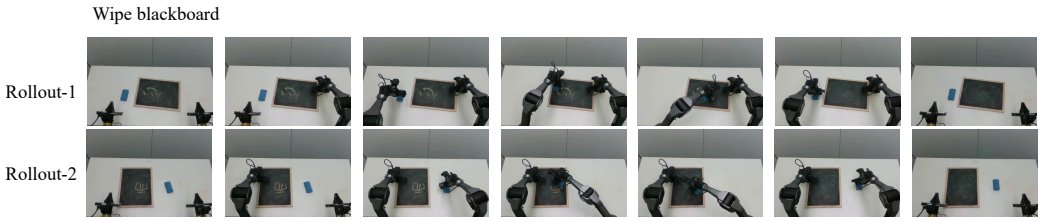


Figure 9: **Real-world Wipe Blackboard.** Two rollouts show one arm stabilizing the board while the other wipes its surface.

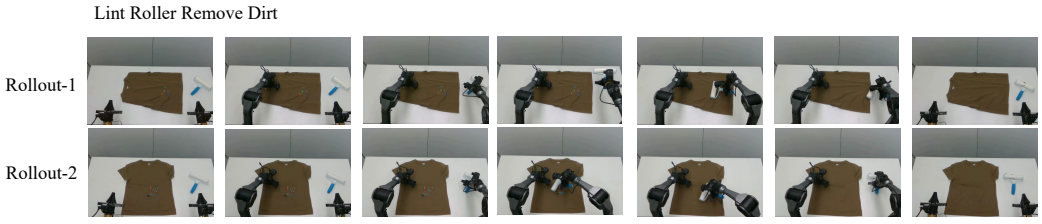


Figure 10: **Real-world Lint Roller Remove Dirt.** Two rollouts show one arm holding the garment in place while the other uses a lint roller to remove debris.