Improving Robotic Generalist Policies via Flow Reversal Steering

Andy Tang[*,1], **William Chen**[*,2], **Andrew Wagenmaker**[2], **Chelsea Finn**[1], **Sergey Levine**[2]
Stanford University[1], UC Berkeley[2]
flow-reversal-steering.github.io

Abstract: Generalist policies can learn a wide range of skills from diverse robot datasets. In order to solve or improve on challenging news tasks, we need a way to infer and invoke the appropriate actions from the policy's rich behavioral prior, especially when directly commanding the policy fails. We focus on flow matching generalists and propose Flow Reversal Steering (FRS): a method that takes suboptimal but "reasonable" actions, finds their latent noises by passing them through the flow policy in reverse, and maps them to nearby generalist action modes. We evaluate FRS across many simulated and real-world manipulation settings. First, FRS can turn coarse semantic guidance from humans or vision-language models (VLMs) into corresponding good robot actions, improving zero-shot control. These gains can be distilled with behavioral cloning by training an auxiliary policy to output noises that the generalist maps to good actions – showing up to 95% absolute task success rate boosts in under a minute of training. Finally, FRS enables policy improvement by bootstrapping reinforcement learning with semantic knowledge, improving on several tasks that standard RL fails to improve on.

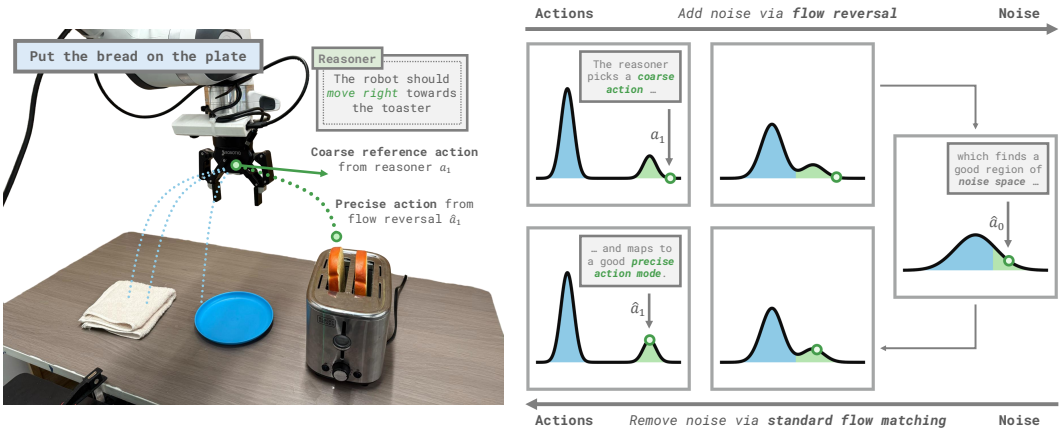


Fig. 1: Flow Reversal Steering (FRS) uses semantic reasonings from humans or VLMs to steer generalist flow policies towards sampling beneficial action modes. It does this by passing coarse but "reasonable" actions through the flow model *in reverse* to identify good regions of noise space to denoise. This enables effective task learning and adaptation via noise-space behavioral cloning and reinforcement learning.

1 Introduction

Robotic foundation models trained on large and diverse datasets provide a powerful recipe for learning multi-task generalist policies [1–5]. While such policies can often follow many commands, they will inevitably encounter new tasks diverging from their training data that require longer-horizon behaviors or demand adaptation through test-time trial-and-error. The standard recourse in such situations would be to simply add more demonstration data, retrain the generalist, and try again. However, we observe that the knowledge in these models goes beyond simply following instructions – it provides a rich prior over reasonable behaviors. For example, a policy trained to interact with bowls, sponges, and towels has many of the skills needed when learning new kitchen-cleaning tasks, like wiping countertops or cleaning dishes. Effectively invoking appropriate actions from this

prior would allow for rapid adaptation. The question then becomes: how do we best access the prior knowledge in generalist policies when faced with new tasks?

Policy steering – that is, guiding the action-sampling process to direct policy outputs to some desired end – offers a way to use the generalist's "reasonable" action prior in novel tasks by upweighting relevant behaviors. In particular, steering could allow generalists to make use of knowledge from *semantic reasoners*, such as humans and large vision-language models (VLMs). For instance, when the policy is learning the novel task of cleaning a kitchen countertop, the knowledge that "sponges are used for wiping spills" could be used to steer the policy to reach for the sponge, instead of attempting other behaviors the robot could reasonably do in the scene. We thus want a steering method that is suitable for eliciting good actions from the generalist prior, based on semantic inferences.

Many steering methods use diffusion or flow matching [6–13], a common parameterization for generalist policies. Notably, flow matching policies learn a *deterministic* map from noise to action [14] that can be steered by finding the noise values that map to desirable actions within the generalist prior [15]. However, the noise space of flow policies lacks immediately-apparent structure, so past works resort to expensive trial-and-error via reinforcement learning to find good noise values. To effectively steer generalist flow policies, we would instead ideally utilize semantic reasoning to quickly identify noise values that map to semantically-appropriate actions.

We thus propose **Flow Reversal Steering (FRS)**: a novel approach that maps coarse reference actions to their noises by passing them through flow policies *in reverse*. When denoised, this yields actions that are fine-grained and "in-distribution" for the generalist, while staying roughly consistent with the reference action. Given even a rough sketch of robot behaviors (e.g., the general direction needed to reach for a target object), FRS can "project" that behavior into the generalist's prior to produce a similar fine-grained action.

This mechanism is especially useful for semantic reasoners, such as VLMs, that can roughly infer appropriate robot behaviors, but cannot ground them into dexterous low-level actions. FRS moves the onus of emitting robot actions to the generalist, while reasoners can focus on broad, high-level steering. As FRS also gives corresponding noise vectors, it meshes well with latent-noise policies, which steer flow generalists by changing their distribution of input noises. This can both (1) enable fast and efficient adaptation via noise-space behavioral cloning (BC) and (2) bootstrap noise-space reinforcement learning (RL) for tasks where exploring via the generalist policy is intractable.

We evaluate FRS on state-of-the-art generalist vision-language-action policies (VLAs [16]) through extensive simulated and real-world manipulation experiments [17, 18]. We show how FRS allows humans and VLMs to effectively guide generalists across diverse tasks, even by simply specifying just the rough direction the robot should move. Then, we show how the noises from FRS can be used for robustly learning tasks in just one minute of active BC training on 10 trajectories. Lastly, we find FRS speeds up noise policy RL, where using just one FRS success as prior data enables efficient improvement on tasks that the base VLA nearly always fails at.

2 Related Works

Foundation models for robotics. Large pretrained foundation models [19] can be applied in robotics in two ways: by fine-tuning them on robotic data *or* by querying them zero-shot. The former can involve training them with BC on large-scale demonstration datasets [18, 20, 21], with two popular approaches fine-tuning (1) VLMs into vision-language-action models [1, 2, 22] or (2) video generators into world-action models [4, 5]. These policies are exposed to many robotic skills during pretraining, letting them follow many user commands. Vitally, this also captures a prior distribution over "reasonable" behaviors, often including behaviors needed to solve novel tasks the policies initially fail at. FRS aims to flexibly elicit behaviors from this prior that are semantically appropriate for novel tasks, which in turn can be used for rapid adaptation and policy improvement. VLMs can also be trained into reward models for RL [23–26]. This is orthogonal and complementary to our work, as using FRS for RL places no restrictions on the chosen reward function.

Both actions and rewards can also be predicted by foundation models *without* training on robot data. VLMs can both invoke robot behaviors via predefined interfaces [27–38] or act as reward functions [39–44]. However, such methods are limited by VLMs' capabilities. While VLMs can effectively compose a limited set of high-level behavioral primitives, they struggle when given lower-level ones, thus giving a tradeoff between performance and flexibility [32]. VLMs also struggle with reward modeling due to their limited fine-grained visual reasoning skills [43]. Our method avoids this issue by relying on the generalist to produce actions. Instead, VLMs can make the high-level semantic inferences they excel at, while FRS grounds this coarse feedback into appropriate actions.

Diffusion policies and steering. Many generalist policies use diffusion or flow matching [3, 4, 12–14, 16, 22, 45], which iteratively denoises Gaussian noise vectors to generate action samples. They can be steered by either modifying this denoising process or changing the partially-noised data inputs that get denoised [6–11, 46]. Notably, flow matching *deterministically* maps noise to outputs, so it can be steered by finding pure noises that denoise to good actions. However, finding good noises is challenging – they are usually identified by trial-and-error with RL [15]. In contrast, we propose combining flow reversal with coarse feedback to quickly identify effective noise. We show how flow reversal enables efficiently training policies that emit noises to steer the generalist toward solving new tasks, *while obviating or accelerating the tedious process of discovering good noises via RL.*

Improving generalist policies. Past methods for improving generalists (like VLAs) with RL often have certain traits. When tuning the full VLA, RL methods tend to use supervised learning – like distillation – for policy extraction [46–49], avoiding needing action probabilities (which are hard to extract from flow models). They are often batched online, as their size makes true online RL unwieldy. To avoid this issue, other works train separate smaller policies with online RL [15, 47, 49], often using the base policy prior to constrain behavior to "reasonable" actions, e.g., via residual RL [49, 50], behavior attenuation with classifier-free guidance [11, 46], or treating the VLA as a latent action decoder [15]. These policies can then be distilled into the generalist [47, 49]. Our method uses semantic feedback to elicit "reasonable" behaviors during generalist policy improvement, yielding gains beyond existing work which solely use the base policy as a behavior constraint.

Flow and diffusion reversal in robotics. While flow or diffusion reversal is common in vision (App. A), fewer works use such techniques in robotics. GenPO [51] inverts actions from a diffusion policy to estimate their likelihood for RL updates. Concurrent to our work, UniSteer [52] inverts human actions via flow reversal to obtain "good" noises, which are used for noise-space RL by adding a behavioral cloning (BC) term. While similar, our approach diverges from these works in three key ways: (1) we use flow reversal to steer generalist flow policies by *refining* coarse actions, admitting guidance from humans *and* scalable VLMs; (2) we show how, even without RL, flow reversal enables efficiently training noise policies with *just* BC; and (3) we show that flow reversal can bootstrap RL with coarse non-human guidance, even if the base policy nearly never succeeds.

3 Preliminaries

Flow matching VLAs. Generalist policies take a pretrained backbone and fine-tunes it with behavioral cloning (BC) to match the action distribution of experts. Given BC data (o, a_1) and sampled times $t \in [0, 1]$, these policies fit a velocity field $v_\theta(a_t, t \mid o)$ that maps noise $a_0 \sim \mathcal{N}(0, I)$ to actions a_1 by denoising partially-noised actions produced by forward diffusion $a_t = t \cdot a_1 + (1-t) \cdot a_0$. The flow v_θ satisfies an ordinary differential equation by minimizing the loss $\mathcal{L}_\theta$:

$$\mathcal{L}_\theta = \mathbb{E}_{o,a_1,a_0,t}\left[||v_\theta(a_t, t \mid o) - (a_1 - a_0)||^2\right] \iff \mathrm{d}a_t = v_\theta(a_t, t \mid o)\,\mathrm{d}t$$

a_1 can be practically solved for via Euler integration with time steps h (with $1/h$ iterations):

$$a_{t+h} \leftarrow a_t + v_\theta(a_t, t \mid o) \cdot h, \qquad \text{for } t \in \{0, h, \ldots, 1-h\}, \quad a_0 \sim \mathcal{N}(0, I)$$

We denote this denoising process as $a_1 \leftarrow \mu_\theta(a_0, o)$. This defines the BC policy, which samples actions by drawing $a_0 \sim \mathcal{N}(0, I)$ and denoising it with μ_θ into a_1. We specifically consider flow matching VLAs [16, 22, 45], though in principle, our method is applicable to other flow matching generalist architectures or diffusion policies with deterministic DDIM sampling too [13, 53].

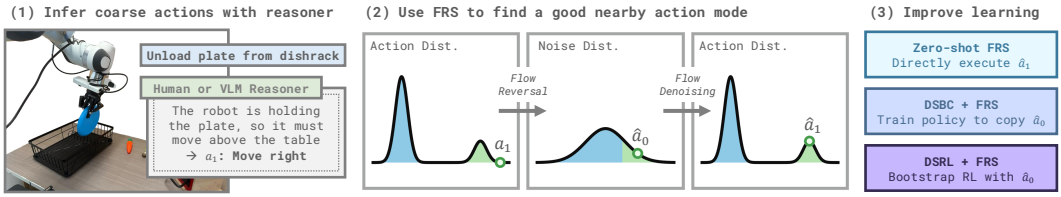


Fig. 2: Overview of FRS. **(1)** A human or VLM semantically reasons about the novel task to determine a reference action capturing roughly what the robot should do. **(2)** This coarse action is passed through flow reversal and denoising, projecting it into the space of generalist actions. **(3)** Both the expert noises and actions can be used for policy improvement by executing the action (zero-shot FRS, Sec. 5.2) or training a noise action policy with supervised learning (DSBC, Sec. 5.3) or reinforcement learning (DSRL + FRS, Sec. 5.4).

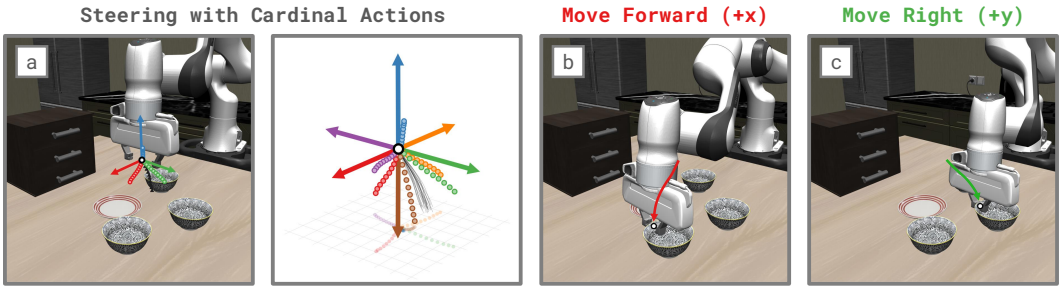


Fig. 3: Illustrative examples of FRS with $\pi_{0.5}$ in LIBERO. **(a)** Solid arrows are directional reference actions, dots are corresponding steered actions, and black represents samples from the base policy without steering. The outputs of FRS are biased towards "reasonable" behaviors given the scene, e.g., reaching for the bowls. **(b)** and **(c)** show rolling out the forward (red) and right (green) steered actions, followed by executing the base policy.

Diffusion steering. Flow policies' outputs can be controlled by finding noises that map to desirable actions, rather than drawing from $\mathcal{N}(0, I)$. One way is to partially noise a reference action, then denoise it with the BC policy μ_θ to produce a similar one [9, 10]. This fails when fully noising, as doing so removes all information about the reference. Good noises can also be found via RL; Diffusion Steering via Reinforcement Learning (DSRL [15]) learns an auxiliary *latent noise action policy* $\pi_\phi^{\text{noise}}(a_0 \mid o)$ by treating the noise a_0 as an action and running RL in a "noise action" Markov decision process. It learns a noise critic $Q^{\text{noise}}(o, a_0)$, and trains the actor $\pi_\phi^{\text{noise}}(a_0 \mid o)$ to maximize it:

$$\pi_\phi^{\text{noise}} \leftarrow \arg\max_\pi \mathbb{E}_{o\sim\mathfrak{B}, a_0\sim\pi(\cdot|o)}[Q^{\text{noise}}(o, a_0)]$$

for replay buffer $\mathfrak{B}$. At deployment, DSRL samples $a_0 \sim \pi_\phi^{\text{noise}}(\cdot \mid o)$, then computes and executes $a_1 \leftarrow \mu_\theta(a_0, o)$. π_ϕ^{noise} thus finds noises that the BC policy μ_θ denoises to good robot actions.

4 Flow Reversal Steering (FRS)

Our method, Flow Reversal Steering (FRS), takes "coarse" actions and refines them using a generalist policy into similar, higher-quality actions. In Sec. 4.1, we show how flow reversal can identify noises that bias the flow matching policy into sampling actions *of the same mode* as the coarse one. Then, in Sec. 4.2, we consider how humans or VLMs can provide such coarse guidance to the robot, based on their semantic knowledge. Flow reversal converts these rough sketches of robot behaviors into in-distribution actions. Finally, in Sec. 4.3, we present ways to use these semantically-guided trajectories to efficiently learn and improve at new tasks. See Fig. 2 for an illustrative overview.

4.1 Reversing the Flow Velocity Field

Since flow denoising is deterministic for a given input, flow can be *reversed* to determine the noise a_0 that corresponds to a given reference action a_1 by simply integrating the ODE it defines backwards in time. That is, using Euler integration:

$$a_{t-h} \leftarrow a_t - v_\theta(a_t, t \mid o) \cdot h, \quad \text{for } t \in \{1, 1-h, \ldots, h\}$$

We denote this *noising* process as $\hat{a}_0 \leftarrow \mu_\theta^{-1}(a_1, o)$, as it inverts μ_θ, where the hat denotes a *computed* noise, not a sampled one. This does not modify the learned model v_θ at all – flow reversal

needs the same computations as standard flow denoising, just starting from an action instead of noise and swapping the order of integration. In turn, $\hat{a}_0$ can be passed through standard flow denoising to yield $\hat{a}_1 \leftarrow \mu_\theta(\hat{a}_0, o)$. As $h \to 0$, this exactly reconstructs the reference $\hat{a}_1 = a_1$. However, for finite iterations, integration error means that $\hat{a}_1$ only approximately reconstructs a_1.

Empirically, we find that this results in reconstructed actions $\hat{a}_1$ being *similar but not identical to* the reference a_1, while also being "in-distribution" for BC policy π_θ. This yields actions from the generalist that are *biased* towards reference behaviors, rather than perfectly reconstructing them. We call this **Flow Reversal Steering (FRS)**.

We show an illustrative example of FRS with an actual VLA in Fig. 3 (see App. B and Fig. 12 for more), using ten noising/denoising steps ($h = 0.1$). The steered actions follow the same general direction as their respective coarse reference, albeit biased by the affordances in the scene – e.g., when the gripper is above the table and empty, the steered actions tend to move down towards objects to grasp; when the gripper is holding something, it tend to move up to lift the object *or* towards containers to place it. These are the "reasonable" actions internalized by the generalist policy.

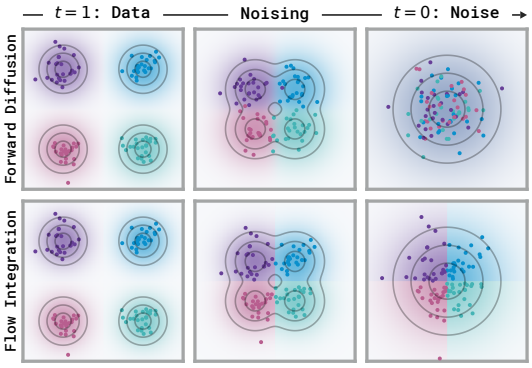


Fig. 4: Noising via the forward diffusion process vs. reverse flow integration. Both have the same marginals, but the former uses noise interpolation (so all signal is gone by $t = 0$), while the latter deterministically maps from data to noise and back.

Note that flow reversal is *distinct* from the forward diffusion process, which linearly interpolates data with Gaussian noise $a_t = t \cdot a_1 + (1 - t) \cdot a_0;\ a_0 \sim \mathcal{N}(0, I)$. While this is used for training flow models, it rapidly destroys the information in the output, whereas flow reversal identifies the noise which deterministically maps to the reference action (Fig. 4). Past works propose using forward diffusion for steering by partially noising reference actions, then passing them through denoising [9, 10]. However, we find these methods to be highly sensitive to how much noise is added, and thus hard to tune and ineffective (Sec. 5.2).

4.2 High-Level Semantic Reasoning for Steering

We now consider sources of semantically-reasonable reference actions as inputs to FRS. Naturally, humans can guide robots to solve tasks, though standard methods, like teleoperation, are costly and tedious (though can also be used with flow reversal, see Sec. 4.3). Similarly, VLMs can roughly identify appropriate robot high-level behaviors [36, 37], even if they cannot emit fine robot actions. Thus, we need a way for these *reasoners* to tap into their semantic knowledge and easily ground it in coarse actions for guiding the robot.

When running FRS online, we opt to have both human and VLM reasoners emit simple *directional* actions to guide the robot. That is, the reasoner can choose Cartesian directions based on how they think the manipulator's end effector should move (see App. D). This is programmatically turned into a rough steering action chunk [54] that servos the robot straight in the specified direction. Unsurprisingly, such action chunks are ineffective when directly executed (Sec. 5.2), but are nonetheless suitable as reference actions a_1 for steering. Finally, both reasoners also have the option to defer to the base policy when steering is inappropriate, e.g., when executing precise grasps.

The online FRS inference loop thus involves (1) querying the human or VLM reasoner at each step to infer the general direction the robot should move; (2) converting that motion into a corresponding directional reference action a_1; (3) using flow reversal to map it back to noise $\hat{a}_0 \leftarrow \mu_\theta^{-1}(a_1, o)$; and (4) denoising it back into an action to execute $\hat{a}_1 \leftarrow \mu_\theta(\hat{a}_0, o)$. This steers the generalist's action generation based on what the high-level reasoner infers to be useful for the task.

4.3 Improving Generalist Policies with FRS

Now that the reasoner can guide the robot toward semantically-sensible behaviors, how can this be used to improve performance? We propose three paradigms for using FRS to improve policies.

Zero-shot online steering. The simplest way is to use FRS **zero-shot**, having the reasoner steer the policy every step. That is, each time the generalist policy would be queried during deployment, the human or VLM reasoner is queried to produce a semantically-meaningful coarse reference action, which is passed through flow reversal and denoising before being executed.

However, constantly querying the reasoner can become expensive (especially for human reasoners), so we also aim to use FRS's elicited trajectories for learning. While these rollouts can work with any policy learning algorithm, it synergizes especially well with *noise policy* learning (Sec. 3). FRS yields noises immediately aligned with the coarse inferences of semantic reasoners, alleviating the usual difficulty of finding good noises with random trial-and-error when training noise policies via RL. This advantage enables two novel methods for efficient generalist policy learning with FRS.

Supervised learning on FRS noise actions. We can treat flow reversal noises as "expert" noise actions for supervised learning, rather than RL. Given observation-noise pairs $(o, \hat{a}_0)$ run BC with:

$$\pi_\phi^{\text{noise}} \leftarrow \arg\max_\pi \mathbb{E}_{o,a_0}[\log \pi(a_0 \mid o)],$$

thereby distilling FRS's good noise actions into $\pi_\phi^{\text{noise}}(\hat{a}_0 \mid o)$. At test time, it is treated exactly like a DSRL noise policy, inferring noises $\hat{a}_0$ at each step that get mapped to actions by the generalist $\hat{a}_1 \leftarrow \mu_\theta(\hat{a}_0, o)$. Naturally, we call this **Diffusion Steering via Behavioral Cloning (DSBC)**.

There are two ways to acquire the noises for supervising DSBC. First, we can use noises from **online** rollouts of zero-shot FRS collected as described above. As these noises have been denoised and executed, they are verified as mapping to good actions if they lead to task success. Second, DSBC can also be applied to *existing* robotic demonstrations. Given an observation o and corresponding demonstrator action a_1, flow reversal can augment each frame with noise $\hat{a}_0 \leftarrow \mu_\theta^{-1}(a_1, o)$ approximately mapping to a_1, providing entirely **offline** data for DSBC. The lack of online execution yields a practical trade-off: as flow reversal does not perfectly reconstruct reference actions, offline DSBC does not ensure that reconstructed actions are free from suboptimality or error. However, this also permits precise reference actions from *any* suitable teleoperation or control interface.

Bootstrapping RL with FRS. When used *in conjunction with* DSRL, FRS's noise trajectories can be used as prior data and behavior regularization via two simple changes. First, we augment the policy learning loss by adding an auxiliary DSBC loss over successful FRS rollouts, $\mathfrak{D}^+$. That is, rather than simply training π_ϕ^{noise} to maximize Q^{noise} as standard DSRL does, we train π_ϕ^{noise} via:

$$\pi_\phi^{\text{noise}} \leftarrow \arg\max_\pi \underbrace{\mathbb{E}_{o\sim\mathfrak{B},\, a_0\sim\pi(\cdot|o)}\left[Q^{\text{noise}}(o, a_0)\right]}_{\text{RL objective}} + \lambda \underbrace{\mathbb{E}_{(o,a_0)\sim\mathfrak{D}^+}[\log \pi(a_0 \mid o)]}_{\text{BC auxiliary objective}}$$

Second, we prefill DSRL's buffer $\mathfrak{B}$ with FRS trajectories (optionally including failed ones). This enables improvement from experience beyond zero-shot FRS and DSBC. We call this **DSRL + FRS**.

These changes improve the efficiency of RL by encouraging the noise policy to explore around FRS's semantically-meaningful behaviors, contrasting the random noise sampling early in DSRL. Furthermore, since robot trajectories do not usually have noise actions, training noise policies with prior data is usually challenging or expensive and requires, for example, distilling robot action space Q-functions into noise action space [15]. Flow reversal circumvents this by rapidly and cheaply identifying good underlying noises from reference actions, including from offline data.

5 Experiments

We now evaluate Flow Reversal Steering by answering: (1) Can FRS improve performance without any training by having VLMs guide generalists towards semantically-reasonable behaviors? (2) Can we use the improved trajectories from FRS to rapidly learn new tasks? (3) Can FRS help generalists more efficiently *improve* from experience? See App. E and App. F for more experimental details.

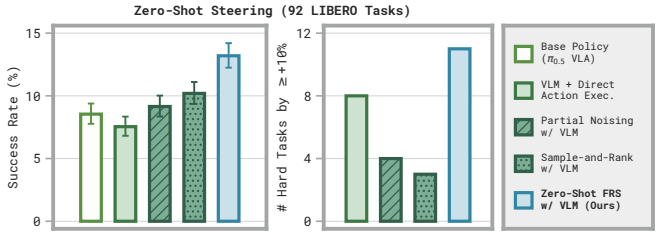


Fig. 5: **Left:** Zero-shot FRS improves over the base VLA across LIBERO tasks by converting coarse VLM actions into precise robot actions. **Middle:** Number of tasks improved by $\geq$10% (where the base VLA gets $\leq$2%). Zero-shot FRS improves the most, yielding the best learning signal for RL. Directly executing VLM actions improves several, but lowers performance overall.

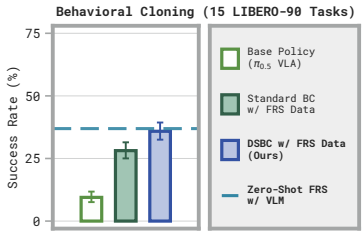


Fig. 6: FRS trajectories can be distilled via Diffusion Steering via Behavioral Cloning (DSBC). This matches the performance of zero-shot VLM steering, while being more effective than standard BC on the same data.

5.1 Experimental Setup

Simulation. We use LIBERO [17] for scalable simulated evaluations. Our zero-shot results consider the full Spatial, Object, and Goal splits, as well as all 62 tasks in 90 that our base policy achieves $\leq$ 40% success on (Sec. 5.2). We then use a 15-task subset of LIBERO-90 where FRS achieves sufficient success to train DSBC policies (Sec. 5.3). Finally, we run DSRL + FRS on that subset, as well as a harder 10-task subset where the base policy nearly completely fails (Sec. 5.4). To allow room for improvement, we use base VLAs that have *not* been trained on the LIBERO splits that we run them on, following Wagenmaker et al. [15]. For LIBERO-90, we use OpenPi's $\pi_{0.5}$-LIBERO [16], which is trained on all splits *except* 90. For all others, we use $\pi_{0.5}$ fine-tuned by Jain et al. [55], trained *solely* on 90 [56]. We focus on VLM steering to accommodate LIBERO's scale.

Real world. We also aim to validate FRS's effectiveness on real robots. We use the DROID setup to evaluate FRS in the real world [18], with $\pi_{0.5}$-DROID as our base flow VLA for steering [16, 57]. As DROID's primary challenge comes from its diversity, we choose a set of six task that require interacting with objects in scenes that admit many possible reasonable behaviors.

5.2 FRS Boosts Zero-Shot Performance on Challenging Manipulation Tasks

We first test if using FRS to refine VLMs' semantic guidance can boost zero-shot performance.

Comparisons. We compare FRS with several baselines. First, we run the **base policy** without steering, to confirm that applying FRS to that same policy improves performance. Second, to show that VLM actions alone are too coarse, we **directly execute the VLM's actions**, as is done in zero-shot control methods [36, 38]. Last, to show that FRS is good for steering, we compare with prior policy steering methods: **partial noising**, where reference actions are interpolated with Gaussian noise before being denoised, providing biased initializations for sampling [9, 10] (Sec. 4.1); and **sample-and-rank** [10, 58–61], where the policy samples multiple actions in parallel, ranks them post-hoc with a scoring function (in our case, cosine similarity with the VLM reference action), then executes the best one. For each split, all approaches use the same flow VLA as the base policy, the same VLM system prompt, and the same Gemini-ER-1.6 VLM [62]. See App. E.1 and App. D.2.

Results. As shown in Fig. 5, FRS outperforms the base policy. Critically, in 11 of the 42 LIBERO tasks where the base policy gets $\leq$ 2% (0 or 1 success out of 50 attempts), our method yields a substantial absolute success increase of at least 10%. While the base VLA may struggle to stumble upon even a single success, FRS allows VLMs to steer the policy towards meaningful behaviors, thereby providing much earlier rewards – and thus, beneficial learning signals – for RL (Sec. 5.4).

We also find directly executing VLM actions is ineffective. This both supports the intuition that VLMs struggle with outputting precise low-level actions zero-shot and also shows how FRS is not simply reconstructing the VLM actions, but using them to steer towards better – yet still semantically-similar – fine-grained actions from the VLA. Finally, not only are partial noising and sample-and-rank less performant than FRS, they only boost 4 and 3 hard tasks, respectively. These baselines tend to work well when the VLA already has high probability on good behaviors, not on hard tasks where success is rare, while FRS is able to still learn in this case.

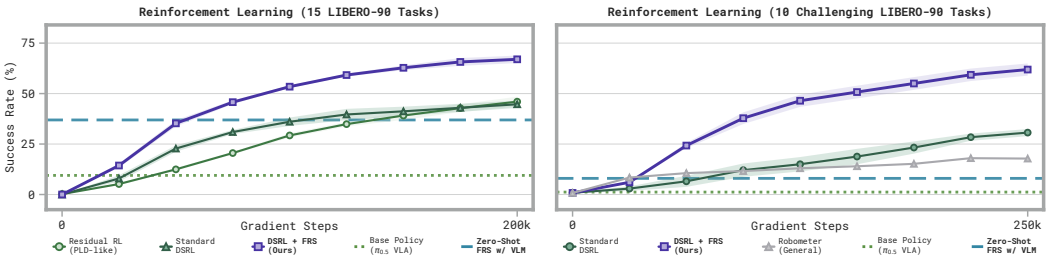


Fig. 7: **Left:** DSRL + FRS improves upon standard generalist RL methods [15, 49]. **Right:** Even if FRS struggles, warmstarting with even one FRS success improves RL on tasks where the base policy gets near-zero.

5.3 FRS Enables Diffusion Steering via Behavior Cloning

We now show how good trajectories from FRS yield expert noise actions, which can be distilled via DSBC. We focus on online DSBC here, and present offline DSBC LIBERO results in App. E.2.2.

Comparisons. We start by considering online DSBC on zero-shot FRS data. Alongside the **base policy** and **zero-shot VLM FRS** as baselines, we also compare against running **standard BC** on the FRS successful trajectories (using the same small architecture as the DSBC noise policy). We note that FRS's successful rollouts can be distilled back into the full VLA as well, but doing so uses much more compute than training an auxiliary policy. See App. E.2.

Results. As shown in Fig. 6, DSBC distills the zero-shot gains of FRS, improving over the base VLA. One empirical benefit of DSBC is that when the noise policy makes mistakes and enters out-of-distribution states, it is often able to recover. We posit that, while the noise policy's actions may be bad at these OOD states, the VLA treats those noises akin to its noise prior, mapping them to "reasonable" in-distribution actions. Essentially, *the DSBC noise policy is implicitly robust against compounding error, as it "falls back" to the VLA's behavioral prior in unfamiliar states*. As DSBC can rely on the VLA's action prior, it is better to run BC on noise actions than regular actions. In LIBERO, DSBC outperforms standard BC (though, as LIBERO has little randomization, memorizing a small dataset can still work [63, 64]). This difference is more salient in the real world, where *standard BC with a small flow policy completely fails* (Fig. 8).

DSBC is also sample-, compute-, and time-efficient. In LIBERO, it trains on only 18 rollouts per task on average. On real robots, it needs just 10 rollouts per task to achieve high performance (Sec. 5.5). The policy is likewise small – in total, training takes around 1 GB of GPU memory (as the VLA does not need to be loaded during training), whereas fine-tuning a full VLA requires hundreds of GBs. Finally, due to the model and data size, DSBC policies take under a minute to train.

5.4 FRS Accelerates and Improves Reinforcement Learning

We finally aim to show how FRS can be used with RL to learn from experience. This allows it to surpass both the fixed performance of zero-shot FRS and distilling FRS's data with DSBC.

Settings and comparisons. We run RL in two LIBERO-90 settings. First, we consider the 15 tasks from Sec. 5.3 where zero-shot VLM FRS is especially effective (yielding $\geq$10% improvement), and run DSRL + FRS by selecting 20 random FRS rollouts to prefill the replay buffer (in place of some initial prefill rollouts). Our baselines are thus two standard VLA RL methods that do *not* use FRS data: (1) **standard DSRL** [15] and (2) **residual RL** (akin to PLD [49]). Second, we consider 10 harder LIBERO-90 tasks where the base VLA nearly always fails *and* zero-shot VLM FRS achieves only 8%. This tests if FRS is useful for RL, *even if steering rarely succeeds*. We thus run DSRL + FRS with only one successful steered trajectory, which can take upwards of 50 trials, given the tasks' difficulty. As densifying rewards is another way to guide RL with VLMs, we also compare against using **RoboMeter** as a reward model. All methods run RL on a small policy to steer a VLA, albeit in different ways. We thus control for the VLA, the small policy's architecture, and the underlying RL algorithm (SAC). See App. E.3.

Results. DSRL + FRS is the most effective and sample-efficient RL method in both our LIBERO settings. As shown in Fig. 7 (left), running RL with FRS rollouts as prior data yields significant gains over standard RL, enabling both faster learning and higher final success rate across 15 tasks.

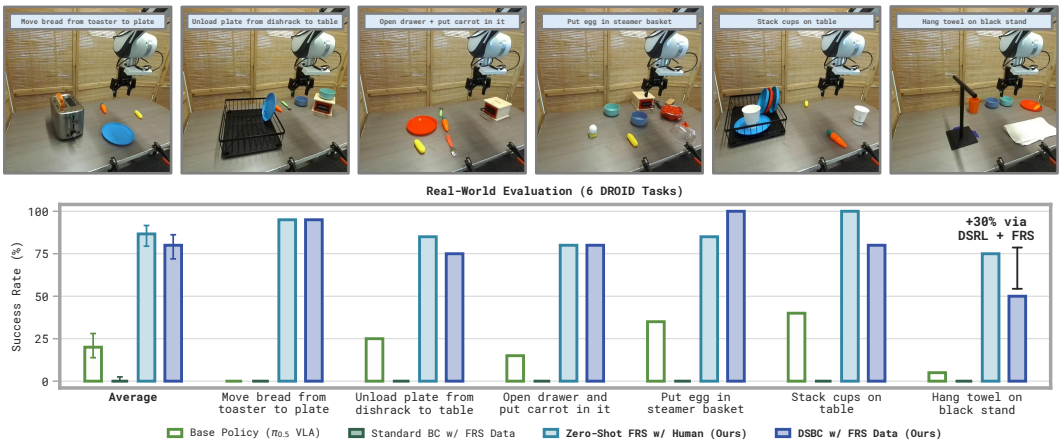


Fig. 8: DSBC boosts performance on real-world tasks when trained with just 10 FRS rollouts, while standard BC *completely fails* in this data regime. DSBC rollouts can also bootstrap RL, as in the towel hanging task.

For our second setting, where the base VLA has success rate near 0%, DSRL + FRS again enables effective improvement as Fig. 7 (right) shows. Naive DSRL struggles to learn – only reaching a final success rate of around 30%, likely due to the poor performance of the base policy. By leveraging VLM FRS to direct the learner to successful behaviors in early stages of learning, DSRL + FRS is able to overcome this, quickly improving and converging to a significantly higher final success rate.

5.5 FRS is Practical and Effective for Real-World Manipulation

Lastly, we show FRS is effective in real-world generalist manipulation (Fig. 8). As steering offloads the effort of producing fine actions to the policy, FRS lets humans solve tasks while only giving very crude feedback (i.e., one Cartesian directional action per action chunk), compared to the dense supervision provided during teleoperation. The successful human-steered trajectories can then be used for DSBC. Across six tasks that the base VLA struggles on, we find that corresponding DSBC policies boost average absolute performance by 60% by training on just 10 successful human FRS rollouts per task. Equivalent *standard* BC flow policies trained on the FRS robot actions completely fail to learn these tasks in this data regime, as they cannot inherently rely on the VLA action prior in unfamiliar situations (see Sec. 5.3). As in the LIBERO DSBC experiments, each training run takes under a minute and requires around 1 GB of GPU memory. Lastly, we show how DSBC can be used to bootstrap trajectories for simple online RL, as we demonstrate on the challenging towel-hanging task by boosting performance from 5% base, to 50% via DSBC, and then to 80% post-RL (App. F).

Real-world offline DSBC. Offline DSBC can use *standard* robotic trajectories (i.e., with only robot actions saved, and no noises), enabling learning with more optimal demonstrations than what is possible through a coarse steering interface. We test this with a real-world task that naïve directional steering struggles on due to imprecision. We collect 20 episodes of the task "hang the tape on the stand" via regular teleoperation (i.e., without noises). Then, we use $\pi_{0.5}$-DROID flow reversal to augment all episodes' actions with their corresponding noises, which DSBC learns from. This noise outperforms the base VLA and standard BC, which struggles to learn precise, temporally-coherent behaviors in our low-data regime (Fig. 9). This validates flow reversal as a simple way for noise policies to make use of standard offline robot data *without* noises.

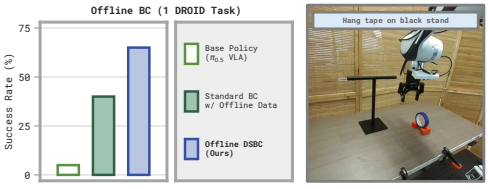


Fig. 9: Offline DSBC enables noise policy learning from *standard* robot data.

6 Discussion

We introduce Flow Reversal Steering (FRS), a way to convert coarse semantic guidance into precise actions by reversing flow generalist policies. Through extensive simulated and real-world tasks with state-of-the-art VLAs, we show how FRS allows reasoners, like humans and VLMs, to guide

policies towards reasonable behaviors for novel tasks. This enables rapid policy learning through our novel Diffusion Steering via Behavior Cloning (DSBC) method or by bootstrapping DSRL. While we showed FRS's effectiveness, some parts are limited. We hope that FRS provides an alternative paradigm for efficiently improving generalist policies, where learning is not only guided by optimizing reward functions, but task-relevant semantic knowledge as well.

Acknowledgments

We would like to thank Qiyang Li, Seohong Park, Charles Xu, Jubayer Ibn Hamid, Alex Swerdlow, Lars Ankile, Arhan Jain, Jenny Pan, Ayush Agrawal, Jesse Zhang, and the other members of RAIL and IRIS Labs for insightful discussions and help with experiments. This research was partly supported by ONR N00014-25-1-2060, ARL DCIST CRA W911NF-17-2-0181, and DARPA TIAMAT. We would like to thank the NVIDIA Academic Grant Program for providing compute resources. This work used GPUs at NCSA Delta and Purdue Anvil through allocation CIS260400 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296 [65].

References

[1] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. 2024.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[3] TRI LBM Team, J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina, N. Kuppuswamy, K.-H. Lee, K. Liu, D. McConachie, I. McMahon, H. Nishimura, C. Phillips-Grafflin, C. Richter, P. Shah, K. Srinivasan, B. Wulfe, C. Xu, M. Zhang, A. Alspach, M. Angeles, K. Arora, V. C. Guizilini, A. Castro, D. Chen, T.-S. Chu, S. Creasey, S. Curtis, R. Denitto, E. Dixon, E. Dusel, M. Ferreira, A. Goncalves, G. Gould, D. Guoy, S. Gupta, X. Han, K. Hatch, B. Hathaway, A. Henry, H. Hochsztein, P. Horgan, S. Iwase, D. Jackson, S. Karamcheti, S. Keh, J. Masterjohn, J. Mercat, P. Miller, P. Mitiguy, T. Nguyen, J. Nimmer, Y. Noguchi, R. Ong, A. Onol, O. Pfannenstiehl, R. Poyner, L. P. M. Rocha, G. Richardson, C. Rodriguez, D. Seale, M. Sherman, M. Smith-Jones, D. Tago, P. Tokmakov, M. Tran, B. V. Hoorick, I. Vasiljevic, S. Zakharov, M. Zolotas, R. Ambrus, K. Fetzer-Borelli, B. Burchfiel, H. Kress-Gazit, S. Feng, S. Ford, and R. Tedrake. A careful examination of large behavior models for multitask dexterous manipulation, 2025. URL https://arxiv.org/abs/2507.05331.

[4] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, G. Kurian, S. Indupuru, Y. L. Tan, C. Zhu, J. Xiang, A. Malik, K. Lee, W. Liang, N. Ranawaka, J. Gu, Y. Xu, G. Wang, F. Hu, A. Narayan, J. Bjorck, J. Wang, G. Kim, D. Niu, R. Zheng, Y. Xie, J. Wu, Q. Wang, R. Julian, D. Xu, Y. Du, Y. Chebotar, S. Reed, J. Kautz, Y. Zhu, L. J. Fan, and J. Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

[5] J. Pai, L. Achenbach, V. Montesinos, B. Forrai, O. Mees, and E. Nava. mimic-video: Video-action models for generalizable robot control beyond vlas, 2025. URL https://arxiv.org/abs/2512.15692.

[6] J. Ho and T. Salimans. Classifier-free diffusion guidance, 2022.

[7] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

[8] R. Singhal, Z. Horvitz, R. Teehan, M. Ren, Z. Yu, K. McKeown, and R. Ranganath. A general framework for inference-time scaling and steering of diffusion models, 2025. URL https://arxiv.org/abs/2501.06848.

[9] T. Yoneda, L. Sun, G. Yang, B. Stadie, and M. Walter. To the noise and back: Diffusion for shared autonomy, 2025. URL https://arxiv.org/abs/2302.12244.

[10] Y. Wang, L. Wang, Y. Du, B. Sundaralingam, X. Yang, Y.-W. Chao, C. Perez-D'Arpino, D. Fox, and J. Shah. Inference-time policy steering through human interactions, 2024.

[11] K. Frans, S. Park, P. Abbeel, and S. Levine. Diffusion guidance is a controllable policy improvement operator, 2025. URL https://arxiv.org/abs/2505.23458.

[12] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[13] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

[14] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

[15] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

[16] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

[17] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

[18] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[19] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech,

E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang. On the opportunities and risks of foundation models, 2022.

[20] Embodiment Collaboration, A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2024.

[21] T. Jiang, T. Yuan, Y. Liu, C. Lu, J. Cui, X. Liu, S. Cheng, J. Gao, H. Xu, and H. Zhao. Galaxea open-world dataset and g0 dual-system vla model, 2025. URL https://arxiv.org/abs/2509.00576.

[22] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[23] A. Liang, Y. Korkmaz, J. Zhang, M. Hwang, A. Anwar, S. Kaushik, A. Shah, A. S. Huang, L. Zettlemoyer, D. Fox, Y. Xiang, A. Li, A. Bobu, A. Gupta, S. Tu, E. Biyik, and J. Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons, 2026. URL https://arxiv.org/abs/2603.02115.

[24] T. Lee, A. Wagenmaker, K. Pertsch, P. Liang, S. Levine, and C. Finn. Roboreward: General-purpose vision-language reward models for robotics, 2026. URL https://arxiv.org/abs/2601.00675.

[25] S. A. Sontakke, J. Zhang, S. M. R. Arnold, K. Pertsch, E. Bıyık, D. Sadigh, C. Finn, and L. Itti. Roboclip: One demonstration is enough to learn robot policies, 2023.

[26] S. Zhai, Q. Zhang, T. Zhang, F. Huang, H. Zhang, M. Zhou, S. Zhang, L. Liu, S. Lin, and J. Pang. A vision-language-action-critic model for robotic real-world reinforcement learning, 2025. URL https://arxiv.org/abs/2509.15937.

[27] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control, 2023. URL https://arxiv.org/abs/2209.07753.

[28] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models, 2022.

[29] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition, 2023.

[30] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor. Chatgpt for robotics: Design principles and model abilities. Technical report, Microsoft, 2023.

[31] J. Shi, R. Yang, K. Chao, B. S. Wan, Y. S. Shao, J. Lei, J. Qian, L. Le, P. Chaudhari, K. Daniilidis, et al. Maestro: Orchestrating robotics modules with vision-language models for zero-shot generalist robots, 2025.

[32] M. Fu, J. Yu, K. El-Refai, E. Kou, H. Xue, H. Huang, W. Xiao, G. Wang, F.-F. Li, G. Shi, J. Wu, S. Sastry, Y. Zhu, K. Goldberg, and L. J. Fan. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation, 2026. URL https://arxiv.org/abs/2603.22435.

[33] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023. URL https://arxiv.org/abs/2307.05973.

[34] N. Kumar, W. Shen, F. Ramos, D. Fox, T. Lozano-Pérez, L. P. Kaelbling, and C. R. Garrett. Open-world task and motion planning via vision-language model generated constraints, 2026. URL https://arxiv.org/abs/2411.08253.

[35] W. Shen, N. Kumar, S. Chintalapudi, J. Wang, C. Watson, E. Hu, J. Cao, D. Jayaraman, L. P. Kaelbling, and T. Lozano-Pérez. Tiptop: A modular open-vocabulary planning system for robotic manipulation, 2026. URL https://arxiv.org/abs/2603.09971.

[36] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, Q. Vuong, T. Zhang, T.-W. E. Lee, K.-H. Lee, P. Xu, S. Kirmani, Y. Zhu, A. Zeng, K. Hausman, N. Heess, C. Finn, S. Levine, and B. Ichter. Pivot: Iterative visual prompting elicits actionable knowledge for vlms, 2024. URL https://arxiv.org/abs/2402.07872.

[37] F. Liu, K. Fang, P. Abbeel, and S. Levine. Moka: Open-vocabulary robotic manipulation through mark-based visual prompting, 2024.

[38] A. J. Sathyamoorthy, K. Weerakoon, M. Elnoor, A. Zore, B. Ichter, F. Xia, J. Tan, W. Yu, and D. Manocha. Convoi: Context-aware navigation using vision language models in outdoor and indoor environments, 2024. URL https://arxiv.org/abs/2403.15637.

[39] Y. J. Ma, J. Hejna, A. Wahid, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, J. Tompson, O. Bastani, D. Jayaraman, W. Yu, T. Zhang, D. Sadigh, and F. Xia. Vision language models are in-context value learners, 2024. URL https://arxiv.org/abs/2411.04549.

[40] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner. Vision-language models are zero-shot reward models for reinforcement learning, 2024. URL https://arxiv.org/abs/2310.12921.

[41] S. Chen, C. Harrison, Y.-C. Lee, A. J. Yang, Z. Ren, L. J. Ratliff, J. Duan, D. Fox, and R. Krishna. Topreward: Token probabilities as hidden zero-shot rewards for robotics, 2026. URL https://arxiv.org/abs/2602.19313.

[42] P. Budzianowski, E. Wiśnios, M. Tyrolski, G. Góral, I. Kulakov, V. Petrenko, and K. Walas. Opengvl – benchmarking visual temporal progress for data curation, 2026. URL https://arxiv.org/abs/2509.17321.

[43] J. Zhang, C. Qian, H. Sun, H. Lu, D. Wang, L. Xue, and H. Liu. Progresslm: Towards progress reasoning in vision-language models, 2026. URL https://arxiv.org/abs/2601.15224.

[44] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models, 2024. URL https://arxiv.org/abs/2310.12931.

[45] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[46] Physical Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

[47] C. Xu, Q. Li, J. Luo, and S. Levine. Rldg: Robotic generalist policy distillation via reinforcement learning, 2024. URL https://arxiv.org/abs/2412.09858.

[48] M. S. Mark, T. Gao, G. G. Sampaio, M. K. Srirama, A. Sharma, C. Finn, and A. Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone, 2024. URL https://arxiv.org/abs/2412.06685.

[49] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Y. Xie, F. Hu, J. Wu, Z. Luo, L. J. Fan, G. Shi, and Y. Zhu. Self-improving vision-language-action models with data generation via residual rl, 2025. URL https://arxiv.org/abs/2511.00091.

[50] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control, 2018. URL https://arxiv.org/abs/1812.03201.

[51] S. Ding, K. Hu, S. Zhong, H. Luo, W. Zhang, J. Wang, J. Wang, and Y. Shi. Genpo: Generative diffusion models meet on-policy reinforcement learning. *Advances in Neural Information Processing Systems*, 38:130443–130474, 2026.

[52] J. Lu, X. Qin, Y. Jiang, K. Wang, C. Zhang, B. Liang, J. Yang, M. Xu, and L. Zhao. Unified noise steering for efficient human-guided vla adaptation. *arXiv preprint arXiv:2605.10821*, 2026.

[53] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[54] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

[55] A. Jain, M. Zhang, K. Arora, W. Chen, M. Torne, M. Z. Irshad, S. Zakharov, Y. Wang, S. Levine, C. Finn, W.-C. Ma, D. Shah, A. Gupta, and K. Pertsch. Polaris: Scalable real-to-sim evaluations for generalist robot policies, 2025. URL https://arxiv.org/abs/2512.16881.

[56] W. Chen, S. Belkhale, S. Mirchandani, O. Mees, D. Driess, K. Pertsch, and S. Levine. Training strategies for efficient embodied reasoning, 2025.

[57] D. Driess, J. T. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Z. Ren, H. Walke, Q. Vuong, L. X. Shi, and S. Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025. URL https://arxiv.org/abs/2505.23705.

[58] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance. *Conference on Robot Learning (CoRL)*, 2024.

[59] J. Kwok, C. Agia, R. Sinha, M. Foutter, S. Li, I. Stoica, A. Mirhoseini, and M. Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models, 2025. URL https://arxiv.org/abs/2506.17811.

[60] Q. Li, S. Park, and S. Levine. Decoupled q-chunking, 2025. URL https://arxiv.org/abs/2512.10926.

[61] Q. Li, Z. Zhou, and S. Levine. Reinforcement learning with action chunking, 2026. URL https://arxiv.org/abs/2507.07969.

[62] Gemini Robotics Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, S. Bohez, K. Bousmalis, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, O. Chang, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, D. D'Ambrosio, S. Dasari, T. Davchev, C. Devin, N. D. Palo, T. Ding, A. Dostmohamed, D. Driess, Y. Du, D. Dwibedi, M. Elabd, C. Fantacci, C. Fong, E. Frey, C. Fu, M. Giustina, K. Gopalakrishnan, L. Graesser, L. Hasenclever, N. Heess, B. Hernaez, A. Herzog, R. A. Hofer, J. Humplik, A. Iscen, M. G. Jacob, D. Jain, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, C. Kew, J. Kirkland, S. Kirmani, Y. Kuang, T. Lampe, A. Laurens, I. Leal, A. X. Lee, T.-W. E. Lee, J. Liang, Y. Lin, S. Maddineni, A. Majumdar, A. H. Michaely, R. Moreno, M. Neunert, F. Nori, C. Parada, E. Parisotto, P. Pastor, A. Pooley, K. Rao, K. Reymann, D. Sadigh, S. Saliceti, P. Sanketi, P. Sermanet, D. Shah, M. Sharma, K. Shea, C. Shu, V. Sindhwani, S. Singh, R. Soricut, J. T. Springenberg, R. Sterneck, R. Surdulescu, J. Tan, J. Tompson, V. Vanhoucke, J. Varley, G. Vesom, G. Vezzani, O. Vinyals, A. Wahid, S. Welker, P. Wohlhart, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, Y. Yang, R. Yao, S. Yaroshenko, W. Yu, W. Yuan, J. Zhang, T. Zhang, A. Zhou, and Y. Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

[63] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2025. URL https://arxiv.org/abs/2510.03827.

[64] G. Wang, C. Zhang, Q. Liu, J. Zhang, J. Cai, J. Liu, and X. Liu. Libero-x: Robustness litmus for vision-language-action models, 2026. URL https://arxiv.org/abs/2602.06556.

[65] T. J. Boerner, S. Deems, T. R. Furlani, S. L. Knuth, and J. Towns. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing*, PEARC '23, page 173–176. ACM, 2023. doi:10.1145/3569951.3597559. URL http://dx.doi.org/10.1145/3569951.3597559.

[66] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6038–6047, 2023.

[67] G. Kim, T. Kwon, and J. C. Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2426–2435, 2022.

[68] N. Tumanyan, M. Geyer, S. Bagon, and T. Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1921–1930, 2023.

[69] B. Wallace, A. Gokul, and N. Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22532–22541, 2023.

[70] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

[71] X. Su, J. Song, C. Meng, and S. Ermon. Dual diffusion implicit bridges for image-to-image translation. *arXiv preprint arXiv:2203.08382*, 2022.

[72] L. Rout, Y. Chen, N. Ruiz, C. Caramanis, S. Shakkottai, and W.-S. Chu. Semantic image inversion and editing using rectified stochastic differential equations. *arXiv preprint arXiv:2410.10792*, 2024.

[73] J. Wang, J. Pu, Z. Qi, J. Guo, Y. Ma, N. Huang, Y. Chen, X. Li, and Y. Shan. Taming rectified flow for inversion and editing. *arXiv preprint arXiv:2411.04746*, 2024.

[74] Y. Deng, X. He, C. Mei, P. Wang, and F. Tang. Fireflow: Fast inversion of rectified flow for image semantic editing. *arXiv preprint arXiv:2412.07517*, 2024.

[75] O. Avrahami, O. Patashnik, O. Fried, E. Nemchinov, K. Aberman, D. Lischinski, and D. Cohen-Or. Stable flow: Vital layers for training-free image editing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7877–7888, 2025.

[76] G. Jiao, B. Huang, K.-C. Wang, and R. Liao. Uniedit-flow: Unleashing inversion and editing in the era of flow models. *arXiv preprint arXiv:2504.13109*, 2025.

[77] H. Chihaoui, A. Lemkhenter, and P. Favaro. Blind image restoration via fast diffusion inversion. *Advances in Neural Information Processing Systems*, 37:34513–34532, 2024.

[78] Z. Yang, K. Zeng, K. Chen, H. Fang, W. Zhang, and N. Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12162–12171, 2024.

[79] Z. Wang, C. Chen, Y. Zeng, L. Lyu, and S. Ma. Where did i come from? origin attribution of ai-generated images. *Advances in neural information processing systems*, 36:74478–74500, 2023.

[80] G. Cazenavette, A. Sud, T. Leung, and B. Usman. Fakeinversion: Learning to detect images from unseen text-to-image models by inverting stable diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10759–10769, 2024.

[81] H. Manor and T. Michaeli. Zero-shot unsupervised and text-based audio editing using ddpm inversion. *arXiv preprint arXiv:2402.10009*, 2024.

[82] H. Liu, J. Wang, X. Li, W. Wang, Q. Chen, R. Huang, Y. Liu, J. Xu, and Z. Zhao. Medic: Zero-shot music editing with disentangled inversion control. *arXiv preprint arXiv:2407.13220*, 2024.

[83] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

[84] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine. Serl: A software suite for sample-efficient robotic reinforcement learning, 2025. URL https://arxiv.org/abs/2401.16013.

[85] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning, 2025. URL https://arxiv.org/abs/2410.21845.

[86] Gemini Team. Gemini: A family of highly capable multimodal models, 2024.

[87] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning*, 2024.

[88] W. Chen, J. S. Bhatia, C. Glossop, N. Mathihalli, R. Doshi, A. Tang, D. Driess, K. Pertsch, and S. Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control, 2026. URL https://arxiv.org/abs/2602.13193.

[89] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts, 2019.

[90] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

[91] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

[92] Y. Luo, J. Kay, E. Grefenstette, and M. P. Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions, 2023. URL https://arxiv.org/abs/2303.17396.

[93] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning, 2021. URL https://arxiv.org/abs/2106.06860.

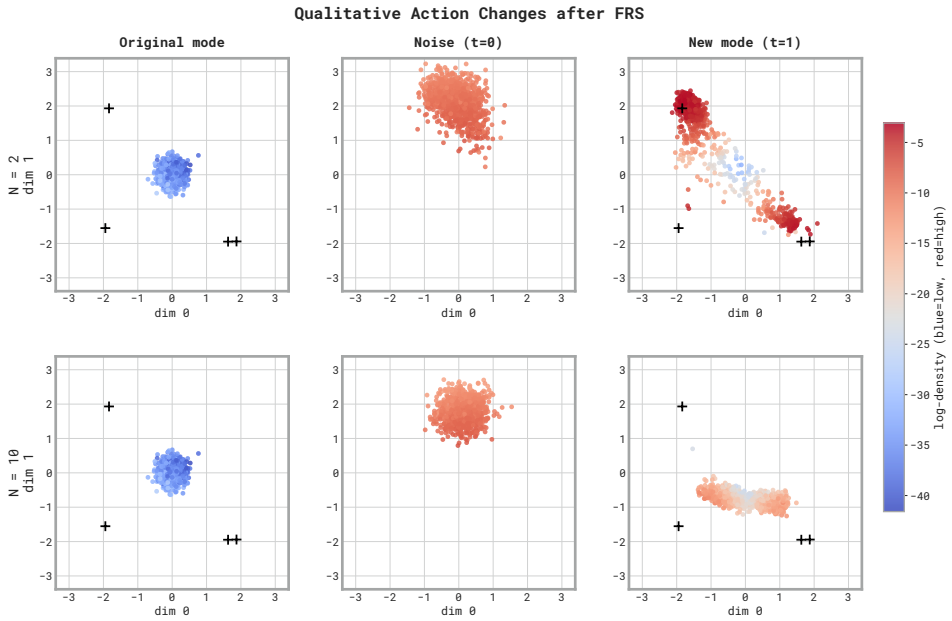


Fig. 10: In a simplified setting, FRS brings actions from the original out-of-distribution mode into more in-distribution actions as measured by log-likelihood. Fewer steps in the Euler approximation results in actions that are more in-distribution but farther from the original mode.

A Additional Related Work

Flow and diffusion inversion in other domains. While this work has focused on applying flow reversal to steering robotic control policies, flow and diffusion reversal (or "inversion") approaches have found applications in a variety of other domains. In particular, in image domains, inversion-based approaches have enabled controllable image editing both for image generation diffusion models sampled with deterministic DDIM [53] sampling [66–71] and flow models [72–76]. Other applications of flow and diffusion inversion in image domains include image restoration [77], watermarking [78], and detection of AI-generated images [79, 80]. Outside of image domains, diffusion inversion has also been applied to editing audio signals [81, 82]. While the techniques utilized in these works are similar to our flow inversion approach, the applications differ substantially—our work shows that flow inversion can lead to significant performance improvements in robotic domains.

B Illustrative Examples of Flow Reversal Steering

B.1 Idealized FRS Example

As the distribution of large-scale flow matching policies such as those used in the paper precludes calculation of log-probabilities of the pre- and post-FRS actions, we consider an idealized setting where the action distribution is fixed to a mixture of Gaussians to allow for exact calculation of the dynamics of FRS. We are interested in three questions:

1. What does the change in coarse action distribution look like qualitiatively?
2. Does FRS make actions more in-distribution? We expect a positive **log-density ratio**, defined as (total log-likelihood post-FRS) - (total log-likelihood pre-FRS).
3. Does FRS keep actions near their respective modes? We should expect FRS to have an average movement of all coarse steering actions by a small amount in action space.

To empirically analyze these questions, we design a setting with four "action modes" in 7-dimensional space, all with standard deviation 0.5 with means in $[-2, 2]^7$. Pairwise distances between modes are fixed to be at least 4, and the norm of each center is also at least 4. This ensures the

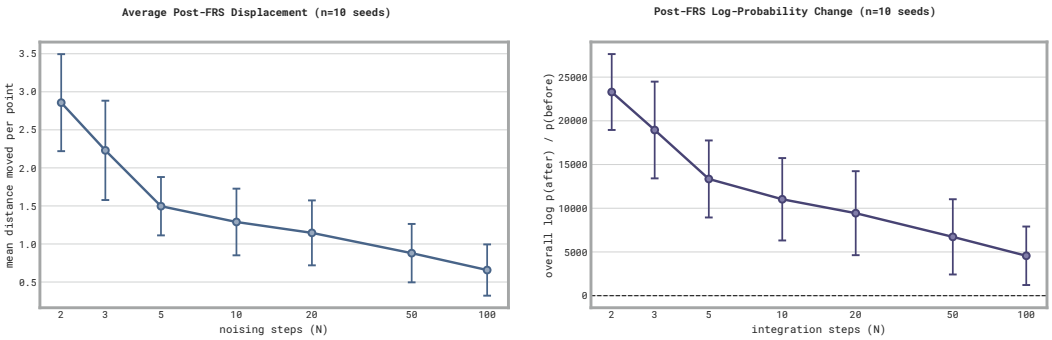


Fig. 11: Fewer steps during FRS result in higher log-density ratio (indicating actions that are more in-distribution on average) but also higher displacement relative to the number of noising steps. Note that on average in this simplified setting, any point from our steering "coarse action" should be expected to have distance around 4 from any mode. We choose 10 noising steps in our robotics experiments as a good balance between action fidelity and improved log-density ratio.

modes are separate and the area around the origin is out-of-distribution. We train a flow-matching model with 10,000 sampled points from this true action distribution using a three-layer MLP.

As a proxy for a steered mode, we sample 1,000 actions from $N(0, 0.2)$, and pass it through $K = [2, 3, 5, 10, 20, 50, 100]$ steps in FRS. We expect that FRS with low K will have higher average movement of each action, but also larger log-probability ratio.

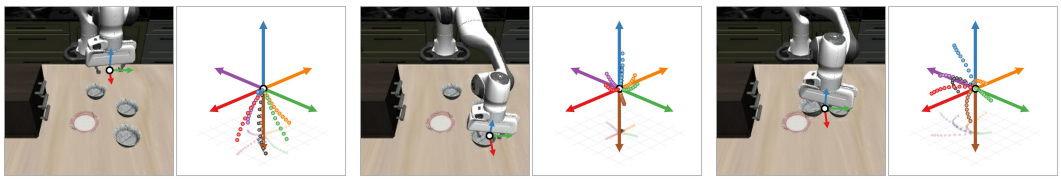
Qualitatively, we visualize $K = 2, 10$ results in Fig. 10. We see that although the original mode has very low-density actions, after noise inversion both the noise seeds and the transformed actions have much higher log-probabilities compared to the original mode. For $K = 2$, we move further towards high density compared to $K = 10$. We conjecture this occurs because integration error during the learned flow process tends to push the policy towards areas with higher probability density, with fewer steps leading to more integration error.

Quantitatively, we confirm over 10 seeds in Fig. 11 that increasing the number of noising steps results in a decrease in the mean distance moved per point, but also a decrease in round-trip log-density ratio, meaning actions are staying closer to the mode but also ending up less in-distribution. With infinite steps, we expect the overall ratio to converge to zero and the displacement to also converge to near-zero. We choose 10 noise steps of size 0.1 in our simulation and real-world robotic experiments to balance these two concerns. We leave optimization of this technique to future work.

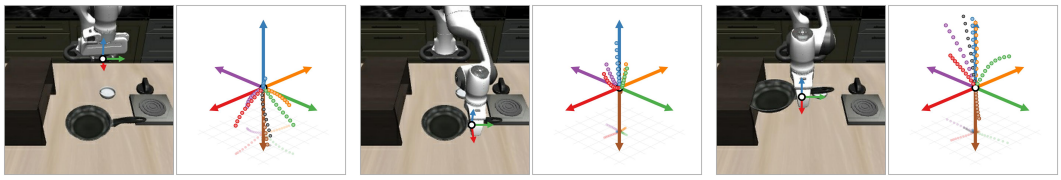
B.2 Robot FRS Example

In Fig. 12, we show actual examples of running FRS in LIBERO-90. The base policy is $\pi_{0.5}$-LIBERO, which we reiterate was not trained on 90 – it thus just places a wide distribution over "reasonable" actions. We steer it with directional actions which we display as solid arrows, both overlaid on the robot observation *and* in auxiliary 3D plots (viewed at a three-quarters angle). Naturally, these are very coarse reference actions (executing them directly on the robot is unlikely to be effective). The corresponding colored dotted lines are the actions produced by running FRS on the coarse arrow actions, while the black represents a sample from the base VLA (which is often incorrect for the task, as these settings are OOD). We always use 10 integration steps ($h = 0.1$).

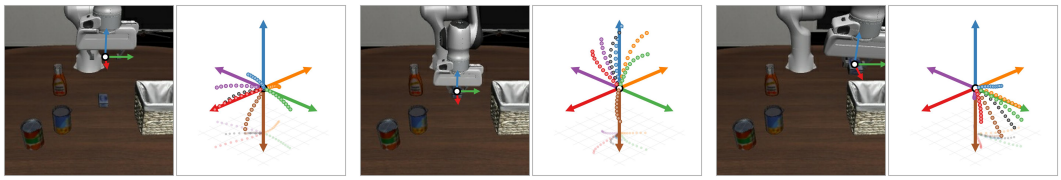
As desired, the steered actions roughly follow the directional reference actions. However, they do not perfectly reconstruct them – instead resembling samples from a nearby action mode of the base generalist VLA. They also depend heavily on the observation: e.g., starting frames tend to bias steering downward (as the policy usually moves down to grasp), or when grasping an object the actions are biased upwards to lift it. Additionally, when different steering directions are close to different "reasonable" behaviors (grasping different objects, placing a held object in different reasonable places), then those directions map to actions that correspond to those behavioral modes.



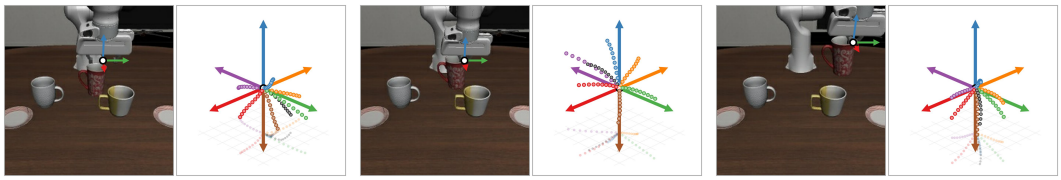
(a) The VLA initially can be steered toward reaching any of the 3 bowls. Once one is grasped, the steered actions are biased up to lift the bowl. Finally, steering can either push the policy to put the bowl on the plate (correct) *or* atop the cabinet (blue, incorrect) – both are "reasonable" behaviors.



(b) Initial steering is flexible, as reaching for the bowl, knob, or pan are all reasonable. Grasps causes bias towards lifting, and once lifted, main action modes are towards the shelf, above the shelf, or to the stove (green).



(c) Initial steering is again very flexible, as moving to any object left of the basket is reasonable. Grasping biases samples upwards. Finally, once close to the basket, samples are biased towards moving to place the object inside.



(d) Initial steering can move towards any of the three mugs (but all are biased to moving down). Once grasped, there is significant variation, as placing on either plate is reasonable (left and right with purple and green have strong reconstructions). Once above a plate, actions are biased toward moving down for placing.

Fig. 12: Examples of running FRS in LIBERO. The colored arrows are the cardinal reference actions, and the dotted points are the FRS actions they result in (black is the policy's base action). We display these actions in 3D side views for clarity, but note that the red, green, and blue axes correspond to the ones overlaid on the robot images (forward, right, and up respectively). In general, the actions move in the rough direction specified by the reference, but are more akin to actions drawn from the VLA's distribution and are highly dependent on the current observation.

C What is Going On in Flow Policy Noise Space?

In testing FRS, we obtained several insights into the structure of flow policies' noise space. We ran several ablative and auxiliary experiments that allow us to form a more complete characterization of noise space. For convenience, we first summarize all results in App. C.1. These experiments also motivate certain design choices when running FRS in practice, which we outline in App. C.5.

C.1 Summary of Noise Space Insights

App. C.2:

- When running flow reversal, more likely noises (i.e., ones closer to 0) tend to map to more in-distribution actions.
- Flow reversal is better able to find noises that reconstruct to more in-distribution actions (lower reconstruction MSE).

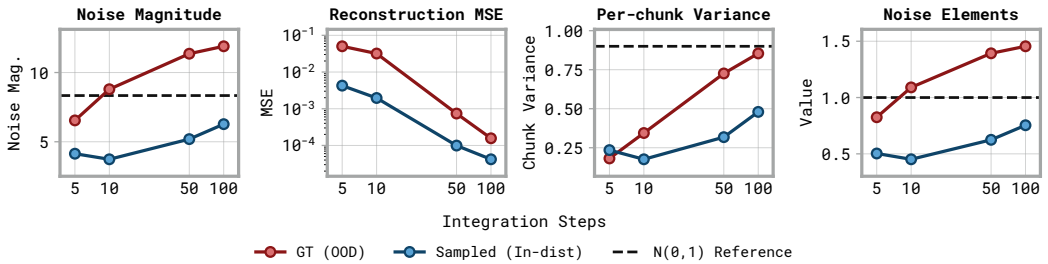


Fig. 13: How mean noise magnitude, reconstruction MSE, noise value distribution, and per-chunk variance change with different numbers of integrations steps when applying FRS to the LIBERO-90 dataset.

- Increasing number of integration steps for flow reversal leads to more accurate reconstruction (due to lower integration error), but at the cost of going more OOD in noise space.

App. C.3:

- FRS finds good regions of noise space. Noises *near* good noises also map to good actions.
- Noises corresponding to padding elements in action chunks do not seem to affect denoising much.

App. C.4:

- When applying flow reversal, action chunks tend to map to temporally correlated noise chunks, with lower variance across the chunk axis than if sampled from $\mathcal{N}(0, I)$.
- When an *action* is averaged and repeated over the chunk axis, it can still find good noises via flow reversal (at least for delta end effector control, where averaging is sensible).
- Likewise, when a *noise* produced by flow reversal is averaged and repeated over the chunk axis, it can still denoise to good actions.

C.2 FRS on In- and Out-of-Distribution Actions

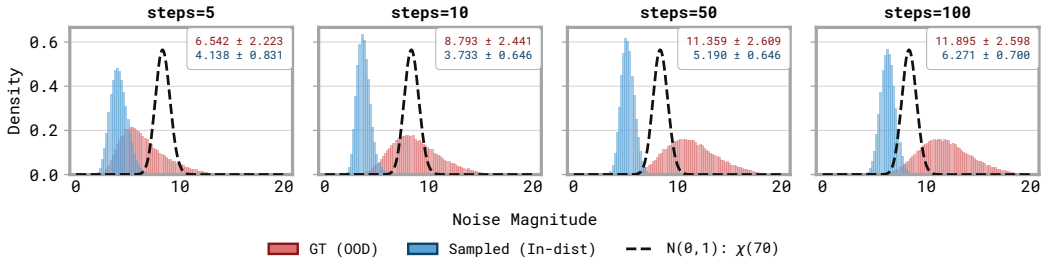
We first investigate how in- vs. out-of-distribution actions affect flow reversal. We use the LIBERO-90 filtered dataset released by Chen et al. [56] in conjunction with the $\pi_{0.5}$-LIBERO checkpoint released by Physical Intelligence et al. [16]. For every 10th frame in that dataset, we procure two action chunks: (1) the ground-truth actions from the dataset (As $\pi_{0.5}$-LIBERO was *not* trained on 90, the actions from the LIBERO-90 dataset are OOD for it) and (2) a single sampled action from $\pi_{0.5}$-LIBERO, given each frame (which is naturally ID).

Now that we have ID and OOD actions, we run flow reversal for both sets with different numbers of integration steps. Specifically, we use $\{5, 10, 50, 100\}$ integration steps. Both noising with flow reversal and denoising with standard flow matching use the same number of steps in each of the four conditions. See Fig. 13 for aggregated trends and Fig. 14 for more details.

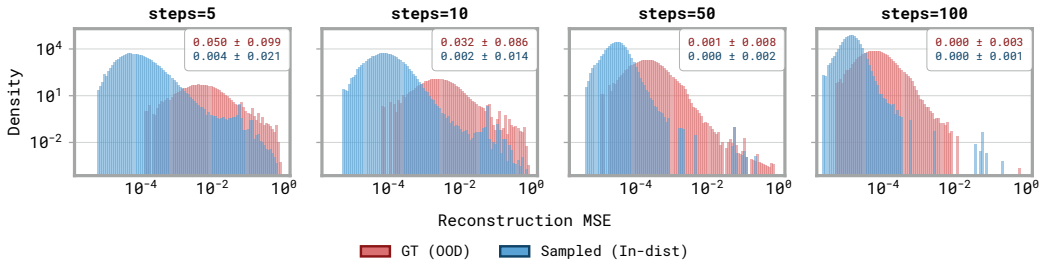
OOD actions map to less likely noises. Intuitively, we find that ID actions consistently map to noise vectors of smaller L_2 magnitude (Fig. 14a). Since noises are typically sampled from $\mathcal{N}(0, I)$, smaller magnitude noises are more likely to be sampled from the noise prior. Flow reversal's noises thus seem to act as a pseudo proxy for how in-distribution the reference actions are. The fact that the noises from zero-shot VLM FRS also tends to have higher noise magnitudes supports this as well, as the straight directional actions emitted by the VLM are OOD (Fig. 15).

When we denoise the outputs of flow reversal, we unsurprisingly find that OOD actions have higher reconstruction MSE than the ID ones (Fig. 14b). It is thus *easier* to find noises that reconstruct to ID actions than OOD ones.

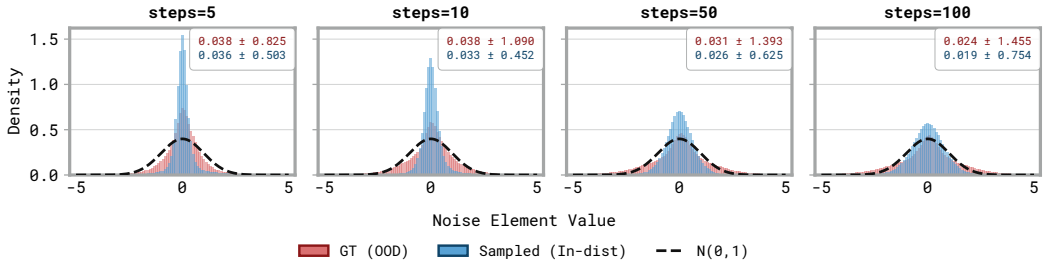
Integration steps control noise likelihood and reconstruction error. For both ID and OOD actions, we find that fewer integration steps results in lower noise magnitudes (Fig. 14a), but higher reconstruction MSE (Fig. 14b). As the number of steps increases, reconstruction MSE decreases



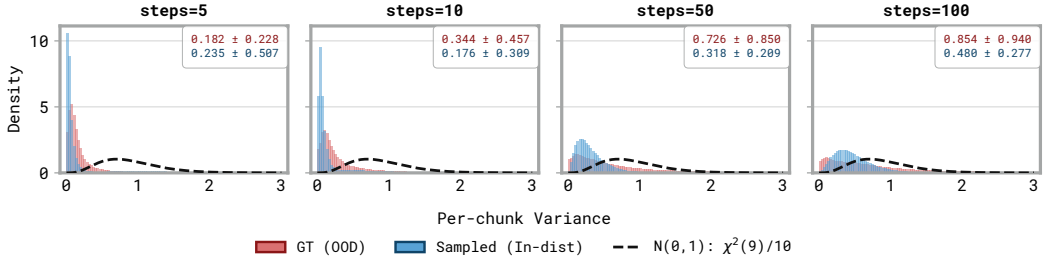
(a) Magnitude of noises produced by running FRS on LIBERO-90 data. Black is the distribution if noises were drawn from $\mathcal{N}(0, I)$, following a χ distribution.



(b) Reconstruction MSE when running FRS on LIBERO-90 data.



(c) Distribution of noise values when running FRS on LIBERO-90 data. Black is the distribution if noises were drawn from $\mathcal{N}(0, I)$.



(d) Variance of noise values across the chunk axis when running FRS on LIBERO-90 data.

Fig. 14: Running FRS on the full LIBERO-90 dataset with varying integration steps. Black is if noises were drawn from $\mathcal{N}(0, I)$, empirical mean and standard deviation are also reported.

(the reconstruction is *better*), but the flow reversal noises grow in magnitude (and, indeed the overall variance of noise elements Fig. 14c. See Fig. 13 for aggregated trends.

The reconstruction trend is expected: more integration steps means there is less integration error, so action reconstructions should get better. In turn, we interpret the noise magnitude trend as suggesting that, by using more integration steps, flow reversal finds more OOD noises that more closely denoise to the reference action.

We thus opt to use 10 integration steps for all our main experiments – not only is this the default for OpenPi flow-matching policies [16, 22], but using more could result in flow reversal finding more OOD/high-magnitude noises, which may be harder to use when learning noise policies for DSBC or DSRL + FRS.

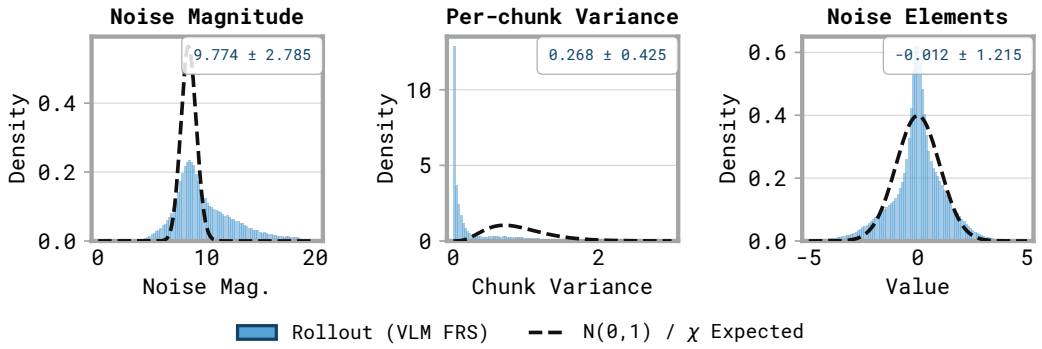


Fig. 15: Analysis of noise statistics from our zero-shot VLM FRS experiments on LIBERO-90.

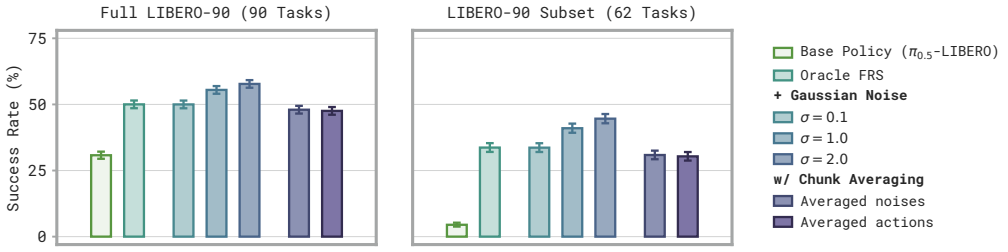


Fig. 16: LIBERO-90 success rates from running FRS using an *oracle* VLA to steer, as well as ablations. The 62-task subset is the one used for zero-shot VLM FRS experiments.

C.3 Better Reference Actions Enable Better Steering Performance

Perhaps unsurprisingly, FRS's zero-shot performance improves as the quality of the steering reference actions improves – e.g., if we replace our coarse cardinal steering actions with high-quality, fine-grained action chunks from an expert policy.

Steering with fine-grained oracle actions. We again consider $\pi_{0.5}$-LIBERO achieves 30.8% on all of LIBERO-90 and 4.5% on the challenging task subset we consider in Sec. 5.2. Rather than using coarse actions from a VLM for steering it, we instead use the expert $\pi_{0.5}$ policy trained by Jain et al. [55], which was trained only on LIBERO-90 and achieves 93.9% performance on that benchmark (92.8% on the challenging tasks). The latter VLA thus acts as as "oracle" for LIBERO-90. See Fig. 16 for all results and ablations.

We find that using FRS with expert VLA steering achieves 50% on all of LIBERO-90 (33.7% on the challenging split, higher than the 11.1% from VLM steering). This does not fully recover the oracle's performance, though still shows that using better reference actions increases zero-shot FRS performance.

FRS finds good *regions* of noise space. We also try adding 0-mean Gaussian noise to the noise deterministically produced by FRS. For reference action a_1 sampled from the oracle, we run FRS $\hat{a}_0 \leftarrow \mu_\theta^{-1}(a_1, o)$ as normal, then execute action $\hat{a}_1$ produced via:

$$\hat{a}_1 \leftarrow \mu_\theta(\hat{a}_0 + \sigma\epsilon, o);\ \epsilon \sim \mathcal{N}(0, I) \tag{1}$$

We test $\sigma \in \{0.1, 1, 2\}$. As shown in Fig. 16, while 0.1 keeps performance the same, we unexpectedly found that using noise scales of 1 or 2 actually substantially boost performance. For simplicity, we do not add extra noise in our main experiments. Still, this shows that it is *not* important to denoise the exact noise outputted by FRS; noises close to it also denoise to good actions.

Padding noises do not affect performance. $\pi_{0.5}$-based VLAs also predict padding action elements, such that its action expert can yield a fixed-size output [16]. Our reference actions use the ground-truth padding value to make actions said fixed size. The noises corresponding to those padding

elements do not seem to affect the output significantly, nor do they affect performance (as verified by setting the padding noises to $\mathcal{N}(0, I)$ after running flow reversal, then denoising). We adopt this technique in all our FRS instantiations.

C.4 Noise Averaging/Repeating is Benign or Beneficial

The noises produced by FRS tend to empirically vary across the action chunk axis less than if they were sampled i.i.d. from $\mathcal{N}(0, I)$. This is the case for VLM steering, where the reference action chunks a_1 passed through flow reversal are already repeated across the chunk axis (Fig. 15, center) – though, in principle, this does not mean that the individual actions *must* map to similar noises. It is *also* the case when running FRS on the LIBERO-90 data that the oracle was trained on (Fig. 14d) – even though said reference actions are *not* repeated, they simply may have low-frequency temporal correlations [83].

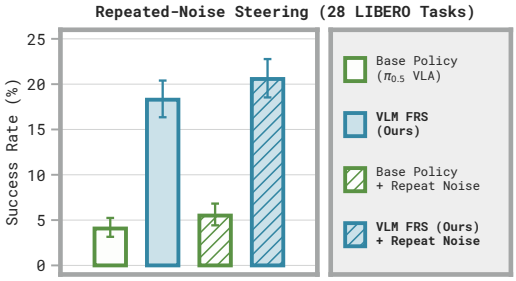


Fig. 17: When running zero-shot FRS, averaging and repeating the noises used for flow reversal still yields good performance.

Averaging and repeating noises. In turn, we find that averaging and repeating the noises from FRS over the chunk axis still comparably boosts performance of zero-shot FRS in LIBERO, when using either VLM or oracle policy actions for steering (Fig. 17 and Fig. 16 respectively). Alternatively, averaging the oracle's actions over the chunk axis still works, motivating our use of uni-directional reference actions.

This further suggests that high fidelity of flow reversal is not necessary; even when averaged, the result ends up in part of noise space mapping to a good behavioral mode. It also corroborates the findings from DSRL [15], where flow steering with repeated noises is both (1) effective at eliciting beneficial robot behaviors and (2) makes finding and predicting good noises easier. *We therefore use noise averaging and repetition for all our LIBERO DSBC and DSRL + FRS experiments.*

C.5 Takeaways for Running FRS

The key FRS design details motivated by this analysis are:

1. Running FRS by averaging and repeating the noise across the chunk axis does not affect performance. All our noise policies are trained to predict such chunk-averaged noises, as it makes learning easier.

2. Averaging and repeating the *good reference actions themselves* can still find good noises. This motivates our use of directional actions for steering (rather than more sophisticated ones).

3. Reference actions should have their padding elements set correctly. Once the underlying noise is computed with flow reversal, the padding *noises* can be set to $\mathcal{N}(0, I)$ without affecting performance.

4. When running flow reversal, there is inverse correlation between reconstruction fidelity and the resulting noises' magnitudes. Increasing the number of integration steps increases fidelity, but makes the noises higher in magnitude (i.e., more out-of-distribution). We err on the side of using fewer steps by choosing the default of 10, as (1) this makes policy inference faster, (2) we want smaller noises for easier noise policy learning, and (3) extremely high action reconstruction fidelity is not always beneficial, especially when reference actions are coarse.

For clarity, these takeaways are also expanded upon in subsequent sections.

D Steering Implementation Details

We now discuss the design choices we made in implementing steering interfaces for both the human operator and the VLM reasoner. We emphasize that, while empirically effective, **these systems are *not* fundamental to FRS, nor are they our main contribution.** More sophisticated VLM systems or human steering interfaces can likewise be used with FRS. See Fig. 2, left.

D.1 Human Steering Interface

For our human FRS experiments, we let an operator steer the arm via keystrokes: six keys (`W/A/S/D/Q/E`) for the three translational Cartesian axes (forward/back, left/right, up/down), a key to change gripper state, and a key to defer to the policy. Rotations are never commanded from the keyboard.

Extracting actions. Each keystroke is turned into a full reference chunk of absolute joint-position targets with the DROID codebase's inverse kinematics [18], starting from the arm's current state. Two hyperparameters govern this. First, `keyboard_pos_scale` (10.0) divides the default Cartesian step size of 1, providing the conversion between keyboard inputs and Cartesian units. This is likewise used in other past teleoperation schemes, e.g., HIL-SERL [84, 85] uses this kind of scale factor when converting a 3D SpaceMouse's inputs to Franka actions. Second, `action_repeats` (5) sets how many steps that motion is held for, i.e., the length of the reference chunk. The resulting actions are seven joint dimensions plus one gripper dimension, repeated `action_repeats` times.

The gripper key cycles through three states: *open*, *closed*, or *noise*. In open/closed mode the gripper dimension is driven to the commanded value and held across steering. In noise mode the gripper is left to the policy: the gripper element of the chunk is held at its current value and its corresponding noise element is randomized, so the denoised action is free to open or close based on the observation.

After passing the small reference through IK to get joint actions, we tile it until reaching the policy chunk length (15, e.g., repeat the length-5 joint angle chunks thrice). This is what gets passed through flow reversal. In turn, we set the tiled parts of the chunk (e.g., everything after the first 5 actions) to $\mathcal{N}(0, I)$ noise, allowing it to be denoised by the policy based on observation. This is akin to diffusion in-painting, albeit in noise space – that is, we want the steered action to match part of the reference chunk, while the rest of it is "filled in" by what the policy thinks is reasonable.

The final human steering loop involves the person pressing a key given the current step, converting it to an action chunk programmatically using inverse kinematics, running flow reversal, reconstructing the action, then executing some or all of it before the operator is queried again. We opt to execute the first 10 actions in the length-15 chunk. As we limit episodes to 400 low-level action steps, each episode requires the human to issue at most 40 keypress reference actions, each 10 low-level steps apart – compared to having to provide continuous dense actions via true teleoperation.

D.2 VLM Steering Prompt and Interface

For the VLM steering experiments, we use a simple prompt to infer directional actions similar to the humans'. Specifically, we query the Gemini-ER-1.6 VLM [62, 86] with the overall task language, and ask it produce a structured JSON output containing:

1. A brief natural-language embodied chain-of-thought reasoning [62, 87, 88], asking it to decide the immediate task-appropriate subgoal for the robot to pursue.
2. Whether or not the robot needs to make a fine manipulation by deferring to the base policy. If so, this overrides the below two terms.
3. If not, the specific axis-aligned end-effector motions necessary for the robot to take, represented with a 3-element vector specifying if the motion should be positive, zero, or negative along each of the three Cartesian axes.
4. How large said motion should be ("more" or "less").

The VLM itself receives the third-person external camera view (*not* the wrist camera), which is annotated with a translucent vertical plumb line from tabletop to gripper. We include this as the front

```
You are an expert giving commands to a robot gripper from a third-person view.  Focus on the gripper (centered
at the top of the blue line), not the base of the arm.  The blue line coming down from the gripper ends at the
table surface (but is NOT an object, only a guide for navigation).

The task is:  <task language>
The task language refers to locations from the camera's perspective.  We should use the following steps to get
to our solution:
1.  Understand the task:  left and right are from the image's perspective.  Front is closer to the foreground,
back is closer to the background and the robot arm's base.  Up is away from the table, down is closer to it.
2a.  Output one line describing the immediate next steps to accomplish this task.  If we are still far from
our next destination, we should say how we should move spatially to get there.  If we are close, describe
whether smaller (fine) motor adjustments are necessary or if spatial movement is sufficient.  If manipulation
may be in progress (gripper not closed around an object and not open and away from object) default towards
fine motor skills.
2b.  Use the bottom of the blue line (on the table surface) to judge whether the gripper needs to move closer
(bring bottom of line towards bottom of image) or farther (bottom of line towards top of image) from the
camera.  In these situations, default to a zero z-axis command unless the gripper is roughly aligned with the
object or needs to move up to avoid obstruction.
3.  Output our structured action:  Set the "fine" flag if fine motor skills (e.g.  positioning, grasping,
pushing, pulling, rotating) are needed as the next low-level action.  Otherwise, give the principal axes of
the next spatial motion in [x, y, z] space.  If we could potentially get stuck on nearby obstacles, prefer to
move in the x-y plane before moving down and prefer to move up before moving in the x-y plane.
4.  Use the "motion_amount" field to determine how much to move.  E.g., if the gripper is far from the target
object, "motion_amount" should be "more" to indicate that significant motion in the specified direction is
needed.  If the gripper is close to the target object, "motion_amount" should be "less" to indicate that only
a small adjustment in the specified direction is needed.

Hints:
[Base:
- The ketchup is in a bottle, while the tomato sauce is in a red can.
]
[Transfer:
- The alphabet soup is in a blue can, while the tomato sauce is in a red can.
- The ramekin is white, the cookie box is mainly yellow and red.
- The salad dressing is in a white and green bottle.  The ketchup is in a red bottle with a white lid.  The
BBQ sauce is in a small red bottle.
- The butter is in a red box, while the cream cheese is in a blue box.
]
- "Middle" does not necessarily mean centered in the table, it means relative to other objects.  E.g., if
there are three bowls, then the middle bowl is the one that is between the other two, even if it's not in the
middle of the scene.

The output should be a JSON with fields:
"reasoning":  str, brief 1 sentence reasoning following the steps indicated above.
"fine":  bool.
"coords":  [x, y, z] with each of x, y, z in -1, 0, 1.
"motion_amount":  str, either "less" or "more".

fine:
False = coarse motion:  use `coords` to steer.  Set `coords` to the discrete direction the gripper should move
next; combine axes when motion is oblique.
True = fine / manipulate (close fingers on object, place, hold steady, fine vertical adjustments).  When `fine
= True`, `coords` should be `[0, 0, 0]`.

Axes for coords:  x = **farther** from camera (-1) / none (0) / **closer** to camera (+1) y = image **left**
(-1) / none (0) / image **right** (+1) z = world **down** (-1) / none (0) / world **up** (+1)

`motion_amount`:
"less" = Less motion in the specified direction is needed.
"more" = Significant motion in the specified direction is needed.

If you are close to grasping or placing an object, ALWAYS prefer outputting fine manipulation over coords and
less motion!  Motion should only be used for larger adjustments.
```

Fig. 18: Prompt for querying Gemini for motions. Brackets indicate minor changes between LIBERO-90 and the other experiments.

view camera used by LIBERO makes depth perception difficult; while policies are able to remediate this with wrist camera view, we found VLMs to be empirically bad at synthesizing information from third-person and wrist views (at least without substantial VLM system engineering). We note that LIBERO provides the table height, end effector pose, and camera calibration matrices necessary to compute the pixels for the plumb line. Calibration and end effector pose are likewise easily extracted in real-world robot setups, e.g., using a ChArUco board with the DROID robot workcell [18].

If a motion is specified in the VLM output, the vector is normalized before being projected into actions. If the motion should be small, then this vector is in turn halved. Note that this is more flexible than the human interface, as the VLM can issue *combinations* of axis-aligned motions at each step. This is a design choice – a more sophisticated interface would allow a human to likewise issue more complex motions, albeit at the cost of intuitiveness. See Fig. 18 for the full prompt and Fig. 19 for example outputs. This prompt is used for *all* zero-shot VLM control and steering experiments in Sec. 5.2, not just FRS.

Note that, as we demonstrate in App. C.3, better steering actions lead to better zero-shot FRS performance. While we demonstrate that FRS can already achieve substantial performance gains with

Fig. 19: Example outputs of VLM steering in LIBERO. Red indicates the direction suggested by the VLM.

our simple and naïve cardinal action VLM system, *this suggests that procuring better actions from a more engineered and complex VLM system would likely improve FRS's zero-shot performance even further*. We leave such VLM engineering improvements to future work.

Extracting actions. LIBERO uses delta end effector actions, wherein the first three terms correspond to Cartesian motions. As $\pi_{0.5}$-LIBERO already includes normalization parameters, we consider the VLM's outputs as *normalized Cartesian actions* – that is, +1 along the x-axis means moving the maximum delta-x value recorded in the normalization statistics, -1 means moving the minimum, and 0 corresponds to the average between those two extremes (which might not be 0). All unnormalized rotations are set to 0 (none), as is the gripper value (halfway between fully open and closed). Note that this does *not* force the end effector to never rotate, nor the gripper to stay partially open – depending on the observed scene, both still seem to denoise to appropriate values. Alternative ways of setting these values can also work, e.g., having the VLM also set them, injecting stochasticity into those elements' noises specifically, etc.

This VLM steering system does *not* require any advanced APIs for control, in contrast to past zero-shot control works [32]; all actions it produces are with this simple programmatic conversion scheme, and are represented in the same LIBERO action space used by learned policies (as steering requires reference actions and policies' sampled actions to live in the same space).

D.3 Action Padding

$\pi_{0.5}$ always predicts 32-dimensional actions, so as to accommodate predicting actions for two bimanual arms and navigation [16]. For DROID and LIBERO, which have 8D and 7D actions respec-

Table 1: Hyperparameters for DSBC and DSRL + FRS on LIBERO-90 with the frozen $\pi_{0.5}$ policy.

Hyperparameter	**DSBC**	**DSRL + FRS (App. E.3.1)**	**DSRL + FRS (App. E.3.2)**
Base policy & environment			
Base policy / model family	$\pi_{0.5}$ (`pi05_libero`)	$\pi_{0.5}$ (`pi05_libero`)	$\pi_{0.5}$ (`pi05_libero`)
Tanh clip magnitude	5.0	5.0	N/A
Query frequency	10	10	10
Image resize	84	84	84
Noise actor network			
Hidden dims	(128, 128, 128)	(128, 128, 128)	(128, 128, 128)
Tanh-squashed policy	True	True	False
Initial std scale	0.01	0.01	0.01
Entropy term	False	False	False
Optimization			
Objective	BC only	SAC + BC auxiliary	SAC + BC auxiliary
Training steps	12,500	200,000	250,000
Batch size	128	16 (BC and SAC)	16 (BC and SAC)
Optimizer	Adam	Adam	Adam
Actor learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}
Critic learning rate	–	3×10^{-4}	3×10^{-4}
Discount γ	–	0.999	0.999
Target smoothing τ	–	0.005	0.005
Critic ensemble size	–	10	10
Critic reduction	–	mean	mean
UTD ratio (`multi_grad_step`)	–	20	10
Start online updates	–	1,000	1,000
SAC actor KL penalty β	–	0.005	0.1
BC loss			
BC loss mode	NLL	NLL	NLL
BC main / aux coef	1.0 (sole objective)	1.0	1.0
BC decay per online success	None	None	0.5
Zero-reg coef	0.0	1.0	0.0
Zero-reg mode	–	KL to $\mathcal{N}(0, I)$	–
FRS prefill data			
Prefill rollouts	VLM-RL avg-noise	VLM-RL avg-noise	VLM-RL avg-noise
Num. trajectories	all successes	20 (success and failure)	1 (success only)

tively, this means the remaining dimensions are filled with some padding element (-1 for DROID and 0 for LIBERO). Thus, when computing reference actions, we likewise set the padding to those values. Once the padded action has been noised via flow reversal, we find that replacing the noises corresponding to padding with true $\mathcal{N}(0, I)$ i.i.d. noise does not affect performance (App. C.3). As ignoring padding noises also means noise policies need to predict from samples from a smaller output space, we universally use this technique.

E Simulation Experimental Details and Hyperparameters

We now present details on the simulated results in Sec. 5.2, Sec. 5.3, and Sec. 5.4. All policy evaluations use 50 trials per task, and all error bars are Wilson confidence intervals.

E.1 Zero-shot Experiments

Tasks. We run zero-shot VLM FRS control on all of LIBERO-Goal, Object, and Spatial, as well as all 62 tasks that $\pi_{0.5}$-LIBERO achieves under 40% success rate on: `[0, 1, 3, 4, 5, 6, 7, 8, 11, 13, 14, 15, 16, 17, 18, 21, 23, 25, 26, 27, 28, 31, 32, 34, 35, 36, 37, 39, 40, 41, 42, 43, 44, 45, 48, 49, 52, 53, 58, 59, 62, 63, 64, 65, 66, 69, 71, 73, 74,75, 76, 78, 79, 80, 81, 83, 84, 85, 86, 87, 88, 89]`. All are run for 50 episodes.

Comparisons. We use 16 parallel action samples per step for the sample-and-rank baseline. We partially-noise to $t = 0.5$ and use 5 denoising steps (each of step size $h = 0.1$) for the partial noising baseline. For all LIBERO experiments, we use a chunk execution horizon of 10 (i.e., predict an action chunk of at least length 10, then execute the first 10 before re-querying the policy), as was done in past works [55, 56].

E.2 DSBC Experiments

E.2.1 Main LIBERO DSBC Details

Tasks. For DSBC, we focus just on the 15 LIBERO-90 tasks where zero-shot FRS makes an improvement of at least 10% (as those are the only ones where there are enough successes to meaningfully run supervised learning on): `13, 31, 35, 41, 43, 44, 49, 52, 69, 73, 74, 75, 78, 79, 81`. We train on all successes of the 50 zero-shot FRS rollouts per task *repeated-noise* rollouts, though similar performance can be achieved with half the data or less. All comparisons are run for 50 rollouters per task.

Hyperparameters. All hyperparameters for DSBC are listed in Table 1.

DSBC training. As with all of our simulated noise policies, the DSBC policy predicts a single 7D vector, corresponding to average noises. To convert it into actual noises, we repeat it along the chunk dimension to the chunk horizon (10 for $\pi_{0.5}$-LIBERO), then pad all actions to be 32D with i.i.d. $\mathcal{N}(0, I)$ noise. As with the original DSRL implementation, we regularize the noise actions by applying tanh clipping to the noise outputs. However, we use a wider clipping range of $[-5, 5]$ (rather than $[-1.1]$ from Wagenmaker et al. [15]), as doing so makes the noise policy more expressive, and able to fit more out-of-distribution actions' noises.

Additionally, we use negative log-likelihood (NLL) as our BC loss. This is a design choice – we also try running with mean-square error loss, and find it works approximately just as well. This has the caveat that it does not train the policy to produce variances (Gaussian policies only learn the mean with standard MSE), and empirically benefits from regularizing the action predictions towards 0 (e.g., either with an auxiliary MSE loss that brings the mean to all 0s, or an auxiliary KL loss with $\mathcal{N}(0, I)$).

Baselines. For the standard BC baseline, we train a small policy of the same architecture as DSBC on the successful *robot* action chunks from FRS (*not* noise actions). It predicts the full action chunk, which is naturally not averaged nor repeated. It is trained with NLL as the BC loss (same as our DSBC policy). This uses a Gaussian policy, which is sufficient for learning LIBERO tasks (whereas real standard BC requires a flow policy).

E.2.2 Offline DSBC in LIBERO

Online DSBC experiments is akin to HG-DAgger [89] or online RL: it uses data from online rollouts controlled via FRS, as that confirms whether or not the noises produced by noise reversal actually map to good actions (akin to receiving rewards). However, given how "forgiving" noise space is, can we run FRS on a completely offline (normal) action dataset to train policies with DSBC?

We use the noises produced in App. C.2 by running FRS on the entire LIBERO-90 dataset, which is usually used for standard BC. We then train individual DSBC policies on all 89 tasks[1], including all the tasks we ran *online* DSBC on. On aggregate, this boosted performance from 30% for the base policy to 40% with offline DSBC (whereas zero-shot VLM FRS with the oracle policy achieves 50%, see Fig. 16).

This indicates that, even if no online rollouts are used to validate whether noises from flow reversal map back to good actions, said noises can still effectively be used for DSBC. Practically, this also gives a simple recipe for turning standard demonstration data – with normal robot actions and no noises – *into noise action datasets*. We note that DSRL can also use normal robot prior data, but requires expensive distillation of standard Q-functions into noise-action ones [15].

E.3 DSRL + FRS Experiments

We run two sets of DSRL + FRS experiments on LIBERO-90 for a total of 25 tasks: one on the 15 tasks where FRS boosts performance significantly, as well as one on a harder split of 10 tasks that

[1] We use one the *success-filtered* version of the dataset released by Chen et al. [56] and used for training the LIBERO-90 VLA by Jain et al. [55]. Within, there was one task with 0 successes.

both the base VLA and FRS struggle on, thus allowing us to test how FRS bootstraps RL in regimes where it helps zero-shot performance significantly *and* when it only improves time to first success.

E.3.1 Prefilling DSRL with Multiple FRS Successes

Tasks. We first run DSRL + FRS on the same 15 tasks as in DSBC. However, we do *not* use all successful trajectories: instead, we sample only 20 of the 50 FRS rollouts (with successes and failures proportional to how many there are in the total 50) to run BC loss on and to prefill the replay buffer. We do this over 3 seeds, each randomly sampling (potentially different) rollouts.

Hyperparameters. See Table 1.

DSRL + FRS training. Prefilling replay buffers is an effective way to consume prior data in RL [90]. As FRS produces expert noise actions during zero-shot steering, it allows DSRL to likewise use those trajectories as prior data for policy and critic learning, without any changes – given that DSRL uses SAC [91] to learn the noise policy, it can consume off-policy data.

The only change we make to the actual RL algorithm is adding a BC loss term to actor learning:

$$\mathcal{L}_{\text{DSRL + FRS, actor}} = \mathcal{L}_{\text{DSRL, actor}} + \beta \mathcal{L}_{\text{BC}} \tag{2}$$

This is a popular soft behavior constraint [92, 93], especially when used with SAC. As with DSBC, we use NLL as the BC loss, but find that using MSE loss achieves similar performance, especially in conjunction with a 0-action regularizer. We use BC loss weight $\beta = 1$.

At each learning step, two batches of equal batch size are sampled: one for standard SAC learning, the other for the BC regularization loss (which is drawn exclusively from the successful prefill trajectories).

Our noise policy is parameterized in the same way as the DSBC one: it predicts average noises (which are repeated and padded during inference), and it uses tanh clipping.

Baselines. For the DSRL baseline, we use the exact same hyperparameters as DSRL + FRS. This includes predicting a single noise corresponding to all non-padding actions, then repeating it across the chunk length and adding unit Gaussian padding for the remaining dimensions.

For the residual RL comparison, we follow some prescriptions from PLD [49], as they also apply residual RL to VLAs. We control for policy architecture with respect to DSRL. To determine the residual RL maximum edit distance, we use 0.5 as the tanh clipping coefficient, as that is the distance used in PLD (which the authors mention being sensitive to tuning). However, as we are *not* aiming to distill the residual policy's outputs into the VLA, we simplify our residual RL implementation from PLD by removing the "probing" stage (wherein the residual policy defers to the base VLA for some number of steps), as this mainly serves to help the policy act as an expert from more starting states. Our residual policy edits the full chunk of the base VLA, and is otherwise trained with the same SAC [91] implementation and hyperparameters as our DSRL runs.

Additionally, the data buffers are filled the exact same way as in DSRL (by running online policy rollouts until 1000 frames are collect). Critically, we also do not use PLD's pretraining phase, as doing so requires collecting 50 successful rollouts per task *from the base VLA*. *Given that we are targeting tasks where the base VLA only achieves 0 or 1 successes in 50 rollouts, collecting 50 successes would be impractical.*

E.3.2 Prefilling DSRL with One FRS Success

Tasks. We consider 10 LIBERO-90 tasks that the base VLA get close to 0% average success rate on *and* which zero-shot FRS achieves an average performance of under 10% on (vs. over 30% on the previous 15-task split). The task IDs are: `1, 3, 18, 25, 36, 37, 49, 63, 69, 78`. The DSRL and DSRL + FRS baselines are run for three seeds, RoboMeter only uses one.

KL-based DSRL. In cases with more difficult and low-success tasks, DSRL needs to be able to sample (rarely) more out-of-distribution actions compared to the vanilla clipped DSRL, which can

only sample within $[-1, 1]$ due to a tanh activation. Therefore, we switch to not using the tanh activation after predicting a mean and standard deviation for a Gaussian, allowing the policy to express all means. To prevent the policy from exploiting this, we add a regularization term of 0.1 KL relative to the standard normal distribution $N(0, 1)$ which incentivizes the policy to stay close to the original DSRL distribution. We find that without this change, overall DSRL performance is very low.

Buffer filling. Since DSRL + FRS may have only one or two successes within 50 rollouts on difficult tasks, we instead run zero-shot VLM FRS until we get one success, and then only load that single success into the replay buffer to jump-start reinforcement learning.

Hyperparameters. See Table 1.

Baselines. For robometer, see the dedicated section for VLM reward methods in App. E.4. For DSRL vanilla, we use the same formulation as KL-DSRL without the warmstart.

E.4 VLM Reward Methods

For VLM reward methods, most VLM reward methods also train *directly* on LIBERO, therefore containing privileged information about success trajectories that we do not assume in our DSRL+FRS approach. We compare with Robometer, a work which is trained generally on a broad set of simulation and real-world environments. We modify the base SAC reward to have reward `-(1 - dense-reward)` instead of the default `-1` for each non-terminal step, and `0` in each terminal step, in the same way as [23]. We do not use the transformer-based chunk prediction modification to DSRL stated in the aforementioned paper. We find that our method greatly outperforms Robometer-guided DSRL, possibly because FRS more effectively focuses the policy onto good action modes without relying on the base policy already choosing good actions at a substantial rate.

F Real-World Details and Hyperparameters

We run our real-world experiments on the DROID Franka setup [18], using the open-source $\pi_{0.5}$-DROID VLA [16]. All evaluations use 20 rollouts per task, and all error bars are Wilson confidence intervals.

Tasks. We consider six tasks, shown in Fig. 8. All involve leveraging skills captured within the DROID dataset, albeit in ways or situations that are uncommon or novel.

Zero-shot human FRS. We run zero-shot human-in-the-loop FRS using the interface from App. D.1. The operator attempts 20 trials of each of the six tasks, recording successful ones. Each episode, the human can issue one task per chunk. As we use receding horizon control (wherein a full action chunk is predicted, the first 10 are executed, and then the policy is queried again) and maximum episode length 400, the person has issue only 40 keystrokes per episode at most.

Real-world DSBC. DSBC reuses 10 successes from the DROID zero-shot FRS experiments (of which all tasks achieve). All frames from these tasks are used for supervised learning with DSBC, amounting to at most 400 frames. We parameterize our DSBC policy in the same way as Wagenmaker et al. [15] (albeit without attending into the flow VLA). However, it does *not* predict averaged and repeated noises, instead predicting the full chunk of noises (no padding, as usual). Additionally, we use MSE loss (and thus do not learn variance) for all tasks except the towel hanging task (see below).

Real-world DSRL + FRS. We run a simple form of DSRL + FRS on the towel hanging task (which was empirically the most difficult one). Specifically, we use batched policy gradients, where all successful trajectories' actions have reward 1 and all unsuccessful have -0.1. This requires action probabilities, motivating us to train on this task with NLL. After training the original BC policy, we roll it out 10 times, then update the policy with 2.5k steps on that data. This is repeated once (i.e., two batches of online rollout updates). The final policy is likewise evaluated over 20 rollouts. All hyperparameters are shared with DSBC, albeit with fewer gradient steps.

Hyperparameters. See Table 2.

Table 2: Hyperparameters for real world learning. DSBC and standard BC with a Gaussian policy share hyperparameters. All share the same dual-camera DSRL-style encoder and MLP trunk; they differ in *what* they predict (the VLA's flow reversal noise vs. the robot actions directly) and in the output head (Gaussian regression vs. flow matching). The DSBC head is trained with MSE by default; an NLL (Gaussian, mean + log-std clamped to $[-5, 2]$) variant is used only to initialize RL finetuning.

Hyperparameter	**DSBC**	**Flow-matching Standard BC**
Task setup & I/O		
Base policy	$\pi_{0.5}$ (`pi05_droid_jointpos`)	none (standalone)
Image resize	84×84	84×84
Action chunk length	15	15
Query frequency (open-loop horizon)	10	10
Prediction target	Flow-reversal noise	robot actions
Per-step output dim	15 x 8 (noise)	15 x 8 (action)
Proprioception input	No	Yes
Policy network (shared trunk)		
Image encoder (per camera)	DSRL small CNN, 4 conv, stride $2/1/1/1$	DSRL small CNN, 4 conv, stride $2/1/1/1$
Latent bottleneck (per camera)	Linear→LayerNorm→tanh, dim 50	Linear→LayerNorm→tanh, dim 50
MLP hidden dims	(256, 256, 256)	(256, 256, 256)
Output head	deterministic mean	flow velocity field $v_\theta(x_t, t, \text{obs})$
Timestep embedding	–	sinusoidal (sin/cos), dim 64
Optimization		
BC loss	MSE	conditional flow matching
Training steps	12,500	50,000
Batch size	128	128
Optimizer	Adam	Adam
Learning rate	3×10^{-4}	3×10^{-4}
Flow matching		
Flow-time sampling	–	$\text{Beta}(1.5, 1) \cdot 0.999 + 0.001$
ODE solver (inference)	–	Euler, 10 steps (t=1→0)
Action normalization	none ($\epsilon \sim \mathcal{N}(0, I)$ already)	per-dim z-score (data stats), un-normalized at sampling
Training data		
Trajectories	successful only	successful only
Num. trajectories per task	10	10

Baselines. We compare against the base policy and a standard BC baseline. The latter is trained on the same data as the DSBC policies (i.e., from the steered rollouts), but are trained to output a chunk of *normal robot actions*, not noises.

We initially tried both a simple Gaussian policy and an expressive flow-matching one. They use the same overall architecture as DSBC (of course, with the final layer training on different objectives, e.g., flow matching). While we ensured both implementations were correct by training on a simpler DROID pick-and-place task, both failed at the six tasks that DSBC excelled at. In particular, DSBC's ability to "fall back on" the VLA's action prior made it much more data efficient, allowing it to learn these challenging tasks in just 10 trajectories. However, as flow BC achieved non-zero success rate for the "hang tape on black stand" task (which we use for real-world offline DSBC), **we consider the flow policy the definitive standard BC approach for our real-world experiments.**

Additionally, we found it empirically important to output normalized actions and input proprioceptive states, so we use the same open-source DROID normalization statistics as $\pi_{0.5}$-DROID for this. Both standard BC policies we tried use the same hyperparameters, though we just train for 4x as long for the flow policies (as flow and diffusion tend to converge slower).