SCOUT: CRITIC-GUIDED SCHRÖDINGER TRANSPORT FOR ONLINE VLA REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Contact-rich and dynamic manipulation often requires well-crafted refinement methods that can learn from sparse rewards while preserving efficient execution beyond a large pretrained policy. We introduce SCOUT, a lightweight framework for refining pre-trained vision-language-action (VLA) policies online through critic-guided Schrödinger transport. SCOUT formulates policy improvement as a Schrödinger bridge from the VLA's reference action chunk to a distribution of promising corrections, where a critic proposes and reweights multiple candidate action chunks, thereby constructing a training target that captures coordinated changes across action space and time. We exploit the point-source structure induced by conditioning on the VLA reference action, enabling tractable bridge regression toward an online critic-weighted target. This enables simulation-free learning through endpoint regression on Brownian interpolants without iterative coupling training. We evaluate SCOUT on simulated benchmarks and real-world precise manipulation tasks. SCOUT improves success rates over RLT and DSRL baseline methods. It achieves comparable measured inference latency to RLT and improves real-world active-execution throughput over RLT and both DSRL variants. Together, these results demonstrate that reference-conditioned action transport strengthens effective online refinement with low deployment overhead. More details and visualizations can be found on our website: https://scout-vla.github.io.

1 INTRODUCTION

A pretrained vision–language–action (VLA) policy can provide the overall structure of a manipulation skill while leaving subtle yet consequential details unresolved. In contact-sensitive tasks, success often hinges on coordinated refinements to the approach pose, contact location, and motion during interaction (Black et al., 2024; Cheng et al., 2023). Although these adjustments may appear minor, they can determine whether an insertion succeeds or a placement remains stable. Online experience offers a direct means of discovering such corrections from task outcomes. The practical goal is therefore to learn these refinements with minimal interaction and execute them without substantially increasing decision latency.

Our central finding is that *critic-guided refinement of the VLA's existing action chunk enables effective online adaptation at low deployment cost*. Across the evaluated LIBERO tasks, SCOUT matches or improves the average success rate of latent-space refinement while requiring approximately half the measured decision latency. Its gains over the action-space RLT baseline are most pronounced on tasks where the frozen policy achieves only moderate or low success rates. Physical-robot experiments further reveal two complementary benefits of refinement: faster execution when the base skill is already effective, and more reliable correction when success requires precise interaction. These results motivate us to refine an actor that treats the pretrained action as a structured initialization and learns when and how to depart from it.

Existing lightweight policy adaptation methods retain a pretrained controller and optimize a smaller policy around it (Johannink et al., 2019; Xu et al., 2026; Wagenmaker et al., 2025). RLT exposes a compact representation for online actor–critic learning, while DSRL learns to steer the latent noise of a frozen generative policy. Built on the prepared RLT interface, SCOUT constructs an explicit action-space training distribution. A critic scores alternatives anchored at the reference chunk, including coordinated candidate action chunks along its gradient direction. Exponential reweighting turns

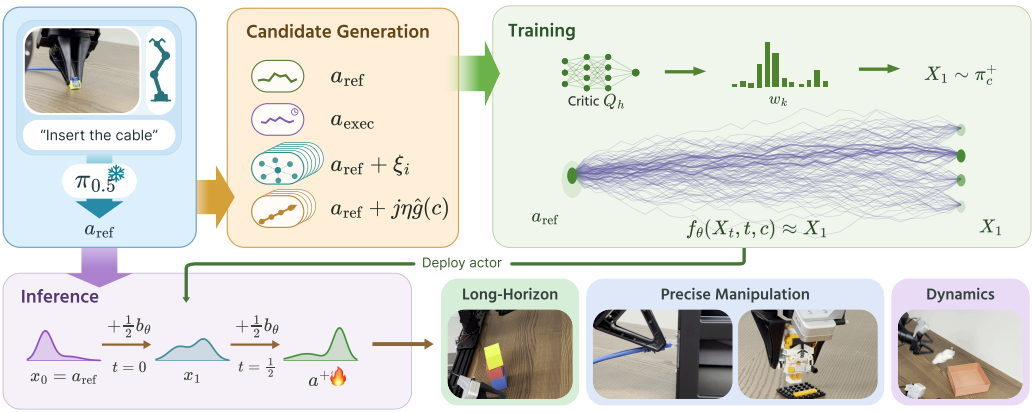


Figure 1: **SCOUT overview:** 1) the frozen VLA provides a reference chunk a_{ref} and 12 candidate actions are generated from the reference; 2) critic scores q_k and normalized weights w_k define an empirical target $\tilde{\pi}_c$; and 3) the Schrödinger bridge learns to refine the point source $\delta_{a_{\text{ref}}}$ toward this target (these quantities are defined in Section 3.2). Two drift-only steps at inference time enable efficient execution in real-world manipulation. The stochastic paths illustrate the bridge construction.

these alternatives into a distribution of promising refinements. Because each candidate is a complete sequence, the target can express changes across joints and control steps, including alternative contact strategies, without requiring a strategy label.

More specifically, SCOUT learns to refine VLA-generated actions online through a Schrödinger bridge. By conditioning on the action chunk produced by the VLA, the source distribution collapses to a point mass, inducing a unique coupling with the target distribution. This structure enables direct training on Brownian-bridge interpolants: a lightweight network predicts the target endpoint from an intermediate action chunk, the transport time, and the context. The critic guides the search during training, while the deployed actor performs only two inexpensive drift updates after the VLA generates an action chunk. Consequently, inference requires neither candidate generation nor critic evaluation. An overview of the SCOUT method is presented in Figure 1.

The main contributions of this paper can be summarized as follows:

- We introduce an online VLA-refinement framework that separates critic-guided action search during learning from efficient actor-based control at deployment.
- We develop a Schrödinger bridge formulation with a tractable endpoint-regression objective, enabling the refined actor to learn efficient improvements to VLA-generated action chunks.
- We carry out comprehensive evaluations spanning task success, inference cost, and physical execution, demonstrating that the proposed refinement improves performance on challenging precision tasks and preserves execution speed when latency-sensitive corrections are critical.

2 Related Work

Vision-Language-Action Models and Their Adaptation. Vision-language-action (VLA) models pair a VLM backbone with an action head trained on large multi-embodiment demonstration datasets (Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025b). They generate action chunks to mitigate compounding error (Zhao et al., 2023) and use generative heads that represent multi-modal action distributions (Chi et al., 2025; Team et al., 2024; Black et al., 2024). It has been shown that VLA models exhibit strong generalization capabilities across different tasks, embodiments, and environments. Since their performance on a particular task is bounded by the demonstrations, task-oriented supervised fine-tuning (SFT) has been proposed as a straightforward improvement (Kim et al., 2025; Ferchau et al., 2026). However, SFT on VLA models is subject to performance collapse under modest distribution shifts (Zhou et al., 2025; Guo et al., 2026), and contact-rich manipulation

exposes precision failure modes that more demonstrations cannot resolve (Parés-Morlans et al., 2026). These limitations motivate online policy refinement rather than learning from demonstrations alone.

Schrödinger Bridges and Transport Policies. SCOUT's refiner is a Schrödinger bridge: a stochastic process that transports one endpoint distribution to another while staying close, in relative entropy, to a reference Brownian motion (Léonard, 2014; Chen et al., 2021). Fitting a general bridge requires estimating the source-target coupling through iterative fitting (De Bortoli et al., 2021; Shi et al., 2023), entropic optimal transport (Tong et al., 2024), or regression on paired endpoints (Liu et al., 2023b; Zhou et al., 2024). Fortunately, the coupling is unique when the source is a point mass, making the optimal drift a Doob h-transform of the reference process (Chen et al., 2021; Tzen & Raginsky, 2019). Hence, it can be estimated simulation-free as the conditional expectation of the endpoint, with the Föllmer process as the variationally optimal choice (Chen & Vanden-Eijnden, 2026). SCOUT uses exactly this construction with no coupling estimation, source network, or simulation of the learned process required.

Online Refinement of Pretrained Policies. Reinforcement learning (RL) to update VLA weights is a powerful path beyond the SFT ceiling. Recent work applies policy gradients (Lu et al., 2025; Tan et al., 2025; Li et al., 2026), advantage conditioning (Intelligence et al., 2025a), or preference objectives (Zhang et al., 2024) to the full model. When the action head is a generative sampler, the denoising or flow integration process can be independently optimized with policy gradients (Ren et al., 2025; Zhang et al., 2025; Chen et al., 2025). Both routes can be costly: on-policy updates need large-scale parallel simulations (Li et al., 2026; Chen et al., 2025) or hours of real-robot interaction per task (Intelligence et al., 2025a), and updating pretrained weights risks forgetting the generalist priors that motivate VLAs (Guo et al., 2026).

A growing family of methods avoids the cost and risks by leaving the pretrained policy frozen and learning a correction in different stages of its action pipeline. Residual RL superposes a learned perturbation on the base policy's output in the action space (Johannink et al., 2019; Ankile et al., 2025b; Yuan et al., 2025; Ankile et al., 2025a). However, value learning near a fixed base action suffers from cold-start and scale-mismatch instabilities (Ma et al., 2026). RLT mitigates this by exposing an RL token from the VLA and conditioning an actor-critic on both the token and the reference action chunk (Xu et al., 2026). Other methods optimize a latent that the frozen action sampler consumes: the seeding noise in DSRL (Wagenmaker et al., 2025), or a bottleneck observation latent in ZPRL (Yu et al., 2026), which preserves the base policy's prior action manifold. Others leverage the strong VLA prior, sample several action candidates, and select the best by critic score (Nakamoto et al., 2024; Xie et al., 2026; Kwok et al., 2026). However, decision cost grows with the number of candidates. The evaluation can instead be amortized: a separate actor is trained to reproduce the critic-scored target, so no candidate is evaluated at deployment (Simmons-Edler et al., 2019; Shao et al., 2022; Mark et al., 2024). The target itself is typically built by exponential reweighting of candidates by critic score (Peters et al., 2010; Abdolmaleki et al., 2018; Peng et al., 2019; Gao et al., 2026), by critic-gradient ascent on the actions (Yang et al., 2023; Mark et al., 2024), or by reward-weighting the flow-matching loss (Pfrommer et al., 2025; Fan et al., 2025; Sochopoulos et al., 2026).

3 SCOUT: Critic-Guided Action Refinement

3.1 Basic Setting

SCOUT operates as a lightweight, online RL module designed to improve the actions of a pre-trained VLA model. At each replanning decision, the frozen VLA processes raw camera images and language instructions to predict an initial sequence of H low-level robot commands. These commands form a high-dimensional reference action chunk, $a_{\text{ref}} \in \mathbb{R}^D$, where $D = Hd_a$ and d_a is the single-step action dimension. Rather than reprocessing the visual inputs from scratch, SCOUT extracts the VLA's internal representation, specifically the RL token z_{RL} (Xu et al., 2026), and combines it with the robot's proprioceptive state p and reference chunk a_{ref}. This creates a single context vector:

$$c = (z_{\text{RL}}, p, a_{\text{ref}}) \tag{1}$$

The refiner uses this context c to output an improved action chunk $a^+ \in \mathbb{R}^D$. During online interaction, the robot executes an action a_{exec}, stores the outcome in a replay buffer, and uses this data to train twin critics that estimate the return of the action chunks.

The index n labels replay transitions, and ℓ labels control ticks within each replay window. Simulation windows coincide with decisions; hardware windows span 30 ticks while replanning occurs every 25 ticks. In contrast, $t \in [0, 1]$ is continuous internal *transport time*, and $X_t \in \mathbb{R}^D$ is the random action chunk at transport time t. Appendix A summarizes our mathematical notation, and Appendix E.2 provides more SCOUT architectural details.

3.2 A CRITIC-WEIGHTED IMPROVEMENT TARGET

To guide the policy refiner toward better actions without discarding the VLA's initial output, we build a structured, critic-weighted target distribution. We train twin critics $Q_h(c, a)$ on sparse terminal success signals and use the conservative minimum score: $q(c, a) = \min_h Q_h(c, a)$.

We construct a discrete pool of $K = 12$ candidate action chunks, $\mathcal{C} = (a_1, \ldots, a_K)$, combining reference-policy behavior, replay experience, and local exploration. The pool contains four types of candidates: (i) the reference action a_{ref} predicted by the frozen VLA; (ii) one previously executed action a_{exec} retrieved from replay; (iii) six Gaussian perturbations of a_{ref}, with noise $\xi_i \sim \mathcal{N}(0, \rho^2 I_D)$; and (iv) four critic-guided candidates generated through gradient updates $\eta\widehat{g}(c)$ toward higher predicted values. This construction provides a finite set of alternatives for refinement while retaining the original policy prediction. Formally, the candidate pool is:

$$\mathcal{C} = \Big(a_{\text{ref}},\, a_{\text{exec}},\, \{a_{\text{ref}} + \xi_i\}_{i=1}^{6},\, \{a_{\text{ref}} + j\eta\widehat{g}(c)\}_{j=1}^{4}\Big). \tag{2}$$

Moreover, the gradient direction is normalized over the entire chunk to ensure stable steps:

$$\widehat{g}(c) = \frac{\nabla_a q(c, a)|_{a=a_{\text{ref}}}}{\|\nabla_a q(c, a)|_{a=a_{\text{ref}}}\|_2 + 10^{-8}} \tag{3}$$

We assign each candidate a probability weight, which defines the distribution of improved action chunks that the actor is trained to produce. Following relative-entropy policy improvement, the weights maximize the expected critic score while a Kullback–Leibler (D_{KL}) penalty keeps them close to uniform, discouraging excessive concentration on individual candidates. The solution is an exponentially tilted distribution $\tilde{\pi}_{c,\mathcal{C}} = \sum_{k=1}^{K} w_k \delta_{a_k}$, in which a candidate's weight grows exponentially with its score:

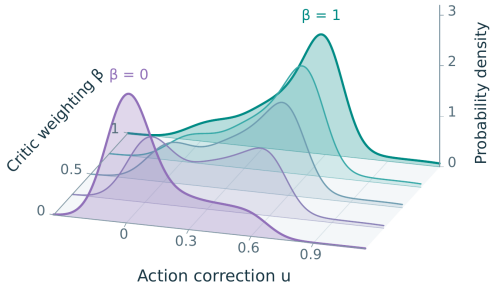


Figure 2: **Effect of critic weighting.** Marginal target densities over action correction u at the final checkpoint, aggregated over five LIBERO-PRO runs and 300 contexts. $\beta = 0$ gives equal candidate weights; $\beta = 1$ recovers SCOUT's target. Construction details are in Appendix D.4.

$$w_k = \frac{\exp\{\alpha(q_k - \bar{q})\}}{\sum_{j=1}^{K} \exp\{\alpha(q_j - \bar{q})\}}. \tag{4}$$

Here, q_k is the critic score for candidate a_k, $\bar{q}$ is the mean score, and $\alpha > 0$ controls the penalty. We monitor weight concentration using the normalized effective sample size $\text{ESS}(w) = 1/(K \sum_{k=1}^{K} w_k^2)$, with $u_k = 1/K$ denoting the uniform index distribution. Our controller dynamically adjusts α based on the batch-average ESS (as detailed in Appendix D) to maintain a healthy distribution of weights across the candidates:

$$D_{\text{KL}}(w \| u) \leq -\log \text{ESS}(w) \tag{5}$$

Figure 2 visualizes the critic-weighted training target, not the learned actor's output distribution; Appendix D.4 gives the construction details.

3.3 LEARNING THE POINT-SOURCE SCHRÖDINGER BRIDGE

We use a conditional Schrödinger bridge to transport the VLA's reference action toward the critic-weighted target distribution. Since the context c includes a_{ref}, the source distribution is the point mass $\delta_{a_{\text{ref}}}$. This point source fixes the source–target coupling and eliminates the need to estimate it.

The endpoint predictor $f_\theta(x, t, c)$ learns to recover the target action X_1 from a noisy intermediate state X_t. For each context and sampled candidate pool, we draw

$$k \sim \mathrm{Cat}(w), \qquad X_1 = a_k, \qquad Z \sim \mathcal{N}(0, I_D).$$

Averaging over candidate-pool draws gives the conditional endpoint law π_c^+, as detailed in Appendix B. At a sampled transport time t, we construct

$$X_t = (1-t)a_{\mathrm{ref}} + tX_1 + \sigma\sqrt{t(1-t)}\, Z, \tag{6}$$

where $\sigma > 0$ controls the bridge noise. We train the predictor by minimizing the endpoint regression loss

$$\mathcal{L}_{\mathrm{SB}}(\theta) = \mathbb{E}\Big[\big\| f_\theta(X_t, t, c) - X_1 \big\|_2^2\Big]. \tag{7}$$

The expectation includes replay contexts, candidate generation, endpoint sampling, transport times, and Gaussian noise.

3.4 Online Learning and Fast Execution

The endpoint predictor f_θ is a multilayer perceptron (MLP) with two hidden layers of 256 units.

During learner updates. The learner updates the critics using temporal-difference learning and trains the endpoint predictor using Eq. (7). Candidate generation and critic scoring occur only during learning. Before learning begins, the reference policy collects experience to populate replay.

The critics use target networks $f_{\bar\theta}$ and $\bar Q_h$. The bootstrap action, TD target, and critic loss are

$$a_n^{\mathrm{boot}} = f_{\bar\theta}\big(a_{\mathrm{ref},n}^{\mathrm{boot}}, 0, c_n^{\mathrm{boot}}\big) + \varepsilon, \tag{8}$$

$$y_n = \sum_{\ell=0}^{H-1} \gamma^\ell r_{n,\ell} + (1-\zeta_n)\gamma^H \min_h \bar Q_h\big(c_n^{\mathrm{boot}}, a_n^{\mathrm{boot}}\big), \tag{9}$$

$$\mathcal{L}_Q = \sum_{h=1}^{2} \mathbb{E}_{\mathcal{D}}\Big[\big(Q_h(c_n, a_{\mathrm{exec},n}) - y_n\big)^2\Big]. \tag{10}$$

Here, $\mathcal{D}$ denotes replay, ζ_n indicates termination, and c_n^{boot} is the context at the replay-window endpoint. Target-smoothing noise follows $\varepsilon \sim \mathcal{N}(0, 0.05^2 I_D)$ in simulation and is omitted in real-world experiments. Each learner step performs one critic update. The actor, Polyak target networks, and ESS controller for α update every second learner step. Replay-window construction and terminal handling are detailed in Appendix E.3; Algorithm 1 summarizes the learner and execution procedures.

Figure 3 shows how the critic-weighted target evolves across four training checkpoints on the same recorded contexts. We denote training progress by e/E, where e counts completed episodes and $E = 300$ is the training budget. The projection directions remain fixed across checkpoints, while directed proposals, critic scores, and controller temperatures follow the corresponding checkpoint. This comparison therefore tracks the evolving target construction at matched contexts.

During online data collection. Exploration uses the one-step endpoint prediction with additive dither of standard deviation 0.05 in simulation, and eight Euler–Maruyama steps in real-world experiments.

During execution (inference). The policy generates a refined action using two deterministic Euler steps, bypassing the critics and candidate generation. The learned drift is

$$b_\theta(x, t, c) = \frac{f_\theta(x, t, c) - x}{1 - t}, \qquad t < 1. \tag{11}$$

Starting from $x_0 = a_{\mathrm{ref}}$ with step size $1/2$, the two Euler updates simplify to

$$\begin{aligned} x_{1/2} &= \frac{a_{\mathrm{ref}} + f_\theta(a_{\mathrm{ref}}, 0, c)}{2}, \\ a^+ &= f_\theta\big(x_{1/2}, \tfrac{1}{2}, c\big). \end{aligned} \tag{12}$$

This readout evaluates the predictor at $t = 0$ and $t = 1/2$, avoiding evaluation of the drift at its terminal singularity $t = 1$. It requires one frozen-VLA forward pass and two evaluations of the small

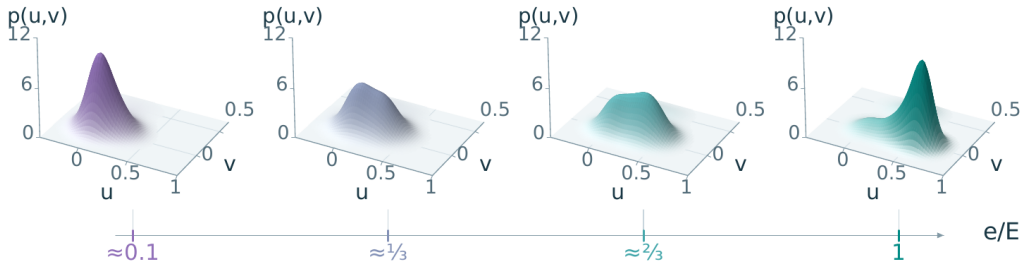


Figure 3: **Evolution of the critic-weighted target during training.** Joint target densities $p(u, v)$ at $\beta = 1$, using the same five runs and 300 contexts as Figure 2. Snapshots are evenly spaced on an approximate training-progress axis e/E; spacing is schematic, and the first snapshot follows warmup. The coordinate v projects action corrections onto the orthogonalized recorded-action direction. All snapshots share fixed projection directions, axes, camera, and smoothing bandwidth. These are target densities, not measured actor distributions. Exact construction details are in Appendix D.4.

actor network. We select two steps empirically from an ablation study on five LIBERO-PRO layouts (Appendix H.2, Table 14).

For a fixed target distribution and the population-optimal endpoint predictor, this readout admits a Gaussian mean-shift interpretation. Let $m_c = \mathbb{E}[X_1 \mid c]$ denote the target mean, and let $p_{\sigma,c}$ be the density of π_c^+ convolved with $\mathcal{N}(0, \sigma^2 I_D)$. Then

$$a^\star_{\text{2step}} = m_c + \sigma^2 \nabla \log p_{\sigma,c}(m_c). \tag{13}$$

Thus, the ideal two-step readout performs one Gaussian mean-shift update initialized at the target mean, reweighting endpoints according to their distance from that mean. The output remains a conditional mean and need not select a mode. Appendix C provides the derivation and its scope.

4 Experiments

We evaluate SCOUT on two simulated benchmarks, LIBERO (Liu et al., 2023a) and LIBERO-PRO (Zhou et al., 2025), and four real-world manipulation tasks. We examine whether online refinement improves task success and execution throughput across diverse manipulation settings. The evaluations shed light on these key questions:

1. Does SCOUT improve performance (a) in distribution and (b) under distribution shift?
2. Does SCOUT improve real-world task throughput under deployment constraints?
3. How does each component of SCOUT affect its performance?

LIBERO is used to study refinement on standard benchmark tasks and LIBERO-PRO to assess robustness under distribution shifts. Real-world experiments span long-horizon, precise assembly, and highly dynamic manipulation. Task performance is measured using binary task success rate and throughput (the number of successes within a fixed time window of 10 minutes; Appendix F.2). The latter is a key metric to assess the feasibility of real-world deployments.

We compare SCOUT against three online refinement baselines:

- **RLT** (Xu et al., 2026) trains a small TD3+BC actor-critic on the frozen VLA's prepared RL token, proprioception, and reference action chunk to output a refined action chunk. SCOUT uses the same RL-token preparation pipeline as the RLT baseline. As the official RLT code is not publicly available, we use an open-source reproduction (Yang et al., 2026).
- **DSRL** (Wagenmaker et al., 2025) leaves the VLA's flow-matching action head frozen and uses SAC to learn a vision-based policy over its input noise, so that decoding the chosen noise steers the VLA toward higher-reward action chunks without changing its weights.

- **DSRL**$_{RLT}$ is a variant of DSRL we constructed that replaces the CNN image encoder with RLT's RL token and proprioception, so its noise-space policy sees the same inputs as the action-space methods. Importantly, this method requires two VLA passes per final output.

In summary, our experiments show that SCOUT (a) achieves the largest gain after online adaptation to shifted layouts, (b) provides the most consistent improvements across all tasks, and (c) delivers higher real-world throughput than all baselines. Implementation details and hyperparameters of all baselines are given in Appendix E.6.

4.1 SIMULATION EXPERIMENTS

We evaluate SCOUT on 20 LIBERO and 10 LIBERO-PRO tasks that have a wide range of base policy performances. For each LIBERO task, 50 initial states are shared between training and evaluation, and each state is visited twice over the 100 total evaluations. LIBERO-PRO evaluates layout generalization by exchanging object-placement regions while retaining object identities; for each layout, training and evaluation use disjoint sets of initial states. Each method is trained online for 300 episodes and evaluated for 100. All methods refine the same frozen $\pi_{0.5}$ LIBERO checkpoint (Appendix F.1). Appendices F.3 and F.4 give further simulation details, and Tables 7 and 8 report per-task results.

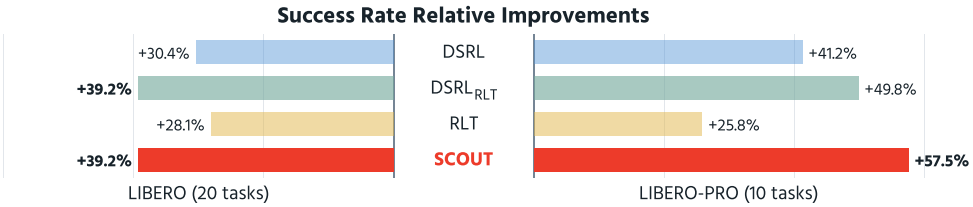


Figure 4: **Average success-rate gain over the base policy.** Mean absolute improvement in success rate (percentage points), averaged over the 20 LIBERO tasks (left) and the 10 LIBERO-PRO tasks (right). SCOUT matches the best baseline on LIBERO (+39.2) and yields the largest gain on LIBERO-PRO (+57.5), 7.7 percentage points above the next best.

SCOUT matches the best baseline for in-distribution setups. As shown on the left of Figure 4, SCOUT improves on the base policy by an average of 39.2 percentage points (from 53.1% to 92.3%), tying DSRL$_{RLT}$ with half its inference cost and leading RLT by over 10 percentage points.

SCOUT improves adaptation to shifted layouts. Between larger domain randomization in LIBERO-PRO and a held-out set of evaluation environments, a refiner's improvements must stem from true generalization over the domain rather than mere trajectory memorization. Across 10 LIBERO-PRO tasks, SCOUT's performance advantage widens with higher task difficulty. It outperforms the next best baseline by 8 percentage points and beats RLT by over 30 percentage points. With identical RL-token construction, the difference lies in the actor and critic-weighted target. This suggests that our point-source Schrödinger Bridge construction effectively manages out-of-distribution physical configurations without collapsing.

Additionally, SCOUT is the only method that improves its success rate over the base policy in all 30 simulation tasks, while all baselines regress in performance in at least two tasks. We provide detailed results and analysis of SCOUT's consistency in LIBERO and LIBERO-PRO in Appendix G.

4.2 REAL-WORLD DEPLOYMENT

We evaluate SCOUT's real-world manipulation performance using a bimanual I2RT YAM embodiment on four tasks spanning long-horizon (Cube-Stacking), precise-manipulation (Ethernet-Insertion, LEGO-Click) and dynamic (Doll-Throwing) tasks. Figure 5 illustrates each task. Every method is trained for 300 episodes followed by 50 evaluations. An author acts as the success judge for all methods and tasks using predetermined rubrics. Appendix F.5 describes the platform, and Appendix F.6 gives the evaluation configurations and success criteria of each task.

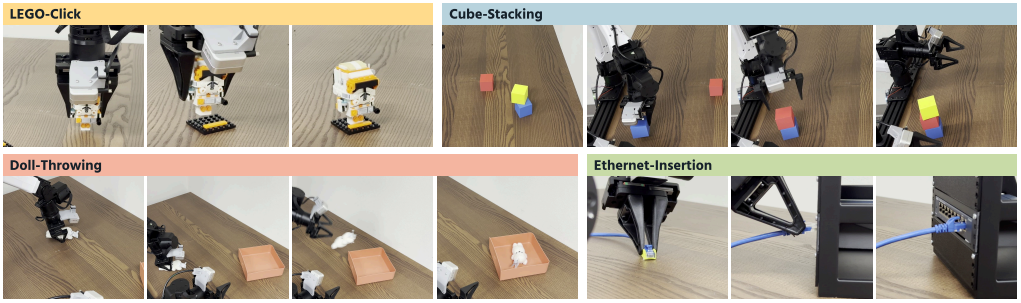


Figure 5: **Real-world tasks on the bimanual YAM platform.** Frames progress left to right. **LEGO-Click**: press a minifigure onto its baseplate until it clicks. **Cube-Stacking**: stack the blue, red, and yellow cubes into a tower. **Ethernet-Insertion**: insert an Ethernet cable into the rightmost port of the network switch. **Doll-Throwing**: throw the doll into the red bin across the table.

SCOUT adaptively improves execution throughput. SCOUT exceeds RLT and both DSRL variants in active-execution throughput on all four tasks (Figure 6), but the source of the gain depends on the task. In Cube-Stacking, where the base policy already succeeds reliably, we observe that the gain comes mainly from faster execution: in a separate 20-trial timing study, SCOUT completes an episode in about half the time of the base policy (13.3 s vs. 26.1 s). In Doll-Throwing, however, throughput increases primarily with better reliability (from 68% to 84%). A faster throw alters the release trajectory and misses the bin, so the critic-weighted target concentrates on a narrow window of precise motion candidates that land the doll most consistently. In contrast, DSRL$_{RLT}$ fails all 50 throwing trials: its two VLA passes per control update stall execution between action chunks, preventing the robot from reaching the velocity needed to throw. SCOUT adds only two small MLP evaluations to a single VLA pass, and thus little latency to the base policy (Appendix G). Table 11 outlines results for the Cube-Stacking timed throughput study, and Table 12 compares the inference costs between SCOUT and the baselines.

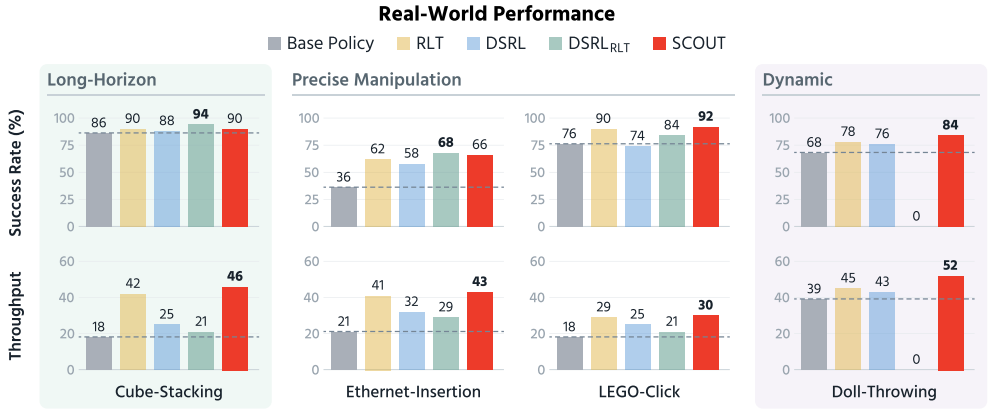


Figure 6: **Real-world success rate and throughput.** Top: success rate (%) over 50 evaluation trials per task. Bottom: throughput, the number of successful completions within 10 minutes of accumulated execution time. The dashed line marks the frozen base policy. SCOUT improves on the base policy in every task, achieves the highest throughput in all four tasks, and produces the best success rate on LEGO-Click and Doll-Throwing.

4.3 ABLATION

We assess the contribution of the point-source Schrödinger bridge actor by replacing it with a Gaussian-source conditional flow-matching actor (SCOUT-CFM) of matched capacity and conditioning inputs. Both actors share the candidate-generation, critic-weighting, and ESS-control procedures, as well as the critic architecture and replay/update protocol.

Actor formulation. Let c denote the context, including the VLA reference chunk, and let X_1 be a sample from the critic-weighted conditional endpoint law $\pi^{+}_{c,\mathrm{CFM}}$. SCOUT-CFM draws $U \sim \mathcal{N}(0, I)$ independently of X_1 conditional on c, samples $t \sim \mathcal{U}[0, 1]$, sets $Y_t = (1 - t)U + tX_1$, and learns a velocity field using

$$\mathcal{L}_{\mathrm{CFM}}(\phi) = \mathbb{E}\Big[\|v_\phi(Y_t, t, c) - (X_1 - U)\|_2^2\Big]. \tag{14}$$

The VLA reference therefore enters SCOUT-CFM through conditioning only, whereas it also defines the transport source in SCOUT. Deterministic evaluation uses two Euler steps from the canonical source $U = 0$, matching SCOUT's two actor-network evaluations per decision.

Table 1: **Schrödinger Bridge vs. Gaussian-source CFM.** Schrödinger Bridge beats the CFM ablation on every simulation and real-world success rate measure, most noticeably in LIBERO-PRO by 36.5 percentage points. It only loses to CFM in the LEGO task by 1 completion in 10-min throughput and leads in all other tasks.

	Successes		Throughput	
Task / benchmark	SCOUT	CFM	SCOUT	CFM
LIBERO	**92.3/100**	87.3/100	–	–
LIBERO-PRO	**90.8/100**	54.3/100	–	–
Doll-Throwing	**42/50**	40/50	**52**	49
LEGO-Click	**46/50**	42/50	30	**31**
Ethernet-Insertion	**33/50**	21/50	**43**	24
Cube-Stacking	**45/50**	44/50	**46**	45

As shown in Table 1, the Schrödinger Bridge achieves higher success than CFM on both benchmarks and all four hardware tasks. The gap is largest on LIBERO-PRO and on the Ethernet-Insertion task, and it is relatively small on the other real-world tasks. In real-world throughput, the Schrödinger Bridge outperforms CFM by 80% on Ethernet-Insertion and achieves similar observed throughput on the other tasks.

We observe that the base Ethernet-Insertion policy fails equally during both grasp and port-alignment, whereas the base LEGO-Click policy fails primarily during alignment. This suggests a distinction between correcting local alignments and coordinating corrections across multiple stages. The larger bridge advantage on Ethernet-Insertion suggests that anchoring transport at the VLA reference may be particularly helpful when grasp corrections must remain compatible with downstream alignment. This interpretation motivates the bridge as a way to preserve useful action structure while learning coordinated refinements.

Additional ablations. Across five LIBERO-PRO layouts and three training seeds per layout, SCOUT averages 82.5 successes per 100 trials, compared with 74.1 for hard-target bridge training, 67.7 for direct weighted regression, and 65.7 for hard-choice distillation (Appendix H.1). Appendix H.2 presents a separate frozen-checkpoint study of deterministic and stochastic deployment readouts at different step counts.

Repeatability across training runs. Across five separately trained SCOUT runs on each of five LIBERO-PRO layouts, equal-layout mean success rates range from 82.2% to 84.4%, with a mean of 83.2% and a sample standard deviation of 1.0 percentage point. Appendix H.3 reports every run and the per-layout variation, combining the main comparison with the newly trained ablation controls.

5 CONCLUSION

We introduce SCOUT, an online action-refinement framework that improves the outputs of VLA models using a highly efficient point-source Schrödinger Bridge. By leveraging twin critics to construct a localized target distribution and projecting this distribution through a lightweight generative actor, SCOUT achieves high final-checkpoint success under the evaluated online-training budgets while adding only two lightweight actor evaluations to the frozen VLA. It consistently elevates the performance of foundational VLAs on challenging simulated tasks and demonstrates exceptional reliability and throughput in real-world physical manipulation. Future work will explore expanding SCOUT's refinement horizon and integrating multi-modal sensory feedback directly into the bridge transport mechanics.

Reproducibility Statement

The appendix specifies the notation, bridge derivation, learning schedules, simulation and real-world hyperparameters, the robot action interface, task definitions, and evaluation procedures. It reports individual simulation outcomes, real-world success counts, and the aggregation rules and totals for physical execution efficiency.

AI Use Statement

Generative AI tools assisted with literature organization, manuscript editing, programming and debugging, discussion of mathematical derivations, and recomputation of summary statistics. The authors collected the experimental observations, annotated task successes, and measured task throughput and other performance metrics. The authors are responsible for the experiments, mathematical claims, references, and final manuscript.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual off-policy RL for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025a.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual RL for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025b.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, et al. π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

Yifan Chen and Eric Vanden-Eijnden. Variational optimality of Föllmer processes in generative diffusions. *arXiv preprint arXiv:2602.10989*, 2026.

Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. Stochastic control liaisons: Richard Sinkhorn meets Gaspard Monge on a Schroedinger bridge. *SIAM Review*, 63(2):249–313, 2021.

Xianyi Cheng, Sarvesh Patil, Zeynep Temel, Oliver Kroemer, and Matthew T Mason. Enhancing dexterity in robotic manipulation via hierarchical contact exploration. *IEEE Robotics and Automation Letters*, 9(1):390–397, 2023.

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

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion Schrödinger bridge with applications to score-based generative modeling. *Advances in neural information processing systems*, 34:17695–17709, 2021.

Jiajun Fan, Shuaike Shen, Chaoran Cheng, Yuxin Chen, Chumeng Liang, and Ge Liu. Online reward-weighted fine-tuning of flow matching with Wasserstein regularization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. MolmoAct2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

Finn Ferchau, Daniel Pommer, and Cristian Axenie. On the efficiency of LoRA fine-tuning for vision-language-action models in industrial robotic manipulation. In *Artificial Neural Networks and Machine Learning–ICANN 2026: 35th International Conference on Artificial Neural Networks, Padua, Italy, September 14–17, 2026, Proceedings, Part V*, pp. 191. Springer Nature, 2026.

Ting Gao, Stavros Orfanoudakis, Nan Lin, Winnie Daamen, Serge Hoogendoorn, and Elvin Isufi. Flow matching policy optimization with mirror descent and entropy constraints. *arXiv preprint arXiv:2603.17685*, 2026.

Xinyu Guo, Bin Xie, Wei Chai, Xianchi Deng, Tiancai Wang, Zhengxing Wu, and Xingyu Chen. PriorVLA: Prior-preserving adaptation for vision-language-action models. *arXiv preprint arXiv:2605.10925*, 2026.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning*, pp. 17–40. PMLR, 2025b.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning*, pp. 2679–2713. PMLR, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025.

Jacky Kwok, Xilun Zhang, Mengdi Xu, Yuejiang Liu, Azalia Mirhoseini, Chelsea Finn, and Marco Pavone. Scaling verification can be more effective than scaling policy learning for vision-language-action alignment. In *European Conference on Computer Vision*, pp. 37–55. Springer, 2026.

Christian Léonard. A survey of the Schrödinger problem and some of its connections with optimal transport. *Discrete and Continuous Dynamical Systems - Series A*, 34(4):1533–1574, 2014.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 125846–125869, 2026.

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

Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos A Theodorou, Weili Nie, and Anima Anandkumar. I^2SB: Image-to-image Schrödinger bridge. In *International Conference on Machine Learning*. PMLR, 2023b.

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. *arXiv preprint arXiv:2505.18719*, 2025.

Guozheng Ma, Lu Li, Haoyu Wang, Zixuan Liu, Pierre-Luc Bacon, and Dacheng Tao. What makes value learning efficient in residual reinforcement learning? *arXiv preprint arXiv:2602.10539*, 2026.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic RL: Offline RL and online RL fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning*, 2024.

Carlota Parés-Morlans, Nils Kuhn, Isabel Liu, Alberta Longhini, and Jeannette Bohg. Demystifying when and why VLAs fail in contact-rich tasks and how to fix them. *arXiv preprint arXiv:2608.01402*, 2026.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Samuel Pfrommer, Yixiao Huang, and Somayeh Sojoudi. Reinforcement learning for flow-matching policies. *arXiv preprint arXiv:2507.15073*, 2025.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Lin Shao, Yifan You, Mengyuan Yan, Shenli Yuan, Qingyun Sun, and Jeannette Bohg. GRAC: Self-guided and self-regularized actor-critic. In *Conference on Robot Learning*, pp. 267–276. PMLR, 2022.

Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion Schrödinger bridge matching. *Advances in neural information processing systems*, 36:62183–62223, 2023.

Riley Simmons-Edler, Ben Eisner, Eric Mitchell, Sebastian Seung, and Daniel Lee. Q-learning for continuous actions with cross-entropy guided policies. *arXiv preprint arXiv:1903.10605*, 2019.

Andreas Sochopoulos, Esmeralda S Whitammer, Nikolaos Tsagkas, João Moura, Michael Gienger, and Sethu Vijayakumar. Optimal transport Q-learning for flow policy steering and acceleration. *arXiv preprint arXiv:2607.06262*, 2026.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free Schrödinger bridges via score and flow matching. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024.

Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pp. 3084–3114. PMLR, 2019.

Tim Van Erven and Peter Harremoës. Rényi divergence and Kullback-Leibler divergence. *IEEE transactions on information theory*, 60(7):3797–3820, 2014.

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. In *Conference on Robot Learning*, pp. 258–282. PMLR, 2025.

Xinyi Xie, Zican Hu, Zhanyu Liu, Yicheng Dong, Wenhao Wu, Zhenhong Sun, Haoran Li, Chunlin Chen, Zhi Wang, and Pichao Wang. Look before you leap: Distilling tree search into action evaluation for frozen VLA models. *arXiv preprint arXiv:2607.03751*, 2026.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL token: Bootstrapping online RL with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

Yi Yang, Huaihang Zheng, Kai Ma, Tian Xie, Guozheng Li, Shenglin Xu, Xiangyu Wang, Yiren Ma, and Baoxu Liu. openpi-rlt: Real-robot rlt reproduction on openpi. `https://github.com/Yyshadow/openpi-RLT`, 2026. Open-source real-robot reproduction of RL Token (RLT) built on openpi.

Dongjie Yu, Kun Lei, Zhennan Jiang, Jia Pan, and Huazhe Xu. Beyond action residuals: Real-world robot policy steering via bottleneck latent reinforcement learning. *arXiv preprint arXiv:2605.19919*, 2026.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, volume 2025, pp. 28129–28164, 2025.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 106282–106319, 2025.

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. GRAPE: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

Linqi Zhou, Aaron Lou, Samar Khanna, and Stefano Ermon. Denoising diffusion bridge models. In *International Conference on Learning Representations*, volume 2024, pp. 8160–8171, 2024.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

A NOTATION

The mathematical development distinguishes environment time, transport time, and evaluation indices. A context is denoted by c. On hardware, replay windows and replanning intervals have different lengths. The symbol $\nu_{c,\mathcal{C}}$ denotes the empirical candidate proposal and π_c^+ the actor's conditional endpoint target.

Table 2: **Notation used throughout the method.**

Symbol	Definition
n, ℓ	Replay-transition index and control-tick index within its replay window
H, d_a, D	Chunk/replay horizon, command dimension, flattened dimension $D = Hd_a$
L_n	Executed ticks in a SCOUT simulation transition, $L_n \le H$
$a_{\mathrm{ref}}, a_{\mathrm{exec}}, a^+$	Frozen reference, replayed executed chunk, refined output
c, z_{RL}, p	Context, frozen RL-token representation, proprioception
$Q_h, \bar{Q}_h, q$	Critic $h \in \{1, 2\}$, its target network, minimum critic score
$\mathcal{C}, K, a_k$	Indexed candidate collection, candidate count, candidate k
I, β_k	Sampled candidate index and its posterior probability given the bridge state
u_k, κ	Uniform candidate-index probability $1/K$; batch-mean ESS threshold
$\rho, \eta, \widehat{g}$	Gaussian proposal std. per coordinate, directed step size, normalized critic gradient
$w_k, \alpha, \mathrm{ESS}(w)$	Candidate weight, inverse temperature, normalized effective sample size
e_1, e_2, β	Projection directions and critic-tilt multiplier for target-distribution visualizations
$\nu_{c,\mathcal{C}}$	Uniform empirical proposal on the candidate collection
$\tilde{\pi}_{c,\mathcal{C}}, \pi_c^+$	Weighted empirical target and its candidate-averaged law
t, X_t, x_t	Transport time, random chunk, realized chunk
x_0	SCOUT transport source, $x_0 = a_{\mathrm{ref}}$
σ, Z, I_D	Brownian noise scale, standard Gaussian vector, identity matrix
f_θ, b_θ, J	Endpoint predictor, learned drift, readout step count
$\pi^+_{c,\mathrm{CFM}}$	SCOUT (CFM)'s critic-weighted conditional endpoint law
U, Y_t, v_ϕ	SCOUT (CFM) Gaussian source, interpolated action, velocity predictor
$c_n^{\mathrm{boot}}, a_n^{\mathrm{boot}}$	Context and target action at the end of a SCOUT or SCOUT (CFM) replay window
γ, ζ_n, y_n	Per-control-tick discount, termination indicator, critic target
$m_c, p_{\sigma,c}$	Conditional endpoint mean; density after Gaussian smoothing of width σ
$\tau, \tau_{\mathrm{EMA}}, \epsilon_t$	Theoretical endpoint-smoothing bandwidth; Polyak coefficient; transport-time cutoff
ρ_t	Bridge marginal density
$\delta_a, D_{\mathrm{KL}}$	Point mass, KL divergence

B BRIDGE CONSTRUCTION AND TRAINING OBJECTIVE

B.1 CONDITIONAL TARGET AND SCOPE OF ANALYSIS

Fix a context c, critic parameters, and replay distribution, and let $x_0 = a_{\mathrm{ref}}$. A collection draw followed by an endpoint draw $k \sim \mathrm{Categorical}(w)$ samples the conditional law

$$\pi_c^+(A) = \mathbb{E}_{\mathcal{C}|c}\left[\sum_{k=1}^{K} w_k(c, \mathcal{C})\, \mathbf{1}\{a_k \in A\}\right] \tag{15}$$

for every measurable set A. Because the actor conditions on c rather than the specific sampled collection, its population target is this collection-averaged law. For a fixed collection, the same derivations hold for $\tilde{\pi}_{c,\mathcal{C}}$. Averaging over collections does not necessarily eliminate point masses: both the reference and directed candidates can remain deterministic conditional on c. All results below pertain to a fixed target; they do not establish convergence of online actor–critic updates or guaranteed task-return improvement.

B.2 Relative Entropy and the Atomic-Target Limit

Let $\mathbb{W}_{x_0}$ denote a Brownian motion starting at x_0 with constant noise scale $\sigma > 0$. For a bandwidth $\tau > 0$, define the smoothed target $\pi_{c,\tau}^+ = \pi_c^+ * \mathcal{N}(0, \tau^2 I_D)$, and assume π_c^+ has a finite second moment. Gaussian smoothing guarantees that the relative entropy $\mathrm{KL}(\pi_{c,\tau}^+ \parallel \mathbb{W}_1)$ is finite, where $\mathbb{W}_1 = \mathcal{N}(x_0, \sigma^2 I_D)$. Consider the regularized Schrödinger bridge problem:

$$P_\tau^\star = \arg \min_{P: P_0 = \delta_{x_0}, P_1 = \pi_{c,\tau}^+} \mathrm{KL}(P \parallel \mathbb{W}_{x_0}). \tag{16}$$

The bandwidth τ serves exclusively to define this regularized formulation; it is distinct from both the bridge scale σ and the Polyak target coefficient τ_{EMA}.

For any feasible path measure P with finite relative entropy, the Radon–Nikodym derivative factors into the endpoint density ratio and the conditional path distribution ratio. Taking the logarithm and expectation yields the chain rule:

$$\begin{aligned}\mathrm{KL}(P \parallel \mathbb{W}_{x_0}) = \mathrm{KL}(\pi_{c,\tau}^+ \parallel \mathbb{W}_1) \\ + \int \mathrm{KL}\left(P(\cdot \mid X_1 = y) \,\|\, \mathbb{W}_{x_0}(\cdot \mid X_1 = y)\right) \pi_{c,\tau}^+(\mathrm{d}y).\end{aligned} \tag{17}$$

The chain rule identifies the unique minimizer of the smoothed bridge problem: the first term is fixed by the endpoint constraint, and the second nonnegative term vanishes if and only if

$$P_\tau^\star(\mathrm{d}\omega) = \int \mathbb{W}_{x_0}(\mathrm{d}\omega \mid X_1 = y)\, \pi_{c,\tau}^+(\mathrm{d}y), \tag{18}$$

where ω denotes a continuous path. Equality of the conditional laws for almost every endpoint guarantees uniqueness.

If a target measure assigns positive probability to a finite set of points, it is not absolutely continuous with respect to $\mathbb{W}_1$. Its endpoint KL divergence is infinite, and by the data-processing inequality, every path measure admitting this endpoint constraint incurs infinite relative entropy against $\mathbb{W}_{x_0}$. Consequently, the unsmoothed atomic problem lacks a well-defined finite-entropy minimizer. Its bridge process is instead defined via the limiting measure of Eq. (18) as $\tau \downarrow 0$.

To make this limit explicit, couple $Y_\tau = Y + \tau Z_0$ with $Y \sim \pi_c^+$ and independent $Z_0 \sim \mathcal{N}(0, I_D)$. Let $\mathcal{B}_t = B_t - tB_1$ be an independent standard Brownian bridge. The parameterized trajectories

$$X_t^{(\tau)} = (1-t)x_0 + tY_\tau + \sigma \mathcal{B}_t \tag{19}$$

induce the path measure $P_\tau^\star$ and converge uniformly almost surely to $X_t^{(0)}$ as $\tau \to 0$, bounded by $\sup_{t \in [0,1]} \|X_t^{(\tau)} - X_t^{(0)}\| \le \tau \|Z_0\|$. For a finite atomic target, the limiting path law simplifies to

$$P^\star(\mathrm{d}\omega) = \sum_{k=1}^K w_k\, \mathbb{W}_{x_0}(\mathrm{d}\omega \mid X_1 = a_k). \tag{20}$$

The limiting notation $P^\star$ designates this explicit mixture distribution rather than an optimal solution to an infinite-entropy variational problem. Note that no spatial smoothing is applied to endpoints in the implemented regression loss.

B.3 Brownian Interpolation and Endpoint Posterior

Under the reference process $\mathbb{W}_{x_0}$, the joint distribution of (X_t, X_1) is Gaussian with mean (x_0, x_0) and covariance matrices $\mathrm{Var}(X_t) = \sigma^2 t I_D$, $\mathrm{Var}(X_1) = \sigma^2 I_D$, and $\mathrm{Cov}(X_t, X_1) = \sigma^2 t I_D$. Standard Gaussian conditioning yields:

$$X_t \mid X_1 = y \;\sim\; \mathcal{N}\big((1-t)x_0 + ty,\; \sigma^2 t(1-t) I_D\big)\,. \tag{21}$$

Equation (21) permits exact direct sampling of the target one-time marginal (X_t, X_1) without simulating full path histories. Independent Gaussian draws across different time steps represent valid training pairs, not time-correlated Brownian realizations.

For a fixed finite collection, let $I \sim \mathrm{Categorical}(w)$ be the sampled candidate index and set $X_1 = a_I$. For $0 < t < 1$, let $\varphi_{k,t}(x)$ denote the conditional Gaussian density defined in Eq. (21) for $y = a_k$. Applying Bayes' rule yields the posterior allocation over indices and the conditional mean:

$$\beta_k(x,t) = \Pr(I = k \mid X_t = x, c, \mathcal{C}) = \frac{w_k \varphi_{k,t}(x)}{\sum_j w_j \varphi_{j,t}(x)},$$
$$\mathbb{E}[X_1 \mid X_t = x, c, \mathcal{C}] = \sum_{k=1}^{K} \beta_k(x,t) a_k. \tag{22}$$

This index-based definition also covers duplicate candidates: the posterior probability of an endpoint value a is $\sum_{j:a_j=a} \beta_j(x,t)$. For the actor's collection-averaged target, these discrete sums generalize to integrals against π_c^+. This distinction is crucial when computing diagnostic quantities on isolated candidate collections.

B.4 Doob Transform, Markov Property, and Drift

Let $p_s(x,y)$ denote the standard heat kernel with variance scale $\sigma^2 s$:

$$p_s(x,y) = (2\pi\sigma^2 s)^{-D/2} \exp\left(-\frac{\|y-x\|^2}{2\sigma^2 s}\right), \qquad s > 0. \tag{23}$$

For a finite target, define the h-function:

$$h(x,t) = \sum_{k=1}^{K} w_k \frac{p_{1-t}(x,a_k)}{p_1(x_0,a_k)}, \qquad t < 1. \tag{24}$$

Each component term satisfies the backward heat equation, so $\partial_t h + \frac{\sigma^2}{2}\Delta h = 0$, with normalization $h(x_0,0) = 1$. On any truncated subinterval $[0,T]$ with $T < 1$, the path mixture $P^\star$ is absolutely continuous with respect to $\mathbb{W}_{x_0}$ with Radon–Nikodym derivative $h(X_T,T)$. The conditional trajectory density given endpoint a_k evaluates to $p_{1-T}(X_T,a_k)/p_1(x_0,a_k)$; linearly weighting by w_k proves this density identity directly without invoking generalized functions at $t = 1$.

The transition kernel of the target process is given by

$$P^\star(X_t \in \mathrm{d}y \mid X_s = x) = p_{t-s}(x,y)\frac{h(y,t)}{h(x,s)}\,\mathrm{d}y, \qquad s < t < 1, \tag{25}$$

confirming that the mixture process $P^\star$ retains the Markov property. By Itô's formula, $\mathrm{d}h(X_t,t) = \sigma \nabla h(X_t,t) \cdot \mathrm{d}B_t$ under the reference measure. The Doob h-transform yields the forward drift $b^\star(x,t) = \sigma^2 \nabla \log h(x,t)$. Noting that $\nabla_x \log p_{1-t}(x,a_k) = \frac{a_k - x}{\sigma^2(1-t)}$, we obtain:

$$\sigma^2 \nabla_x \log h(x,t) = \sum_{k=1}^{K} \beta_k(x,t)\frac{a_k - x}{1-t} = \frac{\mathbb{E}[X_1 \mid X_t = x, c, \mathcal{C}] - x}{1-t}. \tag{26}$$

The agreement between Gaussian posterior weights $\beta_k(x,t)$ and heat-kernel ratios holds because:

$$\varphi_{k,t}(x) = \frac{p_t(x_0,x)p_{1-t}(x,a_k)}{p_1(x_0,a_k)}, \tag{27}$$

where $p_t(x_0,x)$ forms a common factor independent of k. Under appropriate regularity and moment conditions, this extends directly to continuous collection-averaged targets by integrating over π_c^+.

The intermediate spatial state density $\rho_t(x) = \sum_k w_k \varphi_{k,t}(x)$ satisfies the Fokker–Planck equation:

$$\partial_t \rho_t = -\nabla \cdot (b^\star \rho_t) + \frac{\sigma^2}{2}\Delta \rho_t, \qquad \rho_t \Rightarrow \tilde{\pi}_{c,\mathcal{C}} \quad (t \uparrow 1). \tag{28}$$

This differential relation characterizes the limiting distribution of the stochastic bridge at $t = 1$, though it does not imply convergence for deterministic, non-stochastic trajectory rollouts.

B.5 Control-Energy Interpretation

For an arbitrary drifted diffusion matching the initial condition x_0 and noise scale σ, Girsanov's theorem and standard integrability constraints yield:

$$\mathrm{KL}(P \parallel \mathbb{W}_{x_0}) = \frac{1}{2\sigma^2}\mathbb{E}_P\left[\int_0^1 \|b(X_t,t)\|^2 \,\mathrm{d}t\right]. \tag{29}$$

Equation (29) follows by substituting $\mathrm{d}X_t = b\,\mathrm{d}t + \sigma\,\mathrm{d}B_t^P$ into the log-likelihood ratio and taking expectations, where the martingale stochastic integral vanishes under square-integrability. For smoothed, finite-entropy problems, the bridge process minimizes expected drift energy subject to endpoint boundary conditions. This cost represents an abstract generative control effort rather than physical mechanical work. For a deterministic Dirac target at endpoint y, the singular drift $b(X_t,t) = \frac{y-X_t}{1-t}$ exhibits an expected power profile of

$$\mathbb{E}\left[\|b(X_t,t)\|^2\right] = \|y-x_0\|^2 + \sigma^2 D\frac{t}{1-t}, \tag{30}$$

whose integral over $t \in [0,1]$ diverges. This behavior aligns with the necessity of atomic regularization discussed in Appendix B.2.

B.6 Endpoint Regression and Resampling

Let $R = (X_t, t, c)$ be the regression input, let $Y = X_1$, and define $m(R) = \mathbb{E}[Y \mid R]$ assuming finite second moments. Expanding the expected square and conditioning on R causes cross-terms to vanish:

$$\mathbb{E}\left[\|f(R)-Y\|^2\right] = \mathbb{E}\left[\|f(R)-m(R)\|^2\right] + \mathbb{E}\left[\|Y-m(R)\|^2\right]. \tag{31}$$

Because the second term is invariant to the choice of model f, the conditional mean $m(R)$ uniquely minimizes the mean-squared error almost everywhere. Normalizing the loss by dimension $(1/D)$ scales the objective uniformly without altering its minimizer. Truncating the time sampling domain to $t \in [0, 1-\epsilon_t]$ identifies $m(R)$ restricted to that interval. Continuous time sampling does not directly constrain isolated point evaluations at $t = 0$; macro-level readout predictions rely on the continuous limit $f^\star(x_0, 0, c) = \mathbb{E}[X_1 \mid c]$.

For any $t < 1$, the prediction errors for endpoint f_θ and drift b_θ are connected via:

$$\|b_\theta(x,t,c) - b^\star(x,t,c)\|^2 = \frac{\|f_\theta(x,t,c) - f^\star(x,t,c)\|^2}{(1-t)^2}. \tag{32}$$

On $t \le 1-\epsilon_t$, the drift error remains bounded by ϵ_t^{-2} times the endpoint loss. Near $t \to 1$, small unweighted endpoint regression errors do not guarantee bounded drift errors.

To justify categorical candidate resampling, define individual candidate losses $\ell_\theta(a_k) = \mathbb{E}_{t,Z}[\|f_\theta(X_t(a_k),t,c) - a_k\|^2]$. Expectation over $k \sim w$ yields:

$$\mathbb{E}_{k\sim w}[\ell_\theta(a_k)] = \sum_{k=1}^{K} w_k \ell_\theta(a_k). \tag{33}$$

Holding candidates and weights fixed during actor updates ensures that gradients of both formulations match in expectation under standard Leibniz integral conditions. Sampling a single endpoint index k reduces candidate evaluation overhead during training. Analytically summing over candidate indices eliminates index-sampling variance, though the empirical variance of all-candidate estimators remains subject to noise introduced by time and Gaussian realization sampling.

C Deployment Readouts and Analytical Properties

C.1 One-Step Prediction and Weighted Projection

At the source ($t = 0$), $X_0 = x_0$ is deterministic given c. Consequently, the continuous conditional-mean predictor evaluates to $f^\star(x_0, 0, c) = m_c$, where $m_c = \mathbb{E}[X_1 \mid c]$. A single Euler step produces

the state update $x_0 + (f_\theta(x_0, 0, c) - x_0) = f_\theta(x_0, 0, c)$. The population-optimal one-step output is therefore m_c, invariant to the noise scale $\sigma > 0$.

Direct weighted squared-error projection of the target distribution onto a deterministic predictor $g(c)$ yields the identical optimum, as $\mathbb{E}\left[\|g(c) - X_1\|^2\right]$ is uniquely minimized by $g(c) = m_c$. The bridge framework additionally parameterizes intermediate conditional means away from the source ($t > 0$). While this equivalence holds for population-level objectives, empirical finite-sample optimization and multi-step readouts can diverge. Gaussian fitting to empirical endpoints can instead use expected negative log-likelihood with a fixed covariance, an explicit eigenvalue floor $\Sigma \succeq \epsilon I_D$ for $\epsilon > 0$, or a smoothed target. Positive definiteness alone does not prevent covariance collapse: for a rank-deficient empirical target, eigenvalues can approach zero in unsupported directions and the negative log-likelihood is unbounded below. Separately, the relative entropy KL from an atomic measure to a nondegenerate Gaussian is infinite, so it is not a finite fitting objective.

C.2 Two-Step Readout and Gaussian Mean Shift

For any predictor, two consecutive Euler steps reduce explicitly to

$$x_{1/2} = \frac{x_0 + f_\theta(x_0, 0, c)}{2}, \qquad a_{2\text{step}} = f_\theta\left(x_{1/2},\, \frac{1}{2},\, c\right). \tag{34}$$

Evaluated at the population-optimal predictor, the intermediate state is $x_{1/2} = \frac{1}{2}(x_0 + m_c)$. The conditional density of the random variable $X_{1/2}$ given $X_1 = y$, evaluated at $x_{1/2} = (x_0 + m_c)/2$, is proportional to

$$\exp\left(-\frac{\left\|\frac{x_0+m_c}{2} - \frac{x_0+y}{2}\right\|^2}{2(\sigma^2/4)}\right) = \exp\left(-\frac{\|y - m_c\|^2}{2\sigma^2}\right). \tag{35}$$

Applying Bayes' rule yields the optimal two-step readout:

$$a^\star_{2\text{step}} = \frac{\int y \exp\left(-\frac{\|y-m_c\|^2}{2\sigma^2}\right) \pi_c^+(\mathrm{d}y)}{\int \exp\left(-\frac{\|y-m_c\|^2}{2\sigma^2}\right) \pi_c^+(\mathrm{d}y)}. \tag{36}$$

For a fixed empirical candidate collection, this simplifies to a weighted sum $\sum_{k=1}^K \widetilde{w}_k a_k$, where

$$\widetilde{w}_k = \frac{w_k \exp\left(-\frac{\|a_k-m\|^2}{2\sigma^2}\right)}{\sum_{j=1}^K w_j \exp\left(-\frac{\|a_j-m\|^2}{2\sigma^2}\right)}, \qquad m = \sum_{k=1}^K w_k a_k. \tag{37}$$

Define the smoothed spatial density $p_{\sigma,c}(x) = \int \mathcal{N}(x; y, \sigma^2 I_D)\, \pi_c^+(\mathrm{d}y)$. Differentiating this kernel density estimator gives:

$$\nabla \log p_{\sigma,c}(x) = \frac{1}{\sigma^2}\left(\frac{\int y\, \mathcal{N}(x; y, \sigma^2 I_D)\, \pi_c^+(\mathrm{d}y)}{p_{\sigma,c}(x)} - x\right). \tag{38}$$

Evaluating this gradient at $x = m_c$ yields Eq. (13). Although the initial state x_0 cancels from this posterior expression once the target is fixed, the target measure itself retains dependence on x_0 through context and candidate generation. The two-step output remains a conditional mean; for instance, given two symmetric endpoints of equal weight, it evaluates strictly to their midpoint rather than mode-selecting or maximizing a critic function.

C.3 Sensitivity of the Conditional Endpoint Predictor

Let $m_t(x) = \mathbb{E}[Y \mid X_t = x, c]$ with $Y = X_1$. For $0 < t < 1$, the transition likelihood is Gaussian with mean $(1-t)x_0 + ty$ and variance $\sigma^2 t(1-t) I_D$. Its score function with respect to x is:

$$\nabla_x \log p(X_t = x \mid Y = y, c) = -\frac{x - (1-t)x_0 - ty}{\sigma^2 t(1-t)}. \tag{39}$$

Subtracting the posterior expectation over Y yields:

$$\nabla_x \log p(y \mid x, c) = \frac{y - m_t(x)}{\sigma^2(1-t)}. \tag{40}$$

Differentiating under the posterior integral—valid for finite mixtures and target distributions satisfying the required moment bounds—gives:

$$\nabla_x f^\star(x,t,c) = \frac{\mathrm{Cov}(X_1 \mid X_t = x, c)}{\sigma^2(1-t)}. \tag{41}$$

Equation (41) shows that local sensitivity is governed jointly by posterior endpoint uncertainty, time t, and bridge scale σ. Because the conditional covariance $\mathrm{Cov}(X_1 \mid X_t = x, c)$ itself implicitly depends on σ, this identity does not imply a monotonic relationship between noise scale and numerical stability or task completion.

C.4 Stochastic Sampling, Deterministic Readout, and Dither

The continuous target process satisfies the SDE $\mathrm{d}X_t = b^\star(X_t,t,c)\,\mathrm{d}t + \sigma\,\mathrm{d}B_t$, recovering the specified endpoint distribution by construction. An Euler–Maruyama discretization implements updates of the form:

$$x_{j+1} = x_j + \Delta\, b_\theta(x_j, t_j, c) + \sigma\sqrt{\Delta}\,\xi_j, \qquad \xi_j \sim \mathcal{N}(0, I_D), \quad t_j = j\Delta. \tag{42}$$

Practical stochastic readouts omit process noise on the terminal step, terminating in an explicit endpoint prediction f_θ. Under finite discretization steps, function approximation errors, time cutoffs, and step sizes alter the resulting empirical endpoint distribution. Due to the terminal singularity at $t = 1$, exact endpoint recovery or standard weak-order convergence claims require specialized boundary analysis.

By Eq. (28), the formal probability-flow velocity for $0 < t < 1$ is defined as:

$$v_{\mathrm{PF}}(x,t,c) = b^\star(x,t,c) - \frac{\sigma^2}{2}\nabla_x \log \rho_t(x). \tag{43}$$

Substituting v_{PF} into the continuity equation $\partial_t \rho_t = -\nabla\cdot(v_{\mathrm{PF}}\rho_t)$ reproduces the Fokker–Planck equation. On any subinterval $[t_0, T] \subset (0,1)$, initializing with samples from ρ_{t_0} under sufficiently regular velocity fields yields exact spatial marginals. However, integrating a deterministic, non-branching vector field from a single Dirac point source x_0 traces out a single trajectory, failing to preserve marginals without accounting for the boundary singularity. The deployed SCOUT readout integrates the drift model b_θ directly without stochastic noise or score-based probability-flow corrections; it acts as a deterministic control policy rather than a distribution sampler.

During simulation data collection, the single-step prediction receives additive output noise with standard deviation 0.05; real-world collection instead uses eight Euler–Maruyama steps. This dither provides localized exploration around predicted means rather than simulating a Brownian bridge process.

C.5 Closed Form for a Gaussian Endpoint Target

Assume $X_1 \sim \mathcal{N}(m, s_1^2 I_D)$ with $s_1 > 0$, independent of the reference Brownian motion. The spatial marginals and cross-covariances evaluate to:

$$\begin{aligned} \mathbb{E}[X_t] &= (1-t)x_0 + tm, \\ \mathrm{Var}(X_t) &= \left[t^2 s_1^2 + \sigma^2 t(1-t)\right] I_D, \\ \mathrm{Cov}(X_1, X_t) &= t s_1^2 I_D. \end{aligned} \tag{44}$$

For $0 < t < 1$, joint Gaussian conditioning yields the exact conditional endpoint predictor and drift field:

$$f^\star(x,t) = m + \frac{t s_1^2}{t^2 s_1^2 + \sigma^2 t(1-t)}\big[x - (1-t)x_0 - tm\big], \qquad b^\star(x,t) = \frac{f^\star(x,t) - x}{1-t}. \tag{45}$$

At $t = 0$, the predictor evaluates to $f^\star(x_0, 0) = m$. As the variance vanishes ($s_1 \downarrow 0$), $f^\star(x,t)$ converges to the deterministic terminal state m, and the drift field reduces to $b^\star(x,t) = \frac{m-x}{1-t}$. This closed-form solution serves as an analytical baseline for evaluating learned endpoint predictions and drift trajectories away from $t = 1$.

D CANDIDATE REWEIGHTING AND ESS CONTROL

D.1 KL-REGULARIZED CANDIDATE IMPROVEMENT

For a fixed set of candidates, let $u_k = 1/K$ denote the uniform prior and let $\Delta_K = \{w \in \mathbb{R}^K : w_k \geq 0,\ \sum_{k=1}^K w_k = 1\}$ denote the probability simplex. The index-space optimization objective is

$$\max_{w \in \Delta_K} \left[\sum_{k=1}^K w_k q_k - \frac{1}{\alpha} \sum_{k=1}^K w_k \log(K w_k) \right]. \tag{46}$$

For $\alpha > 0$, this objective is strictly concave. Differentiating the Lagrangian with respect to w_k (where ζ is the multiplier for the normalization constraint) yields $q_k - \alpha^{-1}(\log(Kw_k) + 1) + \zeta = 0$, implying $w_k \propto \exp(\alpha q_k)$. Subtracting the sample mean $\bar{q} = \frac{1}{K}\sum_{k=1}^K q_k$ from the scores leaves the resulting normalized weights invariant. Because repeated candidate actions possess identical scores, summing their index-space weights yields the corresponding action-space probability mass relative to the proposal distribution. The optimal objective value evaluates to $\frac{1}{\alpha}\log\left(\frac{1}{K}\sum_{k=1}^K \exp(\alpha q_k)\right)$.

This formulation represents a candidate-restricted improvement in the estimated value. It does not correspond to an exact exponential tilt of the continuous VLA action distribution, nor does it strictly guarantee an improvement in the true environment return. Notably, although the reference action a_{ref} is included in the candidate pool, the uniform candidate proposal is not equivalent to the reference policy.

D.2 KL, RÉNYI DIVERGENCE, AND NORMALIZED ESS

For a probability measure μ, a bounded score function A, and the exponentially tilted measure $\widetilde{\mu}(\mathrm{d}a) = Z^{-1}\exp(\alpha A(a))\,\mu(\mathrm{d}a)$, where $Z = \mathbb{E}_\mu[\exp(\alpha A)]$, the relative entropy satisfies:

$$\mathrm{KL}(\widetilde{\mu} \parallel \mu) = \alpha \mathbb{E}_{\widetilde{\mu}}[A] - \log Z \leq \alpha\,(\sup A - \mathbb{E}_\mu[A])\,. \tag{47}$$

This inequality relies on the trivial bound $\mathbb{E}_{\widetilde{\mu}}[A] \leq \sup A$ and Jensen's inequality $\log \mathbb{E}_\mu[\exp(\alpha A)] \geq \alpha \mathbb{E}_\mu[A]$. Evaluated on a discrete finite collection, this yields $\mathrm{KL}(w \parallel u) \leq \alpha\,(\max_k q_k - \bar{q})$.

The normalized effective sample size $\mathrm{ESS}(w) = 1/(K\sum_{k=1}^K w_k^2)$ lies in $[1/K, 1]$. Uniform weights attain 1; for finite softmax scores and $K > 1$, the lower endpoint is approached but not attained. Relative to the uniform index distribution u, the order-two Rényi divergence (Van Erven & Harremoës, 2014) is directly related to the ESS:

$$D_2(w \parallel u) = \log\left(\sum_{k=1}^K \frac{w_k^2}{u_k}\right) = \log\left(K\sum_{k=1}^K w_k^2\right) = -\log \mathrm{ESS}(w). \tag{48}$$

By the concavity of the logarithm, the KL divergence is bounded by the Rényi divergence:

$$\mathrm{KL}(w \parallel u) = \mathbb{E}_w\left[\log\left(\frac{w}{u}\right)\right] \leq \log \mathbb{E}_w\left[\frac{w}{u}\right] = D_2(w \parallel u), \tag{49}$$

which proves Eq. (5). Consequently, enforcing an ESS floor κ for a given context guarantees that $\mathrm{KL}(w \parallel u) \leq -\log\kappa$ for that context. Mapping candidate indices to their action values cannot increase KL divergence. Here, identical actions have identical critic scores, so w_k/u_k is constant within each duplicate-action group and equality holds:

$$\mathrm{KL}(\tilde{\pi}_{c,\mathcal{C}} \parallel \nu_{c,\mathcal{C}}) = \mathrm{KL}(w \parallel u), \qquad \nu_{c,\mathcal{C}} = \frac{1}{K}\sum_{k=1}^K \delta_{a_k}.$$

D.3 IMPLEMENTED CONTROLLER AND GEOMETRIC SCOPE

The controller monitors the batch-mean normalized ESS, denoted $\overline{\mathrm{ESS}}$, and dynamically updates the inverse temperature:

$$\alpha \leftarrow \begin{cases} \max(0.9\alpha,\ 10^{-3}), & \text{if } \overline{\mathrm{ESS}} < \kappa, \\ \min(1.01\alpha,\ 1000), & \text{otherwise}, \end{cases} \qquad \alpha_0 = 50. \tag{50}$$

The threshold is $\kappa = 0.25$ in simulation and $\kappa = 0.5$ in real-world experiments. The controller executes after each actor update without delay or threshold annealing, and the scalar α is persisted throughout the training run. This feedback mechanism regulates batch-level concentration; it does not impose a hard ESS floor on every individual update or for every context. A batch-mean ESS threshold $\overline{\mathrm{ESS}} \geq \kappa$ does not ensure a per-context ESS floor or the corresponding bound $\overline{\mathrm{KL}(w \parallel u)} \leq -\log \kappa$. For fixed $K > 1$, it does imply the weaker bound

$$\overline{\mathrm{KL}(w \parallel u)} \leq \frac{1 - \overline{\mathrm{ESS}}}{1 - 1/K} \log K \leq \frac{1 - \kappa}{1 - 1/K} \log K, \qquad \kappa \in [1/K, 1].$$

The first inequality follows by bounding $-\log x$ by its chord on $[1/K, 1]$. This is conditional on the batch meeting the threshold; the implemented feedback controller does not enforce it exactly.

Candidate displacement is governed separately by the proposal construction. Directed candidates take prescribed step sizes $j\eta$, while Gaussian perturbations exhibit unbounded support, where ρ defines a standard deviation rather than a strict radius limit. Replay actions are exempt from the directed-candidate distance bounds.

Importantly, uniform weighting over the candidate pool does not necessarily recover the reference action. Given the candidate composition defined in Eq. (2), the empirical mean of the uniform proposal is:

$$\frac{1}{12} \sum_{k=1}^{12} a_k = x_0 + \frac{a_{\mathrm{exec}} - x_0 + \sum_{i=1}^{6} \xi_i + 10\eta \widehat{g}(c)}{12}. \tag{51}$$

Thus, proposal asymmetry can bias the mean even when all sampled scores are equal. If the critic evaluates identically across the local action space, its gradient is zero, the directed term vanishes, and the mean shifts solely based on the executed and noise terms.

D.4 Action-Distribution Visualization

Figures 2 and 3 aggregate critic-weighted action targets over five LIBERO-PRO training runs, one per layout. For each run, we cache 60 contexts from its final replay journal: three decisions from each of 20 online-phase episodes. This gives 300 matched contexts across the five runs. The observations, reference chunks, and recorded executed chunks are held fixed across all evaluated checkpoints. For each context, we construct eight candidate collections using Eq. (2), reusing the same Gaussian perturbations across checkpoints. The four directed proposals are regenerated using the critic at each checkpoint. Aggregation assigns equal weight to each run, context, and proposal draw, with candidate weights normalized within each collection.

For each context, we define a fixed projection basis. The first direction e_1 is the normalized conservative critic gradient at a_{ref}, evaluated using the *final* checkpoint. The second direction e_2 is the normalized component of $a_{\mathrm{exec}} - a_{\mathrm{ref}}$ orthogonal to e_1. Candidate corrections are projected as

$$u_k^{\mathrm{proj}} = e_1^\top (a_k - a_{\mathrm{ref}}), \qquad v_k^{\mathrm{proj}} = e_2^\top (a_k - a_{\mathrm{ref}}).$$

The superscript proj distinguishes these coordinates from the uniform index probabilities $u_k = 1/K$. Critic scores are evaluated in the original action-chunk coordinates before projection. These context-specific directions are held fixed across both sweeps. Consequently, the pooled coordinates describe corrections relative to each context's basis, rather than a single physical action direction shared by all contexts.

Let s index the evaluated checkpoint, with candidate scores $q_k^{(s)}$, mean score $\bar{q}^{(s)}$, and controller inverse temperature α_s. Suppressing the run, context, and proposal-draw indices, the visualization uses

$$w_k^{(s)}(\beta) = \frac{\exp\Big\{\beta \alpha_s (q_k^{(s)} - \bar{q}^{(s)})\Big\}}{\sum_{j=1}^{K} \exp\Big\{\beta \alpha_s (q_j^{(s)} - \bar{q}^{(s)})\Big\}}. \tag{52}$$

The multiplier β controls an offline reweighting sweep: $\beta = 0$ gives uniform candidate weights, while $\beta = 1$ recovers Eq. (4).

Critic-weighting sweep. Figure 2 holds each run's final-checkpoint candidates, scores, and inverse temperature fixed and displays $\beta \in \{0, 0.25, 0.5, 0.75, 1\}$. Each curve shows the one-dimensional

marginal target density along the action-correction coordinate u. The three axes represent action correction, critic-weighting strength, and probability density. Only the candidate weights change across these curves; β is neither training progress nor internal bridge transport time.

Training-progress sweep. Figure 3 fixes $\beta = 1$ and evaluates four checkpoint groups: an early checkpoint after warmup, approximately one-third and two-thirds through training, and the final checkpoint. We write training progress as e/E, where e counts completed episodes and $E = 300$ is the training budget. This notation distinguishes training progress from the internal transport time t used by the bridge.

The actual mean completed-episode fractions across runs are 0.0867, 0.3367, 0.6627, and 1. For compact presentation, the four snapshots are evenly spaced and labeled approximately 0.1, $1/3$, $2/3$, and 1. Their horizontal spacing is schematic rather than proportional to elapsed training. The first group uses a critic trained through warmup, not an untrained initialization. Directed proposals and critic scores follow the selected checkpoint. Final inverse temperatures come from the saved checkpoint payloads; intermediate values are recovered from the controller log immediately following the checkpoint step.

Density estimation and display. We smooth the aggregated weighted candidates with Gaussian kernels of bandwidth 0.09 in projected action units. Figure 2 shows the marginal density $p(u)$ using one-dimensional kernels, whereas Figure 3 shows the joint density $p(u, v)$ using isotropic two-dimensional kernels. The marginal is computed from all weighted candidates, including those outside the displayed joint-density window.

The weighting figure displays $u \in [-0.35, 1.15]$. The training figure displays $u \in [-0.4, 1.0]$ and $v \in [-0.4, 0.7]$, with the same camera, coordinate limits, and vertical density scale across all four surfaces. Both figures retain full-support density normalization; densities are not renormalized within the displayed windows. Each curve or surface represents one evaluated condition, without interpolation between conditions. The $\beta = 1$ curve and the final training surface use the same final-checkpoint weighted candidates, displayed as marginal and joint densities, respectively.

These visualizations characterize the constructed training targets; they do not measure the learned actor's action distribution. Shading conveys surface geometry rather than uncertainty. Changes across training checkpoints jointly reflect the critic scores, directed proposals, and controller inverse temperature. Because the projection uses the final critic gradient, increased alignment with e_1 is a retrospective descriptive diagnostic rather than independent evidence of improved task success.

E Implementation Details

E.1 Shared VLA Interface

RL Token Construction. We prepare the RL token with the open-source RLT reproduction (Yang et al., 2026), following RLT (Xu et al., 2026). The token is a 2048-dimensional compressed representation of the post-transformer image-prefix embeddings. A two-layer, eight-head cross-attention encoder uses one learned query, and a mirror decoder reconstructs the prefix embeddings with mean squared error. The VLA is frozen during this preparation stage. Token preparation runs for 20,000 updates in both simulation and real-world experiments, with batch size 32 in simulation and 16 in real-world experiments. Both the VLA and token module remain frozen online. One server request computes the image prefix once and returns the token and a reference chunk sampled using that prefix.

VLA sampling and requests per method. The VLA flow sampler uses ten Euler steps from standard Gaussian noise of shape $H \times 32$. The server draws fresh noise from its own RNG stream, seeded once at startup, and does not reseed for evaluation trials. Its deterministic flag disables dropout; it does not remove source-noise sampling. Thus refiner evaluation is deterministic conditional on the supplied context and snapshot, while the reference chunk remains stochastic. SCOUT, CFM, and RLT use one VLA request per decision. DSRL$_{RLT}$ uses a feature request followed by a noise-conditioned denoising request; CNN-based DSRL uses its own visual encoder and one denoising request.

E.2 SCOUT ARCHITECTURE DETAILS

Figure 7 illustrates the architecture of SCOUT. Like RLT, SCOUT keeps the VLA frozen and refines its reference chunk from the same context of RL token, proprioception, and reference; unlike RLT, whose actor outputs a refined chunk in one pass and is trained by backpropagating through the critic, SCOUT's actor is an endpoint predictor trained by regression toward critic-weighted candidate chunks and read out in two drift steps from the reference.

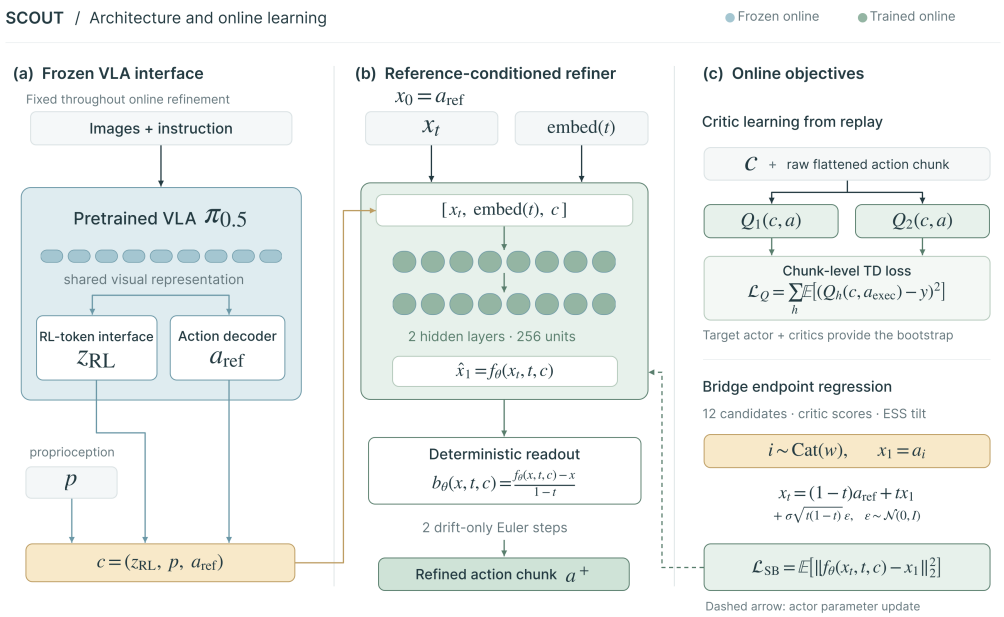


Figure 7: **Architecture design.** Context c integrates the frozen VLA RL token, proprioception, and reference chunk. Twin critics Q_h train on temporal-difference targets y, while the small endpoint predictor MLP learns from critic-weighted candidate endpoints $x_1 = a_i$ with $i \sim \text{Cat}(w)$.

SCOUT actor network. In simulation, action chunks have $H = 10$ steps of $d_a = 7$ ($D = 70$) and proprioception is 8-dimensional; in real-world experiments, $H = 30$, $d_a = 14$ ($D = 420$), and proprioception is 14-dimensional. The actor encodes its context $c = (z_{\text{RL}}, p, a_{\text{ref}})$ with three separate linear projections: the RL token to 256 dimensions followed by LayerNorm, and proprioception and the reference chunk to 64 and 256 dimensions, each followed by LayerNorm and $\tanh$. All LayerNorm operations in these input encoders and the actor and critic trunks have no learned affine parameters. The actor encoders give a 576-dimensional context feature. Transport time t is embedded with 64 sinusoidal features at 32 log-spaced frequencies between 1 and 1000. The endpoint predictor f_θ concatenates the transport state x_t, the time embedding, and the context feature, and passes them through two width-256 hidden layers (linear, LayerNorm without affine parameters, GELU) to a linear D-dimensional output. Weights are Glorot-uniform and biases are initialized to zero.

SCOUT critic networks. Each of the twin critics Q_1, Q_2 has its own token and proprioception encoders of the same form, concatenates the raw candidate chunk a directly, and uses the same two-layer trunk with a scalar output. The critic has no reference-chunk branch. The actor, the critic encoders, and the two critic heads share no parameters. Candidate scoring, proposal gradients, and TD bootstrapping all use $\min_h Q_h$. RLT's critics instead pass the action through a bottleneck projection $\tanh \text{LN}(Wa + b)$ before the trunk. CFM uses the same actor and critic networks as SCOUT; Table 3 compares their sizes with RLT's.

Table 3: Parameter counts measured from initialized parameter trees, including input encoders. Critic counts are per head. SCOUT and CFM share identical networks; RLT's are of comparable size. DSRL architectures are given in Appendix E.6.

Module	Simulation	Real world
SCOUT / CFM actor	809,094	1,078,628
RLT actor	774,790	954,724
SCOUT / CFM raw-action critic	691,265	781,249
RLT bottleneck critic	757,057	847,041

E.3 SCOUT Training Details

CFM shares the candidate construction, critic learning, optimization, and replay/update procedures below. Its Gaussian-source interpolation and velocity-regression objective are defined in Section 4.3; the bridge-actor noise scale and time cutoff below apply only to SCOUT.

Candidate proposals. Candidates are constructed according to Eq. (2). The critic gradient is normalized over the full flattened action chunk (Eq. (3)); a zero gradient produces four copies of the reference action. On hardware, the gradient is first smoothed along the temporal dimension using a centered five-step moving average with edge padding, then normalized. Simulation uses no gradient smoothing. The Gaussian perturbation scale ρ and directed step size η, as well as the bridge noise scale σ, are expressed in the actor's action coordinates. In simulation, these are LIBERO's native seven-dimensional commands, used without additional normalization. On the YAM robot, each action comprises twelve joint values and two gripper values. Joint values are expressed relative to the joint state at the start of the chunk, while gripper values remain absolute; each dimension is then scaled so that its first and ninety-ninth percentiles in the demonstration data map to -1 and $+1$. Thus, hardware proposal scales are measured in these scaled coordinates, not in radians. Candidate proposals themselves are neither clipped nor projected. The replay candidate reflects the command stored during data collection: on hardware, this is the command actually sent to the robot after five-step seam blending and per-tick joint-speed limiting; in simulation, it is the actor-produced command before any internal simulator clipping. Candidate endpoints, critic scores, and weights are treated as constants during actor fitting. Values of ρ and η are listed in Table 4. Since $j \in \{1, 2, 3, 4\}$ and $\|\widehat{g}(c)\| \leq 1$, directed proposals have Euclidean displacement at most 4η from the reference: 0.60 in simulation and 0.20 on hardware, in their respective action coordinates.

Actor objective. The actor minimizes the bridge regression loss of Eq. (7), with squared error averaged over action coordinates. Each training endpoint X_1 is a candidate drawn according to the critic weights w of Eq. (4), transport time is sampled as $t \sim \mathrm{Unif}[0, 1 - \epsilon_t]$ with $\epsilon_t = 10^{-3}$, and the bridge noise is $\sigma = 0.2$. The inverse temperature α is a single scalar per run, adjusted by the ESS controller after each actor step (Appendix D).

Critic target in simulation. Let $L_n \leq H$ be the actual ticks in simulation decision n, $r_{n,\ell}$ the tick reward, and ζ_n the terminal indicator. A tick-discounted return splits as

$$G_n = \sum_{\ell=0}^{L_n-1} \gamma^{\ell} r_{n,\ell} + (1 - \zeta_n)\gamma^{L_n} G_{n+1}. \tag{53}$$

The implementation stores fixed-length windows. In simulation, shortened final chunks are zero-padded in rewards and last-action-padded in actions to $H = 10$; both success and time-limit truncation set $\zeta_n = 1$. Thus using the stored discount γ^H agrees with the actual-length expression on nonterminal full chunks and has no effect on terminal shortened chunks. The bootstrap action, TD target, and critic loss are given in Eqs. (8)–(10).

The bootstrap action is the target actor's one-step endpoint prediction from the reference, rather than the two-step readout used at deployment. Here $\bar{Q}_h$, $f_{\bar{\theta}}$ are target networks, $\mathcal{D}$ is replay, and c_n^{boot} is the context at the window endpoint (the next decision in simulation). Targets are fixed during critic differentiation. Simulation reward is one on the first success tick and zero otherwise, with immediate

termination on success and no reward transformation. The per-tick discount is $\gamma = 0.99$, so a ten-tick bootstrap uses 0.99^{10}. Neither target noise nor learner action outputs are clipped; the simulator clips commands at execution.

Critic target in real-world experiments. For the YAM real-world experiments, replay windows span 30 rows and start every 25 rows, with an additional terminal window. The bootstrap context is taken 30 rows after the window start, rather than at the next replan. The chunk stops at the operator's success-button tick, and reward one is placed on that tick's row. The same success event can therefore appear in up to three overlapping replay transitions, each marked done. The discount is 0.99^{30} and terminal windows suppress bootstrapping; incomplete windows are zero-padded in rewards and last-row-padded in actions. Operator success, failure or done, and the 200-decision cap end an episode. Target-action noise is omitted ($\varepsilon = 0$ in Eq. (8)).

The stored action a_{exec} is the sequence of 30 commands actually sent to the robot, recorded per tick after the five-tick seam blend and the joint-speed limiter; its last five rows therefore come from the next plan. The bootstrap context at row 30 is obtained after the episode by running the frozen VLA on the observation recorded at that tick, which yields its RL token, proprioceptive state, and reference chunk; window starts that coincide with a replan reuse that plan's features.

CFM readouts on hardware. During collection, CFM draws a fresh source $U \sim \mathcal{N}(0, I_D)$ at each decision and integrates the learned velocity with eight Euler steps, with no additional output dither. Critic bootstrapping uses the target actor's one-step canonical readout from $U = 0$, with no target-action noise. Evaluation uses two Euler steps from the canonical source $U = 0$. The hardware replay-window and terminal-handling rules above also apply to CFM.

Optimizer and replay. All networks are trained with Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$), no weight decay, and constant learning rates. Actor gradients and the combined critic gradients are clipped separately to global norm 10. Replay holds up to 200,000 transitions and is sampled uniformly with replacement.

Warmup and update schedule. Before learning starts, the environment executes the frozen reference policy to fill replay. The learner then runs an initial block of critic updates with one actor update every second critic step; Polyak target updates and the ESS controller also run only on actor steps. For the actor and each critic, target parameters are updated as $\bar{\vartheta} \leftarrow (1 - \tau_{\text{EMA}})\bar{\vartheta} + \tau_{\text{EMA}}\vartheta$, where ϑ denotes the corresponding online parameters and $\tau_{\text{EMA}} = 0.005$. Before each episode, the environment checks whether both the replay-size and learner-step thresholds are met, and switches from the reference to the actor once they are. Afterwards, the learner performs five updates per new transition and idles when caught up, and actor snapshots are published to the environment at a fixed interval. Thresholds and snapshot intervals for simulation and real-world experiments are listed in Table 4.

E.4 Algorithm

Algorithm 1 summarizes the SCOUT learner and its collection and evaluation readouts. The replay-window construction above and the settings in Table 4 apply throughout.

E.5 Hyperparameters

Table 4 lists the hyperparameters of SCOUT in simulation and real-world experiments.

E.6 Baselines

RLT. Because the official RLT code is not publicly available, we use the open-source openpi-based reproduction (Yang et al., 2026), which also provides the RL-token preparation shared by all token-based methods. We keep its defaults for architecture, learning rates, Polyak rate, update ratio, actor-update period, and reference-action dropout (probability 0.5 in both simulation and real-world experiments); the settings we change are listed in Table 5. In addition, the actor objective weights the critic value by 0.1 and behavior cloning toward the reference by 10 during warmup and 5 online. Unlike SCOUT and CFM, RLT applies no gradient clipping.

Table 4: **SCOUT hyperparameters.** Values spanning both columns are shared by simulation and real-world experiments. Simulation settings apply to LIBERO and both LIBERO-PRO studies. CFM shares the candidate, critic, optimization, and replay settings; its actor objective and readouts differ. The bridge-actor rows apply only to SCOUT.

Setting	Simulation	Real world
Interface and budget		
Action chunk $H \times d_a$ / control rate	10×7 / 20 Hz	30×14 / 30 Hz
Rows executed per decision	10	25 (5-row blend)
Online episodes (incl. warmup)	300	300
Episode cap	Suite tick limit (App. F.3)	200 decisions
Evaluation trials	100	50
Candidates and critic-weighted target		
Candidates K	12: reference, replay, 6 Gaussian, 4 directed	
Gaussian std. per coordinate ρ	0.05	
Directed step size η	0.15	0.05
Gradient smoothing	None	5-row centered average
Initial / min / max α	50 / 10^{-3} / 1000	50 / 10^{-3} / 1000
ESS threshold κ (batch mean)	0.25	0.5
Bridge actor		
Bridge noise σ	0.2	
Time range t / cutoff ϵ_t	$[0, 1-\epsilon_t]$ / 10^{-3}	$[0, 1-\epsilon_t]$ / 10^{-3}
Critic and optimization		
Per-tick discount γ (per replay window)	0.99 (0.99^{10})	0.99 (0.99^{30})
Target-action noise std.	0.05	0
Actor / critic learning rate	3×10^{-4} / 3×10^{-4}	10^{-4} / 10^{-4}
Batch size / updates per transition	128 / 5	
Actor and target-update period	Every two critic updates	
Polyak rate τ_{EMA} / gradient-norm clip	0.005 / 10	
Replay capacity / sampling	200,000 / uniform with replacement	
Schedule and readouts		
Warmup transitions	500	350
Warmup learner steps	5,000	10,000
Snapshot interval (learner steps)	200	500
Collection readout	One-step prediction + 0.05 dither	Eight Euler–Maruyama steps
Evaluation readout	Two deterministic drift-only steps	

DSRL. DSRL uses the upstream `PixelSACLearner` from `jaxrl2` and does not use the RL token. Its encoder has four convolutional layers with 32 channels, strides $(2, 1, 1, 1)$, VALID padding, and group normalization, followed by a spatial softmax and a 50-dimensional latent; training applies random crops with padding four. It keeps its own replay buffer and optimizers. In real-world experiments, the encoder reads the front camera resized to 64×64. As in the upstream code, both DSRL variants mask bootstrapping only on success, so failure and time-limit transitions still bootstrap, whereas SCOUT, CFM, and RLT treat time-limit endings as terminal (Appendix E.3).

DSRL$_{RLT}$. This variant replaces the CNN with the RL-token and proprioception features used by SCOUT and RLT, so the noise-space policy sees the same state information as the action-space methods. The policy head is initialized with scale 0.01, its log standard deviations are clipped to $[-10, 2]$, and the tanh output is scaled by 1.0. Actor, critic, and temperature use separate Adam optimizers, and the entropy term is omitted from the critic backup. Each decision requires two VLA requests. The two DSRL arms therefore differ in encoder, runtime, and number of VLA calls, as well as in discount.

Table 5 lists the hyperparameters of all baselines in simulation and real-world experiments.

Table 5: **Baseline configuration.** Values are shared by simulation and real-world experiments unless a real-world value is given after "RW:". A tick is one control step (20 Hz in simulation, 30 Hz in real-world experiments). DSRL columns output a noise row that the frozen VLA decodes.

Setting	RLT	DSRL	DSRL$_{RLT}$
Algorithm	TD3+BC	SAC (`jaxrl2` `PixelSACLearner`)	SAC
Inputs	RL token, proprioception, reference chunk	64×64 agent-view image	RL token, proprioception
Encoder	As SCOUT (256/64/256)	4-layer CNN, spatial softmax, 50-d latent	Linear projections, widths 256/64
Actor trunk	2×256, LN/GELU	MLP $(128, 128, 128)$	3×128, LN/GELU
Actor output	Deterministic D-dim. chunk	32-dim. tanh-Gaussian noise row, tiled over H rows	
Critics / reduction	2 bottleneck-action / min	10 on raw noise action / mean	
Actor / critic / temp. LR	3×10^{-4} / 3×10^{-4} / –; RW: 10^{-4} / 10^{-4} / –	10^{-4} / 3×10^{-4} / 3×10^{-4}	
Batch / updates per transition	128 / 5	256 / 20	
Warmup	500 transitions, 5,000 learner steps; RW: 20,000 learner steps	500 noise-prior transitions	Noise prior until first actor update
Target-update cadence	Every two critic steps	Every critic step	
Polyak rate	0.005	0.005	
Entropy	–	Target -16, initial coefficient 1.0, auto-tuned	
Collection / evaluation	Output $+\mathcal{N}(0, 0.05^2)$ / noiseless output; RW: std. 0.01	Policy sample / tanh of policy mean	
Target-policy noise std.	0.05; RW: 0.01	–	
Reference-action dropout	0.5	–	
Joint-delta cloning weight	0; RW: 10	–	
Reward	As SCOUT	-1 per unsuccessful decision, 0 at success	
Per-tick discount	0.99	0.999; RW: 0.9993	0.998; RW: 0.9993

Algorithm 1. SCOUT learning and execution

Setup. Freeze the VLA and RL-token module. Initialize actor f_θ, twin critics Q_1, Q_2, their target networks, and replay buffer $\mathcal{D}$. Set $\alpha = 50$ and use $\kappa = 0.25$ in simulation or $\kappa = 0.5$ on hardware. Follow the warmup and snapshot schedule in Table 4. After warmup, perform five critic updates per new replay transition.

For each critic update:

1. Sample 128 replay windows containing contexts c_n, executed actions $a_{\text{exec},n}$, tick rewards, terminal flags ζ_n, and bootstrap contexts c_n^{boot}.
2. Form fixed TD targets using the target actor's one-step endpoint prediction and the minimum target-critic value (Eqs. (8)–(9)). Use target-action noise with standard deviation 0.05 in simulation and zero on hardware. Take one Adam step on the critic loss (Eq. (10)), clipping the combined critic gradient to norm 10.
3. **Every second critic update:**
   (a) *Construct the target distribution.* Build the 12 candidates in Eq. (2) using the current critics for the directed proposals. On hardware, smooth the gradient before normalization. Score each candidate with $q_k = \min_h Q_h(c, a_k)$ and compute weights w using Eq. (4).
   (b) *Sample bridge training pairs.* Hold candidate endpoints, scores, and weights fixed during actor differentiation. Independently for each context, sample $k \sim \text{Cat}(w)$, $t \sim \text{Unif}[0, 1 - 10^{-3}]$, and $Z \sim \mathcal{N}(0, I_D)$. Set $X_1 = a_k$ and construct X_t using Eq. (6) with $\sigma = 0.2$.

(c) *Update the actor.* Take one Adam step on Eq. (7), averaging the squared error over the batch and action coordinates. Clip the actor gradient to norm 10.

(d) *Adapt the tilt.* Compute the batch mean $\overline{\mathrm{ESS}}$ of $\mathrm{ESS}(w) = 1/(K\sum_k w_k^2)$ using these candidate weights. Update

$$\alpha \leftarrow \begin{cases} \max(0.9\alpha, 10^{-3}), & \text{if } \overline{\mathrm{ESS}} < \kappa, \\ \min(1.01\alpha, 1000), & \text{otherwise.} \end{cases}$$

(e) *Update the target networks.* Polyak-update the actor and both critic targets with coefficient $\tau_{\mathrm{EMA}} = 0.005$.

Collection. Use the frozen reference during warmup. Switch to the actor at an episode boundary once both warmup thresholds are met. Thereafter, start from a_{ref} and use:

- *Simulation:* one endpoint prediction with Gaussian output dither of standard deviation 0.05.
- *Hardware:* eight Euler–Maruyama steps, omitting noise in the final step.

Store executed commands, tick rewards, terminal flags, and window-end contexts according to the replay rules above.

Evaluation. Given c and a_{ref}, return a^+ using the two deterministic drift steps in Eq. (12). No candidate generation or critic evaluation is required.

F Experimental Setup

F.1 Base Policies

All base policies are $\pi_{0.5}$ models. In simulation, all LIBERO and LIBERO-PRO tasks use the $\pi_{0.5}$ LIBERO checkpoint released by Physical Intelligence (Intelligence et al., 2025b) without further fine-tuning, so each refiner adapts this single policy online. For real-world experiments, we first adapt $\pi_{0.5}$ to the bimanual I2RT YAM embodiment by fine-tuning it for 30,000 steps on 800 episodes of MolmoAct-2 Bimanual-YAM teleoperation data (Fang et al., 2026), and then fully fine-tune this model separately for each task on 100 task-specific demonstrations. For every task, SCOUT and all baselines share the identical frozen base policy. The real-world platform and tasks are described in Appendices F.5 and F.6.

F.2 Evaluation Protocol

Training budget and evaluation. Each refiner is trained online for 300 episodes in simulation and 300 episodes in real-world experiments, both including warmup. Only the final checkpoint is evaluated, with 100 trials per task in simulation and 50 in real-world experiments; the frozen base policy is evaluated on the same trials without online training. Real-world evaluation trials start from predetermined initial configurations that are identical across methods. Unless stated otherwise, each reported result comes from a single training run per method and task, and all simulation runs use random seed 7. The main comparisons therefore describe the evaluated training runs; variability across independently trained policies is assessed in the three-seed actor ablations. Aggregate success rates are the arithmetic mean of per-task success rates.

Success. In LIBERO and LIBERO-PRO, success is determined by each benchmark's symbolic goal conditions. In real-world experiments, one author judges success for all methods against a predetermined task rubric (Appendix F.6).

Throughput. We introduce *throughput* for real-world tasks to measure active execution efficiency. It is defined as the number of successful trials a policy completes within 600 seconds of accumulated execution time. The evaluation protocol follows these rules:

- **Scene Selection & Initialization:** Unlike the standard success evaluation (which uses ten scenes per task), throughput is measured on a continuous sequence of trials within a single base scene unseen during training. We select the easiest scene for each task to ensure a baseline level of success across all evaluated methods. At the start of each trial, object positions are slightly perturbed within a 5 cm radius to introduce local variation.

- **Time Accounting:** Execution time is tracked by summing the active durations of all trials, including failures. We strictly exclude scene setup and reset overhead, ensuring we measure active execution rather than wall-clock time.
- **Cutoff Rule:** Counting stops the moment the running total exceeds 600 seconds. The trial that crosses this threshold is excluded from the final count, regardless of its outcome.
- **Checkpoints:** All methods are evaluated using the same final checkpoints as the main success evaluation.

Compute. All training and evaluation runs use a workstation with a single NVIDIA GeForce RTX 4090 or 5090 GPU and 64 GB of RAM; all real-world experiments run on the RTX 5090 workstation.

F.3 LIBERO Simulation Setup

We evaluate on 20 LIBERO tasks: 19 from LIBERO-90 and one from LIBERO-10 (Table 7). Observations, the action representation, and per-suite episode limits follow the openpi LIBERO evaluation setup used to train the base checkpoint. Each decision executes all ten rows of the returned action chunk at 20 Hz before replanning.

Each task provides 50 fixed initial states. Training episode i and evaluation trial j start from states $i \bmod 50$ and $j \bmod 50$, respectively, so the 100 evaluation trials visit every state twice and share their initial states with online training. Every evaluation trial is counted, with no retries or discarded trials. LIBERO-PRO, in contrast, evaluates on held-out initial states (Appendix F.4).

F.4 LIBERO-PRO Simulation Setup

Following LIBERO-PRO (Zhou et al., 2025), each layout perturbs a LIBERO task by exchanging the initial placement regions of two or more objects in its BDDL initial conditions, while keeping object identities, fixtures, and the language instruction unchanged. We report ten layouts drawn from the LIBERO-10, Goal, and Object suites (Table 6). Layouts were selected before online training, using only the frozen base policy's success rate. Within each swapped region, object positions and rotations are sampled by LIBERO's standard reset.

Unlike in LIBERO, training and evaluation use disjoint initial states. Each layout has 50 training states (seeds 1000–1049) and is evaluated for 100 trials on held-out states. The layouts were generated in two campaigns that differ only in their held-out states: the first draws 50 evaluation states (seeds 2000–2049) and visits each twice, while the second draws 100 evaluation states (seeds 4000–4099) and visits each once, and additionally rejects any start in which an object or fixture moves more than 0.15 m during 20 zero-action settling steps. These experiments measure online adaptation to a fixed swapped layout and generalization to new initial states of that layout.

Table 6: **LIBERO-PRO layouts.** Each layout exchanges the initial placement regions of the listed objects.

Instruction	Exchanged regions
Put both the cream cheese box and the butter in the basket	Cream cheese ↔ tomato sauce; butter ↔ orange juice
Pick up the book and place it in the back compartment of the caddy	Black book ↔ white/yellow mug
Open the top drawer and put the bowl inside	Akita black bowl ↔ cream cheese
Put the cream cheese in the bowl	Akita black bowl ↔ wine bottle
Pick up the cream cheese and place it in the basket	Cream cheese ↔ milk
Pick up the salad dressing and place it in the basket	Salad dressing ↔ cream cheese
Pick up the butter and place it in the basket	Butter ↔ tomato sauce
Put the cream cheese in the bowl	Akita black bowl ↔ plate
Pick up the tomato sauce and place it in the basket	Tomato sauce ↔ milk
Pick up the orange juice and place it in the basket	Orange juice ↔ salad dressing

F.5 Real-World Platform

Platform and observations. The platform consists of two I2RT YAM arms, each with six joints and a parallel gripper, giving a 14-dimensional state and action (left joints, left gripper, right joints, right gripper). Three Intel RealSense cameras, one front-facing and one on each wrist, stream 640×360 images at 30 Hz, which are resized with padding to 224×224 for the VLA. The policy also receives the 14 joint and gripper positions and the task instruction.

Action execution. All methods run at 30 Hz. The VLA predicts 30-step action chunks, of which the first 25 steps are executed before replanning; the remaining five steps are blended into the next chunk, giving a replanning interval of about 0.83 s. Joint actions are expressed as offsets from the joint state at the start of the chunk, while gripper actions are absolute. The commanded change of each joint is limited to 0.05 rad per control step, except in Doll-Throwing, where the limit is removed so that the arms can reach throwing speed.

F.6 Real-World Tasks

Cube-Stacking. The robot picks up the blue cube, places it near the midpoint between the two arm mounts, and then stacks the other two cubes on it, giving a blue–red–yellow tower from bottom to top.

- *Evaluation setup.* Evaluation uses ten configurations repeated five times: two configurations with two blocks on the left and one on the right, three with two on the right and one on the left, two with all blocks on the left, one with all on the right, one with yellow already on blue requiring unstacking, and one with blue and red forming the base of a three-block pyramid.
- *Success criteria.* Success requires the final block to be placed on the correctly ordered stack without the stack falling or being knocked down. No fixed stability-duration threshold is imposed. A block leaving the arms' reachable space or the stack toppling during placement ends the trial in failure.

Ethernet-Insertion. The robot picks up one end of an Ethernet cable from a holder and inserts it into the rightmost port of a rack-mounted 8-port Ethernet switch.

- *Evaluation setup.* The switch is fixed in the same position during demonstration collection, refiner training, and evaluation. Evaluation uses ten cable-holder positions, each repeated five times, none of which appear in the demonstrations or online training.
- *Success criteria.* Success requires the RJ45 connector to be fully seated so that it cannot be pulled out with mild force.

Doll-Throwing. The robot picks up a doll from the table and throws it into a red bin on the same table.

- *Evaluation setup.* Three different dolls are used, with one doll and the bin on the table in each trial. Evaluation uses ten scenes that vary the bin and doll positions, with five trials per scene; the three dolls account for 10, 20, and 20 of the 50 trials, respectively.
- *Success criteria.* Success requires the thrown doll to land and remain in the bin. Multiple pickup attempts are allowed, but each trial permits only one throw.

LEGO-Click. The robot picks up a LEGO minifigure and clicks its feet onto the center of a black baseplate, with the figure facing the orange bar on the baseplate.

- *Evaluation setup.* The minifigure's legs are modified from their original assembly to ease teleoperated demonstration collection, and the same modified figure is used for training and evaluation. The baseplate is fixed on the table throughout training and evaluation.
- *Success criteria.* Success requires the minifigure to stand with its feet clicked onto the correct studs, face the orange bar, and not tip over under a mild push from the front or the left.

G ADDITIONAL RESULTS

LIBERO. Table 7 reports per-task results on the 20 LIBERO tasks.

Table 7: LIBERO successes out of 100 evaluation trials per task and method. Bold marks the highest count and underline marks the second-highest count in each row, including ties. Learned methods are evaluated at their final checkpoint after 300 online episodes.

Instruction	Frozen	**SCOUT**	CFM	RLT	DSRL$_{RLT}$	DSRL
Put both moka pots on the stove	70	90	65	85	**91**	76
Put the black bowl in the top drawer of the cabinet	61	**99**	**99**	93	96	80
Open the top drawer of the cabinet	1	**100**	**100**	**100**	**100**	97
Put the black bowl at the back on the plate	57	**97**	96	85	95	80
Put the middle black bowl on the plate	29	69	67	51	**89**	75
Put the moka pot on the stove	91	96	94	97	**98**	92
Close the bottom drawer of the cabinet	98	**99**	98	98	95	95
Close the top drawer of the cabinet	63	97	96	96	**98**	96
Put the black bowl in the top drawer of the cabinet	49	93	86	76	**96**	43
Put the black bowl on the plate	53	**100**	92	86	**100**	96
Close the microwave	87	94	83	**95**	73	83
Put the right moka pot on the stove	53	82	**86**	67	74	85
Put the white bowl on top of the cabinet	14	**100**	99	46	98	95
Pick up the tomato sauce and put it in the tray	30	73	49	43	**88**	70
Pick up the black bowl on the left and put it in the tray	50	99	92	62	98	**100**
Stack the right bowl on the left bowl and place them in the tray	51	**79**	70	73	66	32
Put the chocolate pudding to the right of the plate	75	98	92	98	**100**	96
Pick up the book and place it in the back compartment of the caddy	17	88	**100**	91	**100**	98
Pick up the book and place it in the left compartment of the caddy	55	97	93	92	93	**98**
Pick up the book and place it in the left compartment of the caddy	58	95	88	90	**97**	82

LIBERO-PRO. Table 8 reports results for the ten LIBERO-PRO layouts. Layout definitions and evaluation states are described in Appendix F.4.

Table 8: LIBERO-PRO successes out of 100 evaluation trials on held-out initial states per layout and method. Bold marks the highest count and underline marks the second-highest count in each row, including ties.

Instruction	Frozen	**SCOUT**	CFM	RLT	DSRL$_{RLT}$	DSRL
Put both the cream cheese box and the butter in the basket	13	**66**	2	10	54	25
Pick up the book and place it in the back compartment of the caddy	58	**100**	97	98	96	**100**
Open the top drawer and put the bowl inside	90	**99**	67	88	87	96
Pick up the salad dressing and place it in the basket	44	97	55	83	96	**98**
Put the cream cheese in the bowl	29	61	50	41	64	**74**
Pick up the cream cheese and place it in the basket	33	**98**	87	84	**98**	59
Pick up the butter and place it in the basket	57	**100**	92	93	**100**	**100**
Put the cream cheese in the bowl	2	**90**	5	10	66	46
Pick up the tomato sauce and place it in the basket	0	**100**	24	1	94	69
Pick up the orange juice and place it in the basket	7	**97**	64	83	76	78
Mean over 10 layouts	33.3	**90.8**	54.3	59.1	83.1	74.5

Consistency. As shown in Figure 8, SCOUT is the only method that improves its success rate over the base policy in all 30 simulation tasks. All other baselines regress on base-policy performance in at least two tasks. In particular, both DSRL and DSRL$_{RLT}$ lose at least 3 percentage points on the LIBERO tasks "Close the bottom drawer of the cabinet" and "Close the microwave," where the base policy succeeds in 98% and 87% of trials, respectively. At the other extreme, on the four tasks with base success below 10%, SCOUT reaches 96.8% on average versus

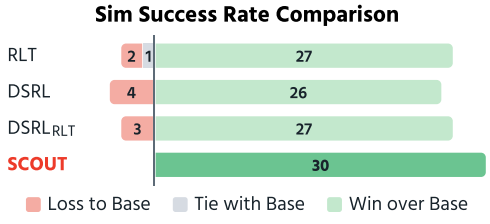


Figure 8: **Simulation success rate improvements/regression over base policy.**

48.5% for RLT. SCOUT therefore neither disrupts a strong base policy nor remains tied to a weak one. The CFM ablation (Section 4.3) suggests this stems from the point-source bridge: with identical critics and targets but a Gaussian source, CFM regresses on four tasks.

Table 9 breaks down the consistency analysis of Section 4.1 by benchmark. The baselines' regressions are concentrated on tasks where the base policy is already strong: 8 of their 13 regressions occur on the four tasks with base success of at least 85%, where there is little headroom and any departure from a working reference is costly. SCOUT makes its smallest gains on these same tasks but never falls below the base policy, and on the tasks where the base policy nearly always fails it still reaches high success rather than staying tied to the failing reference. SCOUT therefore adapts the size of its correction to the quality of the reference: it stays close when the reference already works and departs from it when it does not, which gives it the highest worst-case success of all methods (61%, against at most 54% for any baseline).

Table 9: **Changes relative to the base policy.** W / T / L: number of tasks on which a method's success is above, equal to, or below the base policy's. Worst: smallest per-task change in success rate (percentage points). Low-base mean: mean success rate on the four tasks with base success below 10% (one LIBERO task and three LIBERO-PRO layouts).

	LIBERO (20)		LIBERO-PRO (10)		
Method	W / T / L	Worst	W / T / L	Worst	Low-base mean
SCOUT	**20 / 0 / 0**	**+1**	**10 / 0 / 0**	**+9**	**96.8**
CFM	17 / 1 / 2	−5	8 / 0 / 2	−23	48.2
RLT	19 / 1 / 0	0	8 / 0 / 2	−3	48.5
DSRL$_{RLT}$	18 / 0 / 2	−14	9 / 0 / 1	−3	84.0
DSRL	16 / 0 / 4	−19	**10 / 0 / 0**	+6	72.5

Real-world results. Table 10 reports the success counts, throughput, and estimated speedup of every method on the four real-world tasks.

Table 10: **Real-world results.** Success: successful trials out of 50. Throughput: successful trials within 10 minutes of accumulated execution time. Speedup: throughput gain over the base policy divided by success-rate gain. Bold marks the best learned method in each success and throughput column.

	Cube-Stacking			Ethernet-Insertion			LEGO-Click			Doll-Throwing		
Method	Succ.	Thr.	Speedup	Succ.	Thr.	Speedup	Succ.	Thr.	Speedup	Succ.	Thr.	Speedup
Base policy	43	18	1.00	18	21	1.00	38	18	1.00	34	39	1.00
SCOUT	45	**46**	2.44	33	**43**	1.12	**46**	30	1.38	**42**	**52**	1.08
CFM	44	45	2.44	21	24	0.98	42	**31**	1.56	40	49	1.07
RLT	45	42	2.23	31	41	1.13	45	29	1.36	39	45	1.01
DSRL$_{RLT}$	**47**	21	1.07	**34**	29	0.73	42	21	1.06	0	0	–
DSRL$_{RLT}$	44	25	1.36	29	32	0.95	37	25	1.43	38	43	0.99

Inference cost. Table 12 reports the median total VLA inference time per decision, recorded during the evaluations of the three additional LIBERO-PRO layouts. Each simulation decision produces an action chunk covering ten control ticks. For DSRL$_{RLT}$, the reported time sums its two VLA calls within each decision before taking the median. In these runs, four evaluation processes shared a single $\pi_{0.5}$ server on an RTX 5090, so the timings characterize this implementation under shared-server contention. Methods that query the VLA once per decision take 252–376 ms, whereas DSRL$_{RLT}$, which decodes its noise action with a second VLA pass, takes 705–755 ms, 2.0–2.6 times longer than SCOUT on the same layout. The actor head adds 1.6–2.4 ms per decision for every method.

Table 11: **Cube-Stacking execution time.** A separate timing study with 20 evaluation trials per method, distinct from the 50-trial evaluation in Table 10. Episode time and execution steps are averaged over all trials.

Method	Success	Avg. episode time (s)	Avg. execution steps
Base policy	18/20	26.13	562.3
SCOUT	17/20	13.29	296.0
CFM	18/20	13.06	287.8
RLT	17/20	13.20	298.7
DSRL$_{RLT}$	20/20	33.28	357.6
DSRL	–	–	–

Table 12: **Inference cost.** Median total VLA inference time per decision (ms) on three LIBERO-PRO layouts, measured with four evaluation processes sharing one $\pi_{0.5}$ server. Each decision covers ten control ticks; DSRL$_{RLT}$ timings include both VLA calls within the decision.

Method	Bowl swap	Milk swap	Salad-dressing swap	VLA passes per decision
SCOUT	276	337	351	1
CFM	278	355	351	1
RLT	278	376	252	1
DSRL$_{RLT}$	710	755	705	2

H Actor Ablations and Deployment Readouts

In this section, we report online comparisons against direct action predictors and present a separate frozen-checkpoint readout study. Our results indicate that the SCOUT architecture and its deterministic two-step deployment configuration ($J = 2$) yield strong performance, outperforming the alternative formulations tested on this panel. While bridge derivations characterize fixed-target constructions and population regression objectives, our empirical comparisons evaluate learned policies in practice.

H.1 Online Actor Comparisons

Each online variant has three training seeds per layout. The SCOUT controls are newly trained runs for these ablations on an RTX 4090 GPU, separate from the main comparisons. Following the ablation protocol, learned policies receive 300 online episodes and are evaluated at the final snapshot using 100 deterministic trials per run. Table 13 reports all three supplied seed results, their per-layout means, and the equal-layout mean. The reference is the round-1 SCOUT control, trained contemporaneously with these comparisons.

Compared actors. The *hard-target bridge* keeps the bridge actor, candidates, critic and loss unchanged, but fixes the inverse temperature at $\alpha = 10^4$ and disables the ESS controller. The tilt weights $\mathrm{softmax}(\alpha(q_k - \bar{q}))$ favor higher-scoring candidates, with near-ties at score differences of order 10^{-4} able to share substantial probability mass. Each update resamples an endpoint from these weights, making this an approximately hard target rather than an exact argmax. The two direct predictors instead output a chunk $g_\psi(c)$ without conditioning on an intermediate bridge state or transport time. Direct weighted regression uses the objective:

$$\mathcal{L}_{\mathrm{weighted}}(\psi) = \mathbb{E}_{c,\mathcal{C}}\left[\sum_{k=1}^{K} w_k \left\| g_\psi(c) - a_k \right\|_2^2\right], \tag{54}$$

where $\mathcal{C} = \{a_k\}_{k=1}^K$ is the candidate collection. Hard-choice distillation instead predicts only the single highest-weighted candidate:

$$\mathcal{L}_{\mathrm{hard}}(\psi) = \mathbb{E}_{c,\mathcal{C}}\left[\left\| g_\psi(c) - a_{k^\star} \right\|_2^2\right], \qquad k^\star \in \arg\max_k q_k. \tag{55}$$

Table 13: **Online actor ablations on five LIBERO-PRO layouts.** Each task cell shows the mean successes out of 100 trials, followed by the three individual seed counts in parentheses. The final row averages equally over layouts and seeds. Bold marks the largest mean in each row, including ties.

Task / Instruction	SCOUT (Ours)	Hard-target bridge	Direct weighted	Direct hard-choice
T1: Put both the cream cheese box and the butter in the basket	**52.7** (53, 51, 54)	27.3 (38, 2, 42)	22.7 (27, 25, 16)	13.7 (30, 0, 11)
T2: Put the cream cheese in the bowl	**71.3** (73, 69, 72)	56.3 (52, 42, 75)	43.7 (40, 47, 44)	37.0 (26, 51, 34)
T3: Pick up the cream cheese and place it in the basket	94.7 (98, 94, 92)	**95.3** (92, 96, 98)	89.0 (90, 87, 90)	89.3 (90, 87, 91)
T4: Pick up the salad dressing and place it in the basket	**94.3** (90, 97, 96)	**94.3** (91, 100, 92)	88.0 (93, 75, 96)	93.3 (83, 97, 100)
T5: Pick up the butter and place it in the basket	**99.3** (100, 100, 98)	97.3 (93, 99, 100)	95.0 (98, 90, 97)	95.3 (95, 92, 99)
Mean / 100	**82.5**	74.1	67.7	65.7

Interpretation of Results. The round-1 SCOUT control averages 82.5 successes per 100 trials, compared with 74.1 for hard-target bridge training, 67.7 for direct weighted regression, and 65.7 for direct hard-choice distillation. The corresponding differences, computed before rounding, are 8.3, 14.8, and 16.7 successes per 100 trials. The improvements over the direct predictors are largest on T1 and T2. These online comparisons support the implemented bridge actor relative to the tested alternatives; they do not isolate actor fitting under a shared frozen critic and replay distribution.

H.2 FROZEN-CHECKPOINT READOUT STUDY

CFM readout configurations. Both CFM configurations in Table 14 evaluate the same trained seed-1 SCOUT (CFM) checkpoint; only the evaluation readout differs. The canonical readout, `cfmcanon`, starts from the fixed source $U = 0$, the mode of $\mathcal{N}(0, I)$. The stochastic readout, `cfm`, draws a fresh source $U \sim \mathcal{N}(0, I_D)$ at every decision, using sampler seed 0. Both integrate the learned velocity with J Euler steps. Given a context, `cfmcanon` is a deterministic point readout, while the only randomness in `cfm` is its source draw.

The shared checkpoint in this ablation was trained using a fresh Gaussian source and two Euler steps for data collection, with no additional output dither. Critic bootstrapping used the target actor's one-step canonical readout from $U = 0$, plus Gaussian target noise with standard deviation 0.05. The SCOUT (CFM) baseline in the main comparisons uses `cfmcanon` at $J = 2$ for evaluation; the `cfm` rows are included only as a stochastic readout audit.

Performance of the Two-Step Deterministic Readout. Table 14 shows that the two-step deterministic readout ($J = 2$) is an effective deployment strategy, achieving an equal-layout mean of 84.0 successes per 100 trials. This improves upon the one-step deterministic baseline (71.8), the equivalent stochastic Euler–Maruyama (EM) setting (69.0), and the stochastic EM setting at $J = 8$ (71.6). The gains over the one-step readout are notable on complex tasks: $+17$ on T1, $+32$ on T2, and $+13$ on T4. In comparison, the highest-performing CFM configuration tested averages 73.8. These single-checkpoint results support the deployed readout on this panel; they do not establish terminal-marginal matching or superiority across training seeds.

Appendix C derives the two-step deterministic output and its midpoint-posterior interpretation (Eqs. (34)–(36)).

Analysis of Deterministic Execution. Fix c and let $\widehat{m} = f_\theta(a_{\mathrm{ref}}, 0, c)$; denote the deterministic output of Eq. (34) by $a_{2\mathrm{step}}$. Omitting final-step noise, the matched two-step stochastic readout is:

$$S_2 = f_\theta\left(\frac{a_{\mathrm{ref}} + \widehat{m}}{2} + \frac{\sigma}{\sqrt{2}}\xi, \frac{1}{2}, c\right), \qquad \xi \sim \mathcal{N}(0, I_D). \tag{56}$$

In the Dirac limit, the relationship to conditional concentration aligns: given $\pi_c^+ = \delta_{a^*(c)}$, the ideal predictor is $f^\star(x, t, c) = a^*(c)$, causing both readouts to converge to $a^*(c)$. For a nondegenerate

Table 14: **Readout study using seed-1 actors.** Entries are successes out of 100 trials; the final column is the equal-layout mean. J denotes the readout step count. The SCOUT actors were newly trained for this study, separately from the main comparisons and the online actor ablation. Bold marks the largest mean within each block, and the highlighted row identifies our selected deployment choice: the two-step deterministic readout. These single-training-seed evaluations are distinct from the three-seed online actor comparison.

Readout Configuration	T1	T2	T3	T4	T5	Mean /100
SCOUT: ***deterministic drift (Ours)***						
$J = 1$	37	43	95	84	100	71.8
$J = 2$ **(Deployed)**	**54**	**75**	**94**	**97**	**100**	**84.0**
$J = 4$	39	51	94	84	100	73.6
$J = 8$	38	42	91	72	100	68.6
$J = 16$	37	15	96	80	100	65.6
SCOUT: *stochastic EM*						
$J = 2$	32	40	93	80	100	69.0
$J = 8$	38	48	92	80	100	**71.6**
$J = 16$	40	45	94	78	99	71.2
CFM: `cfmcanon` *configuration*						
$J = 1$	7	65	79	91	91	66.6
$J = 2$	22	54	97	97	99	**73.8**
$J = 4$	18	37	97	98	98	69.6
$J = 8$	25	35	98	99	99	71.2
$J = 16$	16	30	97	98	99	68.0
CFM: `cfm` *configuration*						
$J = 2$	23	40	95	95	98	**70.2**
$J = 8$	33	20	92	92	99	67.2

target, deterministic deployment restricts the refinement evaluation to the expected midpoint. This avoids the variance introduced by stochastically perturbing the input to a nonlinear predictor.

This perspective is supported through a conditional value analysis. Letting $V_c(a)$ denote a twice-differentiable action-value function, setting $d = a_{2\text{step}}$, and assuming $\nabla V_c(d) = 0$ alongside $\nabla^2 V_c(a) \preceq -\lambda I$ (for $\lambda > 0$) on a convex region containing d and the support of S_2, Taylor's theorem provides the bound $V_c(s) \leq V_c(d) - \lambda \|s - d\|^2/2$. Taking expectations subsequently yields the inequality:

$$V_c(d) - \mathbb{E}[V_c(S_2)] \geq \frac{\lambda}{2} \mathbb{E}\|S_2 - d\|^2. \tag{57}$$

Under the stated stationarity and strong-concavity assumptions, this inequality guarantees weakly higher value for the deterministic action, with strict inequality when $\mathbb{E}\|S_2 - d\|^2 > 0$. We have not established these assumptions for the learned policies, so the result is an illustrative explanation rather than a guarantee for our experiments.

Effect of Step Count (J). Empirically, increasing the deterministic step count beyond $J = 2$ systematically degrades the aggregate mean to 73.6, 68.6, and 65.6 for $J = 4, 8, 16$, respectively. One possible explanation is that additional predictor evaluations expose the readout to function-approximation errors. Equation (36), however, describes the output of the particular $J = 2$ discretization, not a target shared by all step counts; even an exact predictor can produce different outputs for larger J. Conversely, the $J = 2$ configuration queries the predictions exclusively at $t = 0$ and $t = 1/2$. This alignment between the derived exact two-call formulation and the empirical outcomes supports $J = 2$ as a highly effective operating point for the SCOUT architecture.

H.3 Repeatability Across SCOUT Training Runs

We consolidate the independently trained SCOUT runs reported on the five LIBERO-PRO layouts shared by the main comparisons and the ablation studies. Each layout has five reported runs: one from the main comparison (Table 8), three from the online actor ablation (Table 13), and one newly trained for the readout study (Table 14). We use only the deployed two-step deterministic readout ($J = 2$) from the latter study, so alternative readouts of the same checkpoint are not counted as additional training runs. Each result evaluates a final checkpoint after 300 online episodes on 100 trials.

Table 15: **SCOUT performance across repeated training runs.** Entries are successes out of 100 trials. A1–A3 are the three online actor-ablation runs; R is the separately trained readout-study run. Mean and sample standard deviation (SD) are computed across all five run results, including the main comparison. These are descriptive statistics across studies, which can differ in evaluation-state sets (Appendix F.4); the SD combines training and evaluation variation. The final row averages equally over the five layouts within each run column, then computes the mean and SD of those five averages.

Layout	Main	A1	A2	A3	R	Mean $\pm$ SD
T1: Cream cheese and butter into basket	66	53	51	54	54	55.6 ± 5.9
T2: Cream cheese into bowl (bottle swap)	61	73	69	72	75	70.0 ± 5.5
T3: Cream cheese into basket (milk swap)	98	98	94	92	94	95.2 ± 2.7
T4: Salad dressing into basket	97	90	97	96	97	95.4 ± 3.0
T5: Butter into basket	100	100	100	98	100	99.6 ± 0.9
Equal-layout mean	84.4	82.8	82.2	82.4	84.0	83.2 ± 1.0

Across all five reported runs per layout, SCOUT averages 83.2 successes per 100 trials over this panel, with a standard deviation of 1.0 percentage point across the five equal-layout averages. Per-layout standard deviations range from 0.9 to 5.9 percentage points. Performance remains high on T3–T5, whose means range from 95.2 to 99.6, while the more challenging T1 and T2 average 55.6 and 70.0. Within the four newly trained ablation controls alone, per-layout standard deviations are 1.0–3.4 percentage points and equal-layout means range from 82.2 to 84.0. These results support consistent performance across retraining; they do not isolate evaluation noise from training or evaluation-campaign variation.