STEERING GENERATIVE ROBOT POLICIES WITH OPTIMAL TRANSPORT MAPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative policies have demonstrated significant potential in robotic manipulation. Despite acquiring a rich repertoire of motor skills, they remain brittle under train-test shifts: even mild changes in geometry or context can cause execution failures, limiting scalable deployment in the real world. While retraining or fine-tuning offers a remedy, it often incurs impractical costs and may impair the pre-trained generalist capabilities. To this end, we propose LOTUS, a training-free framework based on Optimal Transport (OT) that adapts the behavior of frozen policies during inference. Following the VLM-guided paradigm, LOTUS first synthesizes stage-wise reward functions to evaluate candidate actions. Unlike previous efforts that use rewards for best-of-N selection or gradient perturbation, we construct virtual trajectories along high-reward directions, which can be viewed as samples from a task-aligned distribution. Our key idea is to project the original policy distribution toward such reward-induced regions while preserving the learned action prior. This can be formulated as an OT problem with motion discrepancy as the transport cost, and we further relax the target marginal to avoid over-steering toward implausible actions. We evaluate LOTUS on both simulated and real-world manipulation tasks, showing substantial improvement against diverse perturbations. The code is provided in the supplementary material.

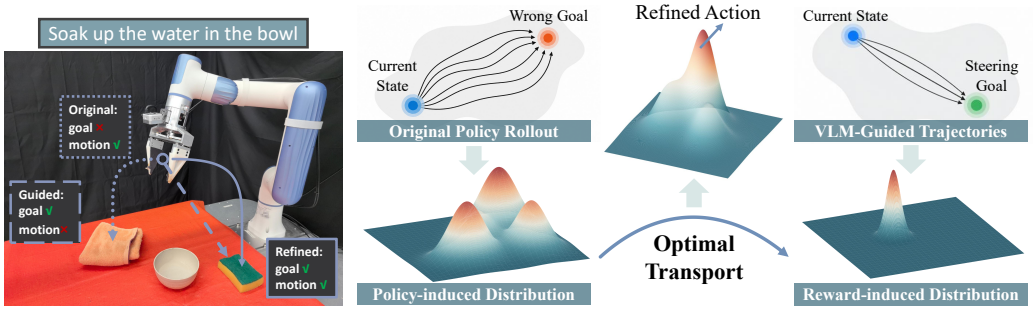

Figure 1: LOTUS formulates inference-time steering as an optimal transport problem that maps the original policy distribution to a reward-induced distribution constructed by external reasoners (e.g., VLMs). It assigns target mass based on reward scores and uses motion discrepancy as the transport cost, yielding refined actions that balance task alignment with motion feasibility.

1 INTRODUCTION

The pursuit of general intelligence has driven progress in Vision-Language-Action (VLA) models, which seek to perceive and interact with the physical world like humans. Generative foundation policies are seen as a key milestone in robot learning due to the broad repertoire of skills across various applications, such as agricultural harvesting (Zhao et al., 2026) and surgical assistance (Ho et al., 2025). Despite learning from large-scale demonstrations (O'Neill et al., 2024), these generative policies remain brittle under out-of-distribution (OOD) observations or task instructions, which are inherent to real-world deployment. To mitigate such issues, one natural direction is to collect task-specific data and retrain the generalist. However, this incurs impractical costs when scaled

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

to open-world demands, and inevitably impairs the generalist capabilities already learned during pretraining. Therefore, it is desirable to address the OOD shifts through a training-free paradigm.

Inference-time policy steering (Nakamoto et al., 2025; Wagenmaker et al., 2025; Li et al., 2026; Liu et al., 2026) offers an elegant solution that expands the training distribution to cover novel variations during deployment. The key intuition is that the pretrained policy provides a rich prior over plausible behaviors that can be reshaped to face OOD shifts. Depending on the use of the reward signal, existing policy steering can be broadly categorized into selection-based and gradient-based methods. Selection-based methods (Nakamoto et al., 2025; Li et al., 2026) generate multiple action proposals and execute those with the highest reward scores. However, under OOD shifts, it is difficult to reach feasible regions by only broadening the action space. Gradient-based methods (Wang et al., 2025; Liu et al., 2026) instead leverage reward gradients to explicitly perturb the denoising trajectory by modifying the predicted noise or velocity field. Despite their effectiveness, such unconstrained perturbations may distort the original action manifold, resulting in unstable or infeasible behaviors.

A question naturally arises: *how to reshape the pretrained distribution toward high-reward regions while preserving its action prior?* To answer this question, this paper presents LOTUS, an inference-time framework that leverages **L**atent-space **O**ptimal **T**ransport maps for diff**U**sion policy **S**teering. As illustrated in Fig. 1, our key idea is to project the original policy distribution into a task-aligned space constrained by the supported action structure. Specifically, inspired by recent advances in the VLM-guided policy steering (Liu et al., 2026), we first use VLMs to interpret OOD inputs, decompose the task into execution stages, and synthesize reward functions to assess how well candidate actions satisfy task requirements. These rewards encode geometric distances from the end-effector to spatial targets. Building on this geometric structure, we can construct virtual trajectories that follow ideal straight-line motions, which can be viewed as samples from a reward-induced action distribution. Then, we leverage OT to map the original distribution toward this reward-induced region in latent denoising space, where the transport cost is defined by motion discrepancy to penalize deviations from the action prior. To strengthen the penalty against implausible motion, we further relax the target marginal via semi-unbalanced OT and allow original policy particles to retain part of the transported mass. The transport plan yields a refined action proposal balancing task alignment with motion feasibility, and can be solved by the rapid Sinkhorn algorithm (Cuturi, 2013).

We comprehensively evaluate LOTUS in both simulation and real-world settings. In simulation, we benchmark on VLABench, Libero-Plus, and Libero-Pro, three popular manipulation suites under diverse train-test shifts. Results show that LOTUS consistently improves frozen VLA policies across all benchmarks, with an absolute improvement of 17% on the challenging Libero-Pro task. Remarkably, LOTUS achieves an average 2.6$\times$ gain across five Dobot Nova2 manipulation tasks, demonstrating its robustness against real-world perturbations during deployment.

2 Related Work

Generative Robot Foundation Policies. Generative policies have recently gained increasing attention for their potential to support scalable robot manipulation across diverse tasks (Bjorck et al., 2025; Wen et al., 2025). By modeling action generation as a conditional denoising process, diffusion policies (Team et al., 2024; Liu et al., 2025a) enable expressive control over long-horizon behaviors. Toward efficient action generation, flow-matching policies (Black et al., 2024; 2025) map noise to action trajectories via learned velocity fields, making them more attractive for real-time robotic control. Building on the perceptual and semantic capabilities of VLMs, the VLA paradigm (O'Neill et al., 2024) further integrates vision-language representations into robot policies and scales policy acquisition through large-scale imitation (Khazatsky et al., 2024). Although capturing a wide range of robotic skills, such foundation models remain brittle under unavoidable variations in real-world deployment: even mild changes in scene geometry or context can substantially degrade performance (Pumacay et al., 2024). A common remedy is to collect target-specific demonstrations and retrain the policy (Chandra et al., 2025; Kim et al., 2025a), but it requires expensive human expertise and may impair the generalist capabilities (Hancock et al., 2026).

These limitations motivate test-time approaches that steer pretrained policies during deployment without parameter updates. In this work, we focus on steering diffusion or flow-matching policies by modifying the generative sampling process to produce refined action trajectories.

Inference-time Steering for Generative Policies. Steering approaches enable generalist policies to make use of external semantic guidance, typically provided by humans (Wang et al., 2025) or VLMs (Liu et al., 2026). Given the accessibility and cost, we focus on VLM-in-the-loop steering, which formulates VLM feedback as a reward function to guide the sampling process toward high-reward action trajectories. Based on the role of the reward signal, existing steering methods mainly fall into selection-based and gradient-based strategies. Selection-based methods (Nakamoto et al., 2025; Li et al., 2026) treat reward models as evaluators that rank sampled candidates and select the most preferred actions to execute. However, for challenging perturbations or novel scenarios (Zhou et al., 2025), simply broadening the action search is insufficient to discover feasible behaviors. Gradient-based methods (Wang et al., 2025; Liu et al., 2026) act more directly on the generation dynamics by using reward gradients to perturb the denoising process. However, unconstrained perturbations may drift away from the learned action manifold and produce unstable or infeasible behaviors (Wu et al., 2026). In addition, some recent works introduce world models (Qi et al., 2026) or internal representations (Häon et al., 2025) to guide policy generation, highlighting the broader potential for improving robotic policies during inference. In contrast, our work pioneers a motion-constrained steering paradigm that adaptively reconciles the original and task-aligned action distributions, enabling more stable and safe steering under practical deployment variations.

3 Preliminaries and Problem Setting

3.1 Diffusion and flow policies.

Denoising generative models learn to transform a noise distribution into complex action distributions, and have proven effective for real-world robotic tasks involving diverse behaviors (Chi et al., 2025; Black et al., 2024). At environment step t, let o_t denote the current observation, ℓ the language instruction, and $a^0_{t:t+H}$ a clean action chunk with horizon H. The diffusion policy initializes $a^K_{t:t+H} \sim \mathcal{N}(0, I)$ and transforms it into $a^0_{t:t+H}$ through K reverse denoising steps. Formally, the update from step k to $k-1$ is (Ho et al., 2020)

$$a^{k-1}_{t:t+H} = \frac{1}{\sqrt{\alpha_k}} \left(a^k_{t:t+H} - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}} \epsilon_\theta(a^k_{t:t+H}, o_t, \ell, k) \right) + \sigma_k z,\ z \sim \mathcal{N}(0, I), \tag{1}$$

where ϵ_θ predicts the noise conditioned on (o_t, ℓ), α_k, $\bar{\alpha}_k$, and σ_k are noise schedule coefficients.

Beyond discrete denoising formulations, flow policies parameterize a conditional velocity field that defines a deterministic transport from Gaussian noise to the action distribution (Lipman et al., 2023; Black et al., 2024). Let $\tau \in [0, 1]$ denote flow time, where $a^1_{t:t+H} \sim \mathcal{N}(0, I)$ is the initial noise and $a^0_{t:t+H}$ is a clean action chunk. The flow evolves according to

$$\frac{\mathrm{d}a^\tau_{t:t+H}}{\mathrm{d}\tau} = v_\theta(a^\tau_{t:t+H}, \tau \mid o_t, \ell). \tag{2}$$

Solving this ODE backward from $\tau = 1$ to $\tau = 0$ transforms the initial noise into a clean action proposal $a^0_{t:t+H}$. Throughout this work, we treat diffusion and flow-based policies within a unified denoising generative framework (Kingma & Gao, 2023) for test-time policy steering.

3.2 Problem Formulation

Given a VLA policy π_θ learned from an expert demonstration dataset $\mathcal{D}_{\text{train}} = \{(o_i, \ell_i, a_{i:i+H})\}_{i=1}^N$, the policy can generate action chunks conditioned on the current observation and instruction, i.e., $a_{t:t+H} \sim \pi_\theta(\cdot \mid o_t, \ell)$. However, at real-world deployment, the policy may encounter contexts $(o_t, \ell)_{\text{OOD}} \notin \mathcal{D}_{\text{train}}$ that lie outside the support of the training distribution, e.g., unseen visual backgrounds, novel object layouts, or semantically different task instructions. In such OOD scenarios, the generated action often fails to satisfy the spatial or semantic requirements (Pumacay et al., 2024; Zhou et al., 2025), even if the underlying manipulation skill is already captured in the policy.

We study test-time steering of diffusion policies under out-of-distribution conditions. The core of policy steering is to guide the model to generate action proposals that maximize a given reward:

$$a^*_{t:t+H} = \arg\max_{a_{t:t+H}} R(a_{t:t+H}, (o_t, \ell)_{\text{OOD}}), \tag{3}$$

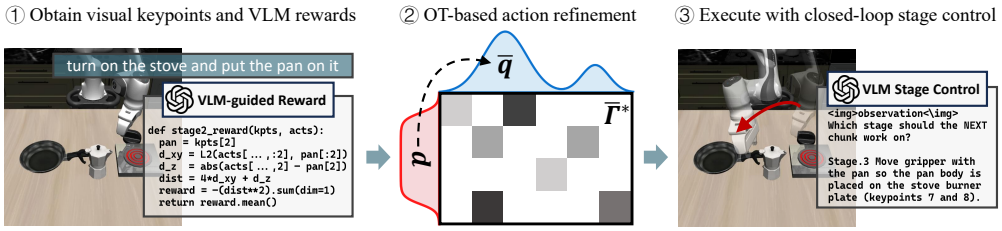

Figure 2: **LOTUS overview.** (1) Vision models ground keypoints, while a VLM decomposes the task and synthesizes stage-wise rewards. (2) Latent-space OT transports the policy-induced distribution toward the reward-induced one. (3) The transport plan reshapes the latent samples, which are then denoised into executable actions, with VLM feedback controlling the stage transitions.

where $R(\cdot)$ is a task-specific reward function that measures how well $a_{t:t+H}$ facilitates successful task execution. Inspired by recent advances in cross-modal reasoning, a common paradigm in policy steering (Liu et al., 2026; Li et al., 2026) first grounds the target object keypoints (denoted by $\mathcal{P} = \{p_i\}_{i=1}^{n}$, where $p_i \in \mathbb{R}^3$) by vision foundation models, and then uses VLMs to synthesize stage-aware, programmatic reward functions. Following this paradigm, we decompose each task into a sequence of stages, and for each stage s, the VLM produces a reward function $R_s(a^k_{t:t+H}, \mathcal{P})$ that evaluates the action proposal with a scalar score, e.g., geometric distances between the end-effector and the grounded spatial keypoints. The details are provided in App. A.

Given such a reward function, existing policy-steering methods mainly follow two routes: 1) selection-based (Li et al., 2026), which samples multiple action proposals and selects high-scoring ones for execution; 2) gradient-based (Liu et al., 2026), which uses the reward gradient $g = \nabla_{a^k_{t:t+H}} R_s(a^k_{t:t+H}, \mathcal{P})$ to perturb the predicted noise or velocity field toward higher-reward action proposals. However, under OOD conditions, selection-based methods struggle to obtain feasible actions by only expanding the search space. While gradient-based methods show promise for adapting to novel scenarios, directly injecting reward gradients into the denoising dynamics may disrupt the policy's learned action manifold, leading to unstable and suboptimal solutions.

4 Methodology

To address the above limitations, we propose LOTUS, an inference-time steering framework that seeks the high-reward policy while preserving the learned behavioral prior. In the following, we first show how policy steering can be viewed as a distributional transport problem in Sec. 4.1. Then, we elaborate on solving refined manipulation policies through optimal transport in Sec. 4.2, and present the robust closed-loop execution strategy in Sec. 4.3. The overview of LOTUS is shown in Fig. 2.

4.1 Policy Steering as Distributional Transport.

At each environment timestep t, given a well-trained policy π^*, LOTUS begins by sampling a batch of action proposals conditioned on the current observation and instruction:

$$a_{t:t+H}[i] \sim \pi^*(\cdot \mid o_t, \ell), \quad \mathcal{X}_t = \{a_{t:t+H}[i]\}_{i=1}^{N}. \tag{4}$$

These proposals define an empirical action distribution that inherits motion-level advantages from the base policy, such as trajectory smoothness and continuity. However, under OOD conditions, such plausible actions may often target incorrect objects, poses, or spatial relations.

To bridge this gap, policy steering induces a task-compliant distribution through reward functions that drive the end-effector toward goal locations. However, this poses a key challenge: **how to make a trade-off between policy-induced and reward-induced distributions**. For example, selection-based methods search for the overlap between the two distributions, which may become sparse under OOD shifts. While gradient-based methods directly push the policy distribution toward reward-induced regions, such unconstrained displacement may disrupt the supported action manifold.

To this end, we frame this trade-off as a distributional transport problem that seeks a motion-constrained mapping from original distributions to reward-induced distributions. Given the stage

keypoints $\mathcal{P}$, the ideal trajectory (i.e., highest reward) should move the end-effector along the straight line toward the target position. Thus, the reward-induced action proposals can be defined as:

$$\tilde{a}_{t:t+H}[j] = \mathcal{G}\left(a_{t:t+H}[0], e_t, \mathcal{P}, \eta_j\right), \quad \mathcal{Y}_t = \{\tilde{a}_{t:t+H}[j]\}_{j=1}^{M}, \tag{5}$$

where e_t is the current end-effector position, $\mathcal{G}$ denotes the virtual-action constructor, and η_j controls scale and vertical offsets for diversity. Specifically, $\mathcal{G}$ sets the translational motion to follow the straight-line direction from e_t to the centroid of $\mathcal{P}$ while copying all remaining dimensions from $a_{t:t+H}[0]$, and details are provided in App. B. To keep the transported result within the original action manifold, we propose to perform distributional transport in latent denoising space rather than the action space. Therefore, we run the forward noising process to bring these reward-induced proposals to a shared intermediate denoising time τ_s (taking diffusion models as an example):

$$\tilde{a}^{\tau_s}_{t:t+H}[j] = \sqrt{\bar{\alpha}_{\tau_s}}\tilde{a}_{t:t+H}[j] + \sqrt{1-\bar{\alpha}_{\tau_s}}\epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, I), \quad \mathcal{Y}^{\tau_s}_t = \{\tilde{a}^{\tau_s}_{t:t+H}[j]\}_{j=1}^{M}. \tag{6}$$

Similarly, we obtain the policy-induced set $\mathcal{X}^{\tau_s}_t$ by running the frozen policy from initial noise to the same intermediate denoising time τ_s. For convenience, we denote these latent particles as $\mathcal{X}^{\tau_s}_t = \{x_i\}_{i=1}^{N}$ and $\mathcal{Y}^{\tau_s}_t = \{y_j\}_{j=1}^{M}$, with their reward scores denoted by $R(x_i)$ and $R(y_j)$, respectively.

4.2 Refining Action Distributions with Optimal Transport.

Intuitively, the reward-induced distribution specifies where the original particles should move to facilitate task execution, while deviations from the policy-supported action structure should be penalized to preserve motion plausibility. This objective can be naturally formalized as an Optimal Transport (OT) problem for coupling two empirical distributions. Formally, given policy-induced samples $\mathcal{X}^{\tau_s}_t$ and reward-induced samples $\mathcal{Y}^{\tau_s}_t$, let $\boldsymbol{p} = \sum_{i=1}^{N} p_i \delta_{x_i}$ and $\boldsymbol{q} = \sum_{j=1}^{M} q_j \delta_{y_j}$ denote the corresponding empirical distributions, where δ is the Dirac measure, p_i and q_j are the mass belonging to the probability simplex. We define the ground cost matrix $\boldsymbol{C} \in \mathbb{R}^{N\times M}$ with $\boldsymbol{C}_{i,j} = c(x_i, y_j)$, where $c(\cdot,\cdot)$ is the cost function that measures the motion discrepancy between two intermediate particles. Optimal transport aims to find a transport plan $\boldsymbol{\Gamma} \in \mathbb{R}^{N\times M}$ that maps the source distribution $\boldsymbol{p}$ to the target distribution $\boldsymbol{q}$ while minimizing the total transport cost:

$$\min_{\boldsymbol{\Gamma}\geq 0} \langle \boldsymbol{\Gamma}, \boldsymbol{C}\rangle_F, \quad \text{s.t.} \quad \boldsymbol{\Gamma}\mathbb{1}_M = \boldsymbol{p}, \quad \boldsymbol{\Gamma}^\top \mathbb{1}_N = \boldsymbol{q}. \tag{7}$$

where $\langle\cdot,\cdot\rangle_F$ is the Frobenius dot-product and $\mathbb{1}_d$ denotes a d-dimensional all-ones vector.

However, the standard OT (Eq. 7) enforces strict marginal constraints, requiring the mass assigned from $\boldsymbol{p}$ to exactly match the prescribed mass of $\boldsymbol{q}$. As a result, reward-induced particles with high cost can still receive mass, weakening the cost penalty against implausible motion. To address this, we adopt a target-relaxed OT formulation that adaptively reduces the mass assigned to reward-induced particles. First, we allow the original particles to receive part of the transported mass by expanding the target support, i.e., $\bar{\mathcal{Y}}^{\tau_s}_t = \mathcal{X}^{\tau_s}_t \cup \mathcal{Y}^{\tau_s}_t = \{\bar{y}_j\}_{j=1}^{N+M}$. We then define the expanded target marginal $\bar{\boldsymbol{q}}$ from the corresponding reward scores, i.e.,

$$\bar{\boldsymbol{q}} = (\bar{q}_1, \ldots, \bar{q}_{N+M}), \quad \bar{q}_j = \frac{\exp\left(\beta R(\bar{y}_j)\right)}{\sum_{k=1}^{N+M}\exp\left(\beta R(\bar{y}_k)\right)}, \tag{8}$$

where β is a temperature parameter controlling the sharpness of $\bar{\boldsymbol{q}}$. Eq. 8 encourages high-reward particles to receive more mass, while enabling satisfied original particles to be retained instead of forcing steering. Second, as the reward function is synthesized from VLM guidance and may contain potential noise (e.g., ambiguous keypoint grounding), strictly matching $\bar{\boldsymbol{q}}$ can force OT to couple wrong targets. We thus employ a semi-unbalanced OT formulation (Liu et al., 2025b) that mitigates assignments to outliers by relaxing the target marginal constraint, i.e.,

$$\bar{\boldsymbol{\Gamma}}^{\star} = \arg\min_{\bar{\boldsymbol{\Gamma}}\geq 0} \left\langle \bar{\boldsymbol{\Gamma}}, \bar{\boldsymbol{C}}\right\rangle_F + \lambda\,\mathrm{KL}\left(\bar{\boldsymbol{\Gamma}}^\top \mathbb{1}_N \,\middle\|\, \bar{\boldsymbol{q}}\right) - \varepsilon H(\bar{\boldsymbol{\Gamma}}), \quad \text{s.t.} \quad \bar{\boldsymbol{\Gamma}}\mathbb{1}_{N+M} = \boldsymbol{p}. \tag{9}$$

Here, we define $\boldsymbol{p}$ as a uniform distribution as the source particles are randomly sampled; $\bar{\boldsymbol{C}} \in \mathbb{R}^{N\times(N+M)}$ uses cosine distance as cost function to measure the motion discrepancy between two latent particles; $\lambda > 0$ is a hyperparameter to relax the target marginal constraint by KL divergence; $\varepsilon > 0$ is a regularization parameter for the entropic term $H(\bar{\boldsymbol{\Gamma}}) = -\sum_{i=1}^{N}\sum_{j=1}^{N+M}\bar{\boldsymbol{\Gamma}}_{ij}\log\bar{\boldsymbol{\Gamma}}_{ij}$,

enabling the transport plan to be solved by the computationally cheaper Sinkhorn-Knopp algorithm (Cuturi, 2013), and the detailed solution is presented in App. C.

For each original particle, the optimal transport plan $\bar{\mathbf{\Gamma}}^\star$ identifies a projection direction that balances reward attraction and motion plausibility. Formally, we use the first source particle as an anchor to compute the refined latent state, and obtain the final executable action chunk by continuing the remaining denoising process $\Phi_\theta^{\tau_s \to 0}$ from time τ_s to 0:

$$a^*_{t:t+H} = \Phi_\theta^{\tau_s \to 0}\left(\hat{x}^{\tau_s}; o_t, \ell\right), \quad \text{where} \quad \hat{x}^{\tau_s} = \frac{\sum_{j=1}^{N+M} \bar{\mathbf{\Gamma}}^\star_{1j} \bar{y}_j}{\sum_{j=1}^{N+M} \bar{\mathbf{\Gamma}}^\star_{1j}}. \tag{10}$$

4.3 Intra-stage and Inter-stage Robust Control

In practice, we further introduce steering adjustment strategies for both intra-stage and inter-stage to enable robust closed-loop execution. Specifically, at the beginning of each stage, strong steering should be applied to dominate the motion direction. While near stage completion, the original policy should take over for fine-grained manipulation. To this end, we propose an adaptive target marginal to control the steering strength. At stage s, let R_s^0 denote the guidance reward of the first action chunk generated by the frozen policy, and let R_s^{t} denote the reward of the action chunk generated at chunk index t. The adaptive target marginal is computed by replacing β in Eq. 8 with:

$$\beta_t = \beta \cdot \mathrm{sigmoid}\left(k\left[b - \left(1 - \frac{R_s^{\mathrm{t}}}{R_s^0}\right)\right]\right), \tag{11}$$

where $k > 0$ and $b \in [0, 1]$ control the decay sharpness and progress threshold of steering reduction, respectively. The term $1 - \frac{R_s^{\mathrm{t}}}{R_s^0}$ estimates the stage progress, and β_t decays once it exceeds b.

To ensure robust transitions between stages, we follow the Schmitt-trigger-based switching strategy proposed by VLS (Liu et al., 2026). Specifically, let R_{high} and R_{low} denote the upper and lower reward thresholds. When R_s^t crosses either threshold, the VLM is queried to interpret the execution outcome and decide whether to advance to the next stage or retain the current one.

5 Experiments

5.1 Experimental Setup

Simulation. We use Libero-Pro (Zhou et al., 2025), Libero-Plus (Fei et al., 2025), and VLABench (Zhang et al., 2025) for simulated evaluations. To systematically evaluate LOTUS under OOD conditions, we consider controlled shifts along two dimensions: 1) **object-centric shifts**, where the perturbations originate from the external manipulation context, including changing task targets, resetting object positions, and introducing unseen categories. Specifically, we use VLA policies fine-tuned on the Libero (Liu et al., 2023) dataset and test them on Libero-Pro under task perturbation and position perturbation settings. In addition, we use VLA policies fine-tuned on VLABench and evaluate them on the Cross-Category track with unseen objects and categories. 2) **observation-centric shifts**, where the perturbations originate from the robot-side context, including changing camera viewpoints and varying robot initial states. Specifically, we test Libero fine-tuned policies on Libero-Plus under camera perturbation and robot perturbation settings. For test-time steering, we choose the representative policies π_0 (Black et al., 2024) and $\pi_{0.5}$ (Black et al., 2025) as base models for LOTUS. Detailed steering settings are presented in App. D.

Real world. We evaluate LOTUS on a 6-DoF Dobot Nova2 robotic arm equipped with a 1-DoF gripper and two Intel RealSense depth cameras, including a wrist-mounted D405 and a front-facing D455. As shown in Fig. 3, we consider five manipulation tasks. For each task, we use a $\pi_{0.5}$ policy fine-tuned on task-specific demonstrations and evaluate it under corresponding OOD variants. Specifically, we replace the original button with buttons of different colors in *Push Button*, replace the plate with a box in *Put Grape*, add distractor flowers in *Insert Flower*, specify an explicit placement order in *Tidy Desk*, and reverse the stacking relation in *Stack Bowl*.

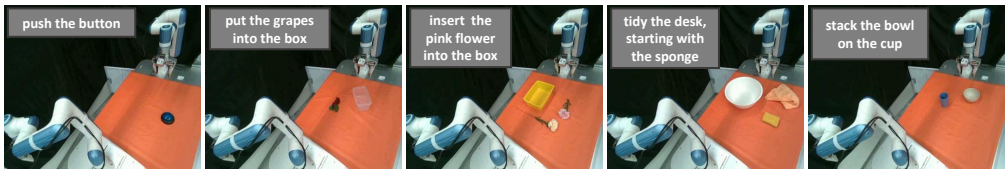

Figure 3: **Real-world evaluation tasks.** We evaluate LOTUS on 5 different tasks in the real world with different OOD suites to demonstrate the effectiveness of steering.

Table 1: **Libero-Pro results.** We compare several VLA baselines and apply LOTUS to a frozen $\pi_{0.5}$ policy. Evaluations are conducted under task and position perturbations across the four LIBERO suites: Goal, Spatial, 10 (Long-Horizon), and Object, with each suite containing 10 tasks. For each task, we run 20 episodes and report the average success rate (%).

Method	Task Perturbation					Position Perturbation					Overall
	Goal	Spatial	10	Object	Avg.	Goal	Spatial	10	Object	Avg.	Avg.
OpenVLA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\pi_{0.5}$	0.00	1.00	1.00	1.00	0.75	38.00	20.00	8.00	17.00	20.75	10.75
$\pi_{0.5}$ (LeRobot)	12.00	48.50	21.50	10.50	23.13	29.00	41.00	11.00	16.00	24.25	23.69
$\pi_{0.5}$ (LeRobot) + LOTUS	**31.00**	**56.50**	**37.50**	**43.50**	**42.13**	**36.50**	**58.00**	**15.00**	**48.50**	**39.50**	**40.81**

Table 2: **Libero-Plus results.** We compare several VLA baselines and apply LOTUS to a frozen π_0 policy. Evaluations are conducted under camera and robot perturbations across the four LIBERO suites: Goal, Spatial, 10 (Long-Horizon), and Object, with each suite containing 10 tasks. For each task, we sample 10 episodes following the LIBERO-PLUS difficulty settings, except for Spatial under camera perturbation, where one semantic task contains only 6 official variants and the remaining 4 are redistributed across other tasks. We report the average success rate (%).

Method	Camera Perturbation					Robot Perturbation					Overall
	Goal	Spatial	10	Object	Avg.	Goal	Spatial	10	Object	Avg.	Avg.
OpenVLA	2.00	0.00	0.00	1.00	0.75	2.00	3.00	3.00	4.00	3.00	1.88
π_0	12.00	19.00	13.00	29.00	18.25	5.00	7.00	6.00	9.00	6.75	12.50
π_0 (LeRobot)	57.00	73.00	41.00	73.00	61.00	27.00	32.00	16.00	18.00	23.25	42.13
π_0 (LeRobot) + LOTUS	**64.00**	**74.00**	**43.00**	**77.00**	**64.50**	**38.00**	**45.00**	**19.00**	**38.00**	**35.00**	**49.75**

5.2 Main Results

Compared to Base VLA Policies. We first study whether LOTUS can adapt frozen policies to OOD scenarios in simulated environments. Following the evaluation protocol in VLS, we compare against four representative VLA baselines spanning different model families: OpenVLA (Kim et al., 2025b), π_0 (Black et al., 2024), $\pi_{0.5}$ (Black et al., 2025), and the LeRobot-finetuned $\pi_{0.5}$ variant. All models are evaluated without any fine-tuning on the test-time OOD perturbations. The results on the Libero-Pro and Libero-Plus are summarized in Table 1 and Table 2, respectively. From the results, one could have the following observations and conclusions. 1) Despite strong pretrained capabilities in visual perception and language understanding, the base VLA policies struggle to adapt to external or internal shifts at test time. In contrast, LOTUS could realign frozen policy behavior with task objectives, thus consistently outperforming the base VLA baselines across various distribution shifts. 2) LOTUS shows greater robustness against more severe external shifts (e.g., Task and Position), where even the strongest VLA policy degrades significantly under these challenging settings. Compared to visual perception perturbations (Camera), the spatial distance-based reward could provide more direct rectification for object-centric shifts. 3) LOTUS demonstrates strong generalization across different base models, perturbation types, and task suites. Specifically, it improves $\pi_{0.5}$ on Libero-Pro and π_0 on Libero-Plus by an average of 17.12% and 7.62%, respectively.

We next validate LOTUS on a Dobot Nova2 robot across five real-world manipulation tasks. As shown in Table 3, VLA policies are vulnerable to train-test shifts: even simple color changes can substantially degrade the performance of well-trained models. By steering the policy toward specified target locations, LOTUS increases the overall success rate by approximately 2.6× across all five tasks. However, LOTUS fails on the Stack Bowl task, which requires precise alignment of the

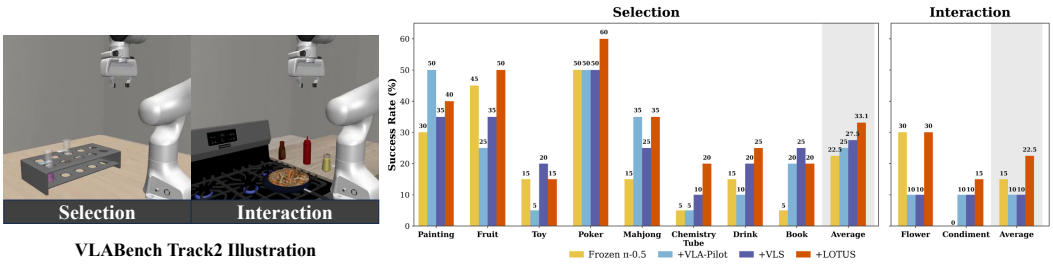

Figure 4: **Steering methods comparison on VLABench.** We compare the official fintuned $\pi_{0.5}$ with VLA-Pilot, VLS, and LOTUS on the Track 2 setting. We evaluate 20 episodes per task and report the average success rate, except for the insert flower task that contains only 10 official episodes.

bowl base with the narrow cup rim, indicating that geometric guidance cannot recover fine-grained motions absent from the learned action prior. App. E shows the visualization of execution processes.

Table 3: Real-world success rates (%) over 20 episodes per task across five OOD suites.

Method	Push Button	Put Grape	Insert Flower	Tidy Desk	Stack Bowl	Overall
Base	20	50	25	20	0	23
LOTUS	90	75	65	65	0	59

Compared to Existing Steering Methods. We compare with VLA-Pilot (Li et al., 2026) and VLS (Liu et al., 2026), two representative methods from selection-based and gradient-based steering paradigms, respectively. Specifically, we use the official VLABench fine-tuned $\pi_{0.5}$ as the base policy and evaluate on the Cross-Category Track (Track 2), which comprises 10 task categories with unseen object categories and instances. Based on the required manipulation, we further divide these tasks into two groups: selection (e.g., select mahjong) and interaction (e.g., add condiment). As illustrated in Fig. 4, although existing steering methods improve the base policy on average, their performance varies substantially across tasks, with both VLA-Pilot and VLS even degrading performance on the insert flower task. This is mainly due to the large variation across VLABench tasks in target geometry and manipulation objectives, which makes existing methods brittle to the steering strength and prevents a single setting from consistently improving all tasks. In contrast, LOTUS adaptively reconciles the original and task-aligned action distributions, enabling robust steering across all tasks using a unified steering strength. Remarkably, LOTUS achieves superior performance in adapting to unseen variations, with relative improvements of 47.1% and 50.0% over the base policy on selection and interaction tasks, respectively. Moreover, we compare LOTUS with VLS on Libero-Pro, with the results reported in App. F.

5.3 Ablation and Analytical Study

Impact of Each Component. To study the influence of specific components in LOTUS, we ablate the contributions of three key components, i.e., target-relaxed OT, reward-induced virtual particles, and adaptive steering strength. Fig. 5 (right) reports the performance on Libero-Pro and VLABench Track 2, and supports the following conclusions. First, the full LOTUS achieves the best performance on both benchmarks, showing that all three components are essential to improve the robustness against OOD shifts. Second, removing virtual particles reduces LOTUS to selection-based steering over policy-induced actions, which struggles under OOD conditions, with performance on Libero-Pro only on par with the base policy. Third, using standard OT with strict constraints may cause severe performance degradation, e.g., a 19.5% drop on VLABench. This indicates that forcing the policy to follow external guidance can be counterproductive, while our target-relaxed OT mitigates this by adaptively balancing the policy prior and external guidance.

Parameter Analysis. We next investigate the sensitivity of key hyperparameters in the OT formulation, i.e., the number of policy-induced samples N, the number of reward-induced samples M, the target marginal sharpness parameter β, and the target marginal relaxation weight λ. As shown in Fig. 5 (left), we could observe that LOTUS is robust to the numbers of policy-induced and reward-induced samples, obtaining stable improvements over relatively broad ranges, e.g., $N \in [8, 32]$ and $M \in [8, 16]$, with absolute gains of $17.0\% \sim 19.5\%$ over the base policy. In addition, LOTUS

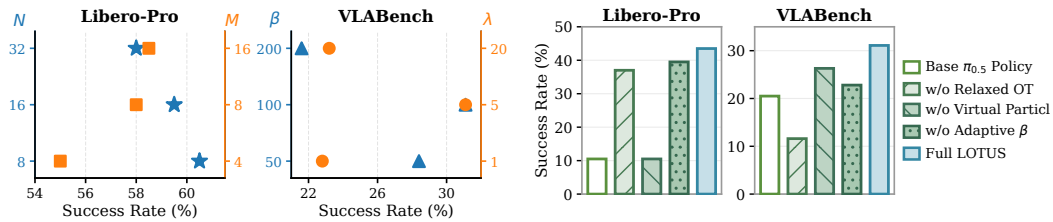

Figure 5: **Parameter analysis and component ablation of LOTUS. Left:** We analyze the effects of the numbers of policy-induced and reward-induced samples (N and M) under Position Perturbation in Libero-Pro Spatial, and the effects of the target-marginal parameters β and λ on VLABench Track 2. **Right:** We ablate three key components under Task Perturbation in Libero-Pro Object and on VLABench Track 2: target-relaxed OT, reward-induced virtual particles, and adaptive β.

is more sensitive to the target marginal parameters β and λ. To be specific, a larger λ strengthens the marginal constraint to be strictly matched, while a smaller value reduces the influence of reward guidance. Similarly, a larger β sharpens the reward-induced distribution, whereas a smaller β makes it more uniform. Empirically, we find that $\beta = 100$ achieves a good trade-off across all experiments.

Efficiency Analysis. We further evaluate the computational overhead of inference-time policy steering. In detail, we compare LOTUS with VLS under two OOD settings in Libero-Pro. As shown in Table 4, although LOTUS requires more steering overhead for computing refined action chunks, these task-aligned actions better preserve the policy prior and remain more feasible, resulting in higher success rates and fewer average execution steps. Additionally, the main overhead of VLM-guided steering comes from communication with the VLM, where latency can fluctuate due to server load or network conditions. To further improve steering efficiency, a promising direction is to distill a smaller, locally deployable VLM for reward construction and stage assessment.

Table 4: Efficiency comparison on Libero-Pro under Spatial suite with position perturbation and Long suite with task perturbation, where each suite contains 50 episodes across 10 tasks.

Method	Suite	Success rate (%)	Episode steps	Steering overhead (s)	Average time (s)
VLS	10-task	24.0	487.3	56.0	328.4
	spatial-swap	48.0	206.5	24.2	116.5
LOTUS	10-task	32.0	455.3	76.7	319.0
	spatial-swap	58.0	202.6	31.4	119.8

Visualization Result. To intuitively show the behavior of the refined actions, we visualize the similarity between the refined and original policy action chunks, and provide a case study of the refinement process in LOTUS. From Fig. 6, one could see that LOTUS exhibits a more pronounced peak in the high-similarity region than VLS, indicating that the refined actions better preserve the original policy prior. The higher task success rate further demonstrates motion-constrained steering's effectiveness in maintaining feasible behaviors. Moreover, App. G visualizes the steering directions over a case episode, showing how LOTUS guides the end-effector toward stage-specific targets.

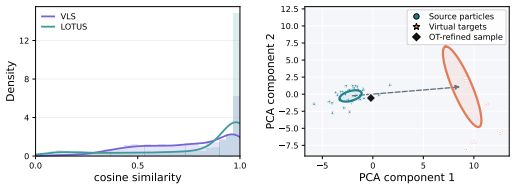

Figure 6: Visualization of action similarity (left) and the refinement process (right) on the Long suite with task perturbation of Libero-Pro.

6 Conclusion and Future Work

This work offers a new perspective on inference-time steering for generative VLA policies. Instead of selecting high-reward samples or injecting reward gradients into the denoising process, we seek an optimal transport plan between the original policy distribution and a reward-induced one. By minimizing a transport cost that captures motion discrepancy, the resulting actions align with task objectives while preserving the learned action prior. Extensive experiments in both simulation and the real world verify the effectiveness of our method under diverse OOD scenarios. In the future, we plan to extend LOTUS toward test-time self-training, where refined actions can supervise continual policy updates, enabling generalist policies to retain persistent adaptations to specific new scenarios.

AI USE STATEMENT

In this work, we use large language models (GPT-5.6 Sol and GPT-6 Astra) to assist with drafting and polishing this manuscript, including correcting grammar, improving clarity, and restructuring sentences for readability. We also use these tools to refine mathematical descriptions and organize experimental tables. Our inference-time steering framework follows a VLM-in-the-loop paradigm. Specifically, GPT-5.5 decomposes manipulation tasks into stages and constructs the corresponding stage-wise reward functions to guide VLA policies. Compared with human-in-the-loop guidance, this paradigm reduces the need for expensive human intervention, making task-specific guidance easier to scale across diverse tasks.

All outputs generated by LLMs were critically evaluated and edited by the authors, and no content was used without verification. The authors retain full responsibility for the manuscript, including its text, mathematical claims, and experimental interpretations.

REPRODUCIBILITY STATEMENT

We provide the implementation code, experiment configurations, and execution scripts in the supplementary materials to facilitate reproduction of our results.

REFERENCES

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Hao-Huan Wang, and Ury Zhilinsky. $\pi 0$: A vision-language-action flow model for general robot control. *ArXiv*, abs/2410.24164, 2024. URL https://api.semanticscholar.org/CorpusID:273811174.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. Diwa: Diffusion policy adaptation with world models. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3378–3400. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/chandra25a.html.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Asher Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. In *International Conference on Learning Representations*, volume 2026, pp. 75333–75360, 2026.

Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2743–2762. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/haon25a.html.

Chonlam Ho, Jianshu Hu, Lei Song, Hesheng Wang, Qi Dou, and Yutong Ban. Diffusion stabilizer policy for automated surgical robot manipulations. *arXiv preprint arXiv:2503.01252*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Diederik P. Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 65484–65516, 2023.

Zhuo Li, Junjia Liu, Zhipeng Dong, Tao Teng, Quentin Rouxel, Darwin Caldwell, and Fei Chen. Towards deploying vla without fine-tuning: Plug-and-play inference-time vla policy steering via embodied evolutionary diffusion. *IEEE Robotics and Automation Letters*, 11(5):6234–6241, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

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

Shuo Liu, Ishneet Sukhvinder Singh, Yiqing Xu, Jiafei Duan, and Ranjay Krishna. VLS: Steering pretrained robot policies via vision–language models. *arXiv preprint arXiv:2602.03973*, 2026.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025a.

Weiming Liu, Xinting Liao, Jun Dan, Fan Wang, Hua Yu, Junhao Dong, Shunjie Dong, Lianyong Qi, and Yew Soon Ong. Solving discrete (semi) unbalanced optimal transport with equivalent transformation mechanism and kkt-multiplier regularization. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 105330–105365. Curran Associates, Inc., 2025b. doi: 10.52202/085713-3519. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/97a36b5f6db9e6703e4154cf0c582214-Paper-Conference.pdf.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4996–5013. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/nakamoto25a.html.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The COLOSSEUM: A benchmark for evaluating generalization for robotic manipulation. In *Robotics: Science and Systems*, 2024.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2026.

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

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/wagenmaker25a.html.

Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Pérez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15626–15633. IEEE, 2025.

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. In *ICML*, 2025.

Lingxuan Wu, Zijian Zhu, Lizhong Wang, Chengyang Ying, Huayu Chen, Xiao Yang, Fangming Liu, and Jun Zhu. Pact: Self-evolving physical safety alignment for diffusion policies in embodied manipulation. *arXiv preprint arXiv:2606.08414*, 2026.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152. IEEE, 2025.

Ziyang Zhao, Shuheng Wang, Zhonghua Miao, and Ya Xiong. Harvestflex: Strawberry harvesting via vision-language-action policy adaptation in the wild. *arXiv preprint arXiv:2603.05982*, 2026.

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

A KEYPOINT GROUNDING AND GEOMETRIC REWARD

Following VLS (Liu et al., 2026), LOTUS converts the RGB-D observation into a set of spatial keypoints to construct reward functions. Specifically, we first identify objects relevant to the instruction and obtain their masks. We then extract dense DINO features within these masks and combine them with 3D coordinates recovered from depth. Next, we cluster the resulting visual and geometric features to obtain representative 3D keypoints for each object. We then query a VLM with the instruction and grounded keypoints to generate stage-wise geometric rewards.

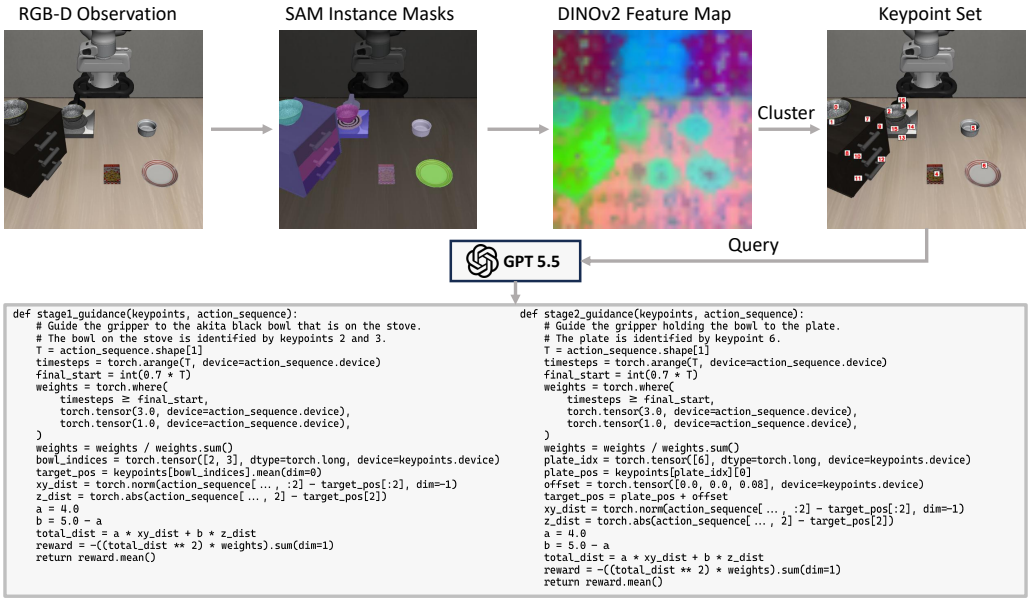

Figure 7: Illustration of the keypoint grounding and geometric reward construction pipeline. Visual grounding identifies task-relevant object regions, dense visual features and 3D geometry are clustered into a set of spatial keypoints. After that, a VLM uses the instruction and the indexed keypoint set to generate stage-wise geometric reward programs.

As illustrated in Fig. 7, the task is decomposed into two stages with corresponding reward functions. For each stage, the VLM selects keypoints that encode the relevant object or spatial relationship and generates a programmatic reward function. These functions measure geometric properties of a proposed end-effector trajectory, such as its distance to a target point. For the example in Fig. 7, the first stage uses the keypoints on the bowl to guide the grasping motion, while the second stage uses the keypoint on the plate to guide placement. Based on these reward functions, LOTUS evaluates target particles and forms the reward-induced distribution for latent-space optimal transport.

B VIRTUAL-ACTION CONSTRUCTION

We construct virtual proposals using straight-line motion from the current end-effector position e_t toward the centroid of the stage keypoints. Taking the widely used $\pi_{0.5}$ as an example, we consider a 7-dimensional action whose first three dimensions represent translation and whose remaining four encode orientation and gripper control. For each proposal j, we construct a complete virtual action chunk as

$$\tilde{a}_{t:t+H}[j] = \mathcal{G}\left(a_{t:t+H}[0], e_t, \mathcal{P}, \eta_j\right) = \left[\, \tilde{A}_j^{XYZ},\ a_{t:t+H}[0]_{:,4:D} \,\right], \tag{12}$$

where D is the total action dimension, $\tilde{A}_j^{XYZ} \in \mathbb{R}^{H\times 3}$ contains the virtual XYZ translational actions, and $a_{t:t+H}[0]_{:,4:D}$ copies all remaining action dimensions from the first policy sample.

Let $\mathcal{P}_s$ denote the keypoints selected for the current stage and let $\bar{p} = \frac{1}{|\mathcal{P}_s|}\sum_{p\in\mathcal{P}_s} p$ be their centroid. For proposal j, η_j controls the travel scale and a small vertical perturbation. We define the nominal

Algorithm 1 Sinkhorn algorithm for target-relaxed OT

Require: Cost matrix $\bar{\boldsymbol{C}} \in \mathbb{R}^{N\times(N+M)}$, uniform source distribution $\boldsymbol{p} = \frac{1}{N}\mathbb{1}_N$, reward softmax target distribution $\bar{\boldsymbol{q}}$ with $\bar{q}_j = \frac{\exp(\beta R(\bar{y}_j))}{\sum_{k=1}^{N+M}\exp(\beta R(\bar{y}_k))}$, entropy coefficient $\varepsilon > 0$, KL weight $\lambda > 0$, and maximum iterations $it_{\max}$

Initialize $\boldsymbol{K} \leftarrow \exp(-\bar{\boldsymbol{C}}/\varepsilon)$, $\boldsymbol{b} \leftarrow \mathbb{1}_{N+M}$, $\rho \leftarrow \lambda/(\lambda+\varepsilon)$, $it \leftarrow 0$
while $it \le it_{\max}$ and $\boldsymbol{a}, \boldsymbol{b}$ have not converged **do**
  $\boldsymbol{a} \leftarrow \frac{\boldsymbol{p}}{\boldsymbol{K}\boldsymbol{b}}$
  $\boldsymbol{b} \leftarrow \left(\frac{\bar{\boldsymbol{q}}}{\boldsymbol{K}^\top \boldsymbol{a}}\right)^\rho$
  $it \leftarrow it + 1$
end while
$\boldsymbol{a} \leftarrow \frac{\boldsymbol{p}}{\boldsymbol{K}\boldsymbol{b}}$ // Enforce the source marginal
return Transport plan $\bar{\boldsymbol{\Gamma}}^\star = \mathrm{diag}(\boldsymbol{a})\boldsymbol{K}\,\mathrm{diag}(\boldsymbol{b})$

Table 5: Experimental settings and steering hyperparameters.

Hyperparameter	**Libero-Pro**	**Libero-Plus**	**VLABench**	**Real world**
Evaluation protocol				
VLM name	GPT-5.5	GPT-5.5	GPT-5.5	GPT-5.5
Number of episodes	1600	800	190	200
VLM query limit	50	50	20	20
Schmitt upper / lower thresholds	0.8 / 0.6	0.8 / 0.6	0.8 / 0.6	0.8 / 0.6
Latent-space OT				
Policy-induced samples N	32	32	32	8
Reward-induced samples M	8	8	8	8
Transport time τ_s	0.1	0.1	0.3	0.4
Entropy coefficient ε	0.1	0.1	0.1	0.1
Target relaxation weight λ	100	100	5	10
Target sharpness β	100	100	100	100
Sinkhorn iteration limit $it_{\max}$	300	300	300	300
Adaptive steering				
Progress sigmoid sharpness k	50	50	50	50
Progress threshold b	0.85	0.85	0.85	0.85

virtual trajectory as the straight line from e_t to $\bar{p}$. To obtain diverse proposals, we perturb its travel scale and vertical direction with η_j. The resulting position at timestep $h \in \{1, \ldots, H\}$ is defined as:

$$\tilde{e}^{(j)}_{t+h} = e_t + \frac{h}{H}\left[\alpha_j(\bar{p} - e_t) + \delta_j \mathbf{e}_z\right], \tag{13}$$

We convert the desired translational displacement into the first three action dimensions using the policy's action scaling. Varying $\eta_j = (\alpha_j, \delta_j)$ produces proposals with different travel distances and approach heights, while copying the remaining four dimensions preserves the orientation and gripper behavior of the base policy. The resulting 7-dimensional chunks are then forward-noised to the shared denoising time τ_s and used as target particles for the OT process.

C Sinkhorn Solver for Target-relaxed OT

Algorithm 1 solves Eq. 9 using Sinkhorn scaling updates. The source distribution is uniform, and the target distribution is obtained by applying softmax to the reward scores. Unlike balanced Sinkhorn, which uses $\boldsymbol{b} \leftarrow \bar{\boldsymbol{q}}/(\boldsymbol{K}^\top \boldsymbol{a})$ to enforce the target marginal exactly, our target update uses the exponent $\rho = \lambda/(\lambda+\varepsilon)$ to account for the KL relaxation. Note that all divisions, exponentials, and powers in the algorithm are element-wise. The final update of $\boldsymbol{a}$ ensures $\bar{\boldsymbol{\Gamma}}^\star \mathbb{1}_{N+M} = \boldsymbol{p}$ after convergence or after reaching the iteration limit. The resulting target marginal is $(\bar{\boldsymbol{\Gamma}}^\star)^\top \mathbb{1}_N$, which generally differs from $\bar{\boldsymbol{q}}$ because of the KL relaxation. When $\lambda \to \infty$, $\rho \to 1$ and the updates recover balanced Sinkhorn; for finite λ, the target marginal is only softly matched. We use $it_{\max} = 300$ and implement the equivalent updates in the log domain for numerical stability.

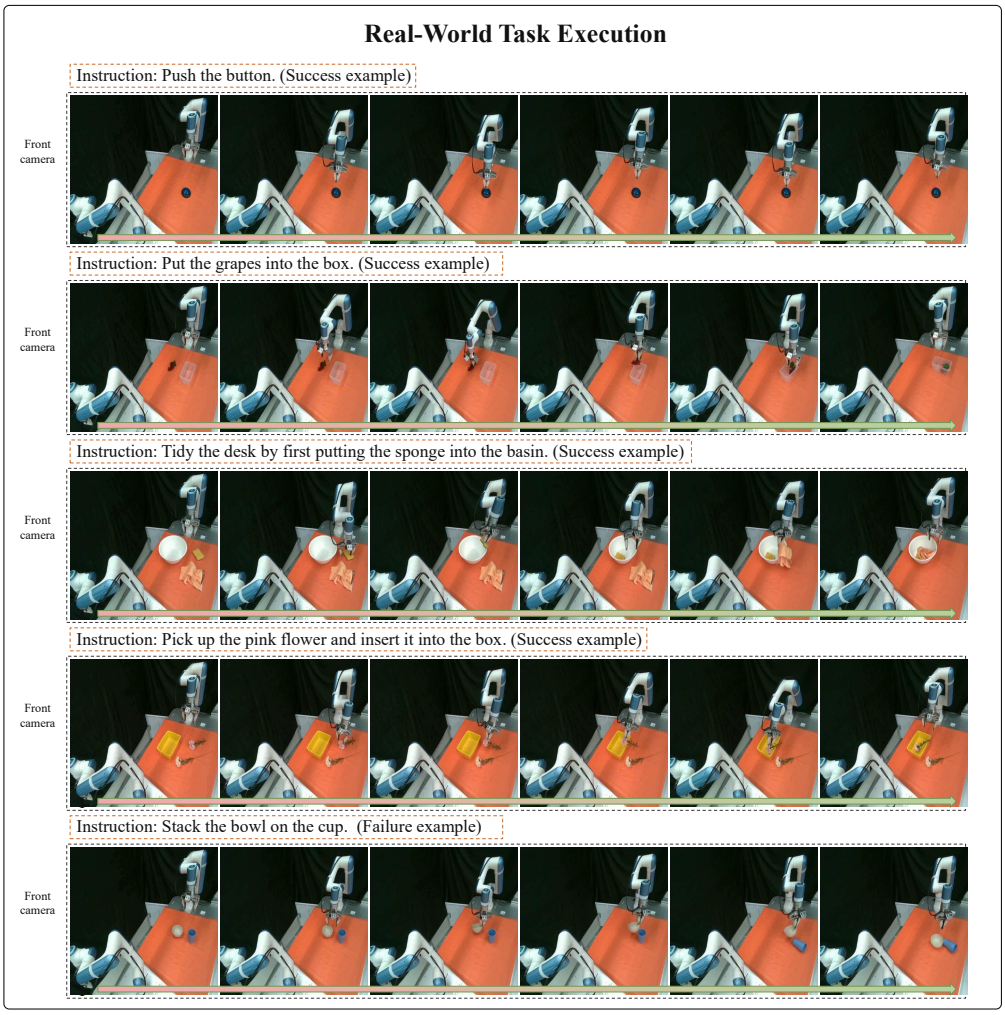

Figure 8: Representative real-world execution processes for five manipulation tasks.

D Experimental Details

We evaluate LOTUS on three simulated benchmarks (i.e., Libero-Pro, Libero-Plus, and VLABench) and real-world manipulation tasks. The specific testing settings are presented in Table 5.

E Visualizations of Real-World Execution Processes

Figure 8 visualizes representative execution sequences from the front camera. The examples cover four successful tasks and one failure case, with arrows indicating the temporal progression of each execution.

F Comparison with Existing Steering Methods

We further compare LOTUS with the SOTA inference-time steering method, i.e., VLS (Liu et al., 2026), on the Libero-Pro evaluation protocol described in Sec. 5.1. For a fair comparison, we reproduce VLS using the same GPT-5.5 VLM as LOTUS while retaining all other hyperparameters from its official implementation. We additionally report the results from the original VLS paper, denoted as VLS*. As shown in Table 6, one could observed that inference-time steering is necessary

for adapting VLA policies to OOD scenarios, as both VLS and LOTUS substantially outperform all pretrained models. Thanks to the adaptive balance between the action prior and task-aligned guidance, LOTUS achieves a higher overall success rate and outperforms VLS by 5.8%.

Table 6: We compare VLS and LOTUS on the Libero-Pro. Evaluations are conducted under task and position perturbations across the four LIBERO suites: Goal, Spatial, 10 (Long-Horizon), and Object, with each suite containing 10 tasks. For each task, we run 20 episodes and report the average success rate (%).

Method	**Task Perturbation**					**Position Perturbation**					**Overall**
	Goal	Spatial	10	Object	Avg.	Goal	Spatial	10	Object	Avg.	
OpenVLA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
π_0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\pi_{0.5}$	0.00	1.00	1.00	1.00	0.75	38.00	20.00	8.00	17.00	20.75	10.75
$\pi_{0.5}$ (LeRobot)	12.00	48.50	21.50	10.50	23.13	29.00	41.00	11.00	16.00	24.25	23.69
$\pi_{0.5}$ (LeRobot) + VLS*	33.50	54.00	25.50	41.00	38.50	38.00	42.00	15.50	45.00	35.13	36.81
$\pi_{0.5}$ (LeRobot) + VLS	32.00	44.00	28.00	43.00	36.75	37.50	35.50	15.00	45.50	33.38	35.06
$\pi_{0.5}$ (LeRobot) + LOTUS	31.00	56.50	37.50	43.50	42.13	36.50	58.00	15.00	48.50	39.50	40.81



Figure 9: Case study of LOTUS on Libero-Pro Task 4 under task perturbation. Frames follow the episode's execution order from left to right and then from top to bottom. Blue arrows indicate the motion directions of the steered action chunks produced by LOTUS.

G Case Study of the Steering Process in LOTUS

Figure 9 illustrates an episode from Task 4 under task perturbation in Libero-Pro. The instruction requires placing the yellow-and-white mug on the left plate and the white mug on the right plate.

LOTUS guides the robot through successive grasping and placement stages, directing each mug toward its specified destination. The steering direction changes with the current stage, illustrating how LOTUS aligns the policy's actions with the task instruction throughout execution.