

# VLS: Steering Pretrained Robot Policies via Vision–Language Models

Shuo Liu<sup>1,4</sup>, Ishneet Sukhvinder Singh<sup>2</sup>, Yiqing Xu<sup>3,4</sup>, Jiafei Duan<sup>1,4,\*</sup>, Ranjay Krishna<sup>1,4,\*</sup>

<sup>1</sup>University of Washington <sup>2</sup>University of Oxford

<sup>3</sup>National University of Singapore <sup>4</sup>Allen Institute for Artificial Intelligence

\*Co-advising

[vision-language-steering.github.io](https://vision-language-steering.github.io)

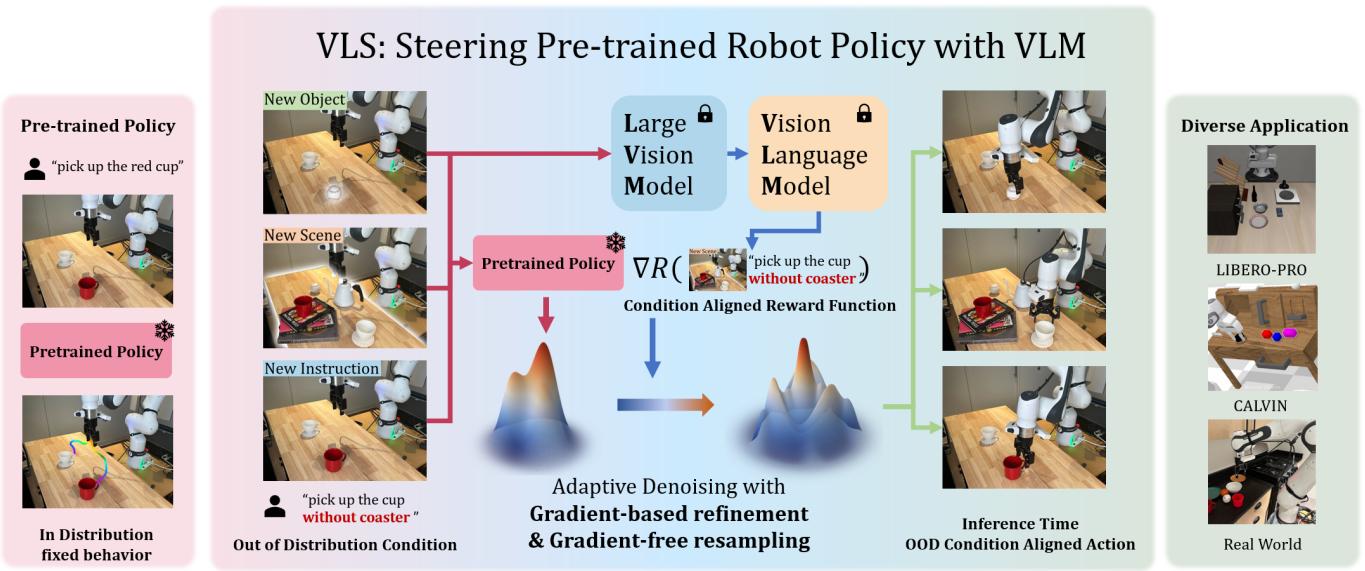


Fig. 1. We present **Vision–Language Steering (VLS)**, a training-free framework for inference-time steering of frozen generative robot policies. Our core idea is to leverage the open-world understanding capabilities of VLMs to generate reward functions for partially denoised action proposals, helping the base policy successfully operate in out-of-distribution (OOD) scenarios such as object changes, scene changes or instruction changes by correcting the denoising path. VLS demonstrates excellent performance in simulation benchmarks as well as real-world experiments, proving its effectiveness.

**Abstract—**Why do pretrained diffusion or flow-matching policies fail when the same task is performed near an obstacle, on a shifted support surface, or amid mild clutter? Such failures rarely reflect missing motor skills; instead, they expose a limitation of imitation learning under train–test shifts, where action generation is tightly coupled to training-specific spatial configurations and task specifications. Retraining or fine-tuning to address these failures is costly and conceptually misaligned, as the required behaviors already exist but cannot be selectively adapted at test time. We propose **Vision–Language Steering (VLS)**, a training-free framework for inference-time adaptation of frozen generative robot policies. VLS treats adaptation as an inference-time control problem, steering the sampling process of a pretrained diffusion or flow-matching policy in response to out-of-distribution observation–language inputs without modifying policy parameters. By leveraging vision–language models to synthesize trajectory-differentiable reward functions, VLS guides denoising toward action trajectories that satisfy test-time spatial and task requirements. Across simulation and real-world evaluations, VLS consistently outperforms prior steering methods, achieving a 31% improvement on CALVIN and a 13% gain on LIBERO-PRO. Real-world deployment on a Franka robot further demonstrates robust inference-time adaptation under test-time spatial and semantic shifts.

## I. INTRODUCTION

Once a child learns to place a cup at the center of a table, they have not merely mastered a single task. The same motor skill generalizes to placing the cup near an edge, atop a stack of books, or inside a crowded cabinet. For humans, motor execution naturally transfers across such spatial and task variations [18, 19]. For robots, however, skill execution is often tightly coupled to the specific spatial configurations and instructions encountered during training. As a result, even state-of-the-art manipulation policies [26, 38, 48, 9, 2, 5, 6, 41, 15, 1, 25] can fail when the observation or instruction at test time deviates from the training distribution: a robot that succeeds at placing an object at the center of a table may hesitate, collide, or miss entirely when asked to place it near an edge. These failures do not reflect missing motor capability, but rather the absence of a mechanism to adapt existing skills to new spatial requirements at test time.

Recent advances in robot learning have produced expressive pretrained policies, particularly those based on diffusion or flow-matching objectives, that achieve strong in-distribution performance [3, 33, 7]. However, these generative policies

remain brittle under out-of-distribution (OOD) observation–language inputs [40], where the required motor behaviors are already present in the training data but must be executed under altered spatial structure. Addressing such failures through retraining or fine-tuning is costly and conceptually misaligned, as it attempts to relearn behaviors rather than control their execution [53, 55, 47]. Expanding the training distribution to cover all possible spatial variations is therefore a brute-force solution to what is fundamentally an inference-time control problem.

In this work, we propose **Vision–Language Steering (VLS)**, a training-free framework for inference-time adaptation of pretrained robotic policies. VLS operates on a frozen base policy and addresses OOD inputs—joint observation–language pairs  $(o, l)_{OOD}$  that lie outside the expert dataset—by steering the policy’s sampling process at test time. Rather than modifying policy parameters, VLS reshapes the action distribution during inference so that generated trajectories satisfy the spatial and task structure implied by  $(o, l)_{OOD}$ . This formulation explicitly decouples skill execution from OOD task specification: the base policy provides reusable motor primitives, while inference-time steering controls how those primitives are composed and instantiated under OOD inputs.

Our approach is inspired by inference-time steering techniques developed for large language models and image generation models [11, 13, 20, 36, 27, 46], where a pretrained model’s output distribution is reshaped to elicit desired behaviors without additional training. VLS extends this paradigm to robotics by treating action generation as a controllable denoising process. Specifically, we leverage vision–language models (VLMs) to interpret OOD observation–language inputs, decompose tasks into execution stages, and synthesize differentiable reward functions that score action proposals with respect to spatial structure. These rewards provide dense, trajectory-level gradients that guide diffusion or flow-matching policies during inference. By grounding OOD inputs into geometric representations and injecting reward-based guidance into the denoising process, VLS enables existing skills to be executed reliably under spatial variation and unseen task specifications, while preserving the robustness of the frozen base policy.

We evaluate VLS in both simulation and real-world settings under observation and language shifts at test time. In simulation, we benchmark on CALVIN [35] and LIBERO-PRO [54], two widely used manipulation suites that explicitly stress inference-time adaptation to out-of-distribution observation–language inputs. On CALVIN, VLS consistently outperforms prior inference-time steering methods such as ITPS [49] and DynaGuide [16], achieving up to a 31% absolute improvement in success rate on long-horizon tasks. On LIBERO-PRO, VLS improves the success rate of frozen VLA policies, including OpenVLA [28],  $\pi_0$  [3], variants of  $\pi_{0.5}$  [4, 31] by up to 13% under both spatial (object layout) and semantic (task specification) perturbations. Finally, real-world experiments on a Franka robot demonstrate that VLS enables stable execution of multi-stage, language-specified

tasks under unseen object appearances, positional changes, and target substitutions, validating its effectiveness for practical deployment.

## II. RELATED WORK

### A. *Imitation-Trained Policies under Small Environment Shifts*

Large-scale imitation learning has enabled impressive generalist and VLM-conditioned robot policies, including diffusion- and flow-matching generators [2, 26, 38, 48]. However, a consistent failure mode remains: even mild changes in scene geometry or context (e.g., clutter, support-surface shifts, different object layouts) can cause sharp degradation. This brittleness is a known limitation of imitation learning: policies learn correlations between action and training context, and thus do not reliably produce constraint-satisfying behaviors when the environment configuration departs from the training manifold. This motivates methods that adapt execution at test time without assuming new data coverage.

### B. *VLM-based Scene Understanding with Re-optimization*

A common way is to use VLMs to generate scene representations that improves spatial understanding and then re-optimize actions online via planning, search, or iterative refinement [22, 23, 30]. While these approaches can handle unseen observations, they typically require rollouts, repeated evaluation, or online optimization loops, which are computationally heavy and often incompatible with real-time control. Moreover, they shift the burden of generalization to optimization at deployment, whereas our goal is to retain the pretrained policy as the skill prior and adapt behavior through lightweight inference-time control.

### C. *Inference-time Steering of Generative Policies*

Most related to our approach are inference-time methods that steer the sampling of a pre-trained policy.

**Value/critic-guided steering.** V-GPS re-ranks actions using an offline-learned value function to improve generalist policies without updating the backbone [37]. For diffusion policies, VGD injects gradients from a learned value/Q model into denoising to bias trajectories toward higher value while keeping the policy frozen [52]. These methods provide dense guidance, but they do so through an auxiliary learned objective, which can effectively reshape the policy toward the critic’s preferences. However, we view this as undesirable as the base policy should remain the invariant, and only the test-time constraints should modulate execution.

**Dynamics/world-model guided steering.** DynaGuide uses an external dynamics model to guide denoising, enabling multi-objective steering while preserving the diffusion prior [16]. Latent Policy Barrier uses a learned dynamics model to predict and optimize future latent states so trajectories remain within an expert manifold under covariate shift [45]. These approaches increase dependence on predictive modeling and rollout-style evaluation, and can become sensitive to model error and inference cost as it pushes adaptation burden into heavier test-time optimization.

**Human/VLM-in-the-loop steering and verification.** ITPS steers generative sampling through human interaction signals at inference time [49]. FOREWARN uses VLMs as open-vocabulary verifiers to select among candidate plans [51], and Do What You Say similarly checks reasoning-action faithfulness by filtering candidate action sequences using VLM-based alignment [50]. These methods demonstrate the power of semantic feedback, but their supervision is typically discrete and sparse, which forces adaptation to occur through selection/rejection over candidates rather than through continuous, differentiable steering within generation, making them sample-inefficient when the desired behavior requires fine-grained constraint satisfaction. The work most closely related to VLS is VLA-Pilot [32], but our focus is on guiding pre-trained policies to handle OOD scenarios, combining gradient-guided denoising processes with dynamic stage transitions, and conducting extensive testing in both simulation and real-world.

**Online improvement without finetuning the base policy.** Policy Decorator adds a residual refinement policy for online correction while preserving the backbone imitation policy [53]. USR unifies latent steering with residual refinement via a lightweight actor for online improvement of diffusion policies [55]. DSRL optimizes in diffusion latent/noise space to enable fast online improvement with black-box access to the base policy [47]. These methods rely on online learning or interaction, whereas we focus on pure inference-time adaptation with no additional training at deployment.

### III. PROBLEM FORMULATION

#### A. The OOD Dilemma in Imitation Learning

Imitation learning aims to learn a policy  $\pi_\theta$  from an expert demonstration dataset  $\mathcal{D}_{expert} = \{(o_i, \mathbf{a}_i), l_i\}_{i=1}^N$ , modeling the state-conditional action distribution  $p(\mathbf{a}|o)$ . Typically, at environment time step  $t$ , the training target of a policy  $\pi_\theta$  is to maximize the likelihood of an action chunk  $\mathbf{a}_{t:t+T}$  with chunk horizon  $T$ , conditioned on the observation  $o$  (typically RGB images and robot proprioception) and language instruction  $l$ :

$$\max_{\theta} \mathbb{E}_{(\mathbf{a}, o, l) \sim \mathcal{D}_{expert}} \left[ \sum_{t=1}^T \log \pi_\theta(\mathbf{a}_{t:t+T} | o_t, l) \right]. \quad (1)$$

After training, the policy  $\pi_\theta$ 's weight  $\theta$  can be frozen, which we refer to as the **base policy**  $\pi^*$ . However, this training objective is inherently static and distribution-dependent. When the policy is deployed in real world, it inevitably encounters out-of-distribution (OOD) scenarios  $\{o, l\}_{OOD} \notin \mathcal{D}_{expert}$ , which ranging from observation shift ( $o_{OOD}$ ) such as change of visual backgrounds or object layouts to semantic ambiguity ( $l_{OOD}$ ) such as unseen instructions. Since the base policy  $\pi^*$  tends to overfit on the spatial and semantic correlations present in the training manifold, it exhibits severe brittleness when faced with such OOD scenarios [42].

#### B. Diffusion and Flow Matching Policies

In recent years, denoising generative models have become a cornerstone for imitation learning [9]. These models learn

to transform a simple Gaussian distribution into a complex target action distribution  $q(\mathbf{a}_{t:t+T}^0)$  through forward diffusion and reverse denoising [21]. The forward process gradually adds Gaussian noise to the clean action trajectory  $\mathbf{a}_{t:t+T}^0$ , transforming it into a Gaussian distribution  $\mathbf{a}_{t:t+T}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . A network  $\epsilon(\mathbf{a}^k, o, l, k)$  is then trained to predict the added noise at denoising step  $k \in [0, 1, \dots, K]$ , conditioned on the observation  $o$  and instruction  $l$ . The reverse process samples action from  $\mathbf{a}_{t:t+T}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and applies the update

$$\mathbf{a}_{t:t+T}^{k-1} = \frac{1}{\sqrt{\alpha_k}} \left( \mathbf{a}_{t:t+T}^k - \frac{1 - \alpha_k}{\sqrt{1 - \bar{\alpha}_k}} \epsilon(\mathbf{a}_{t:t+T}^k, o, l, k) \right) + \sigma_k \mathbf{z}, \quad (2)$$

where  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ,  $\alpha_k$  and  $\bar{\alpha}_k$  are noise schedule coefficients, eventually producing  $\mathbf{a}_{t:t+T}^0$  that approximates  $q(\mathbf{a}_{t:t+T}^0)$ .

Beyond denoising diffusion, flow matching simplifies the denoising process by learning a continuous velocity field  $v$ . While flow-matching also iteratively refines samples from a Gaussian distribution, different from discrete indices  $\{K, K-1, \dots, 0\}$  that often used in diffusion models, flow matching utilizes a continuous time interval  $[0, 1]$ , where  $k = 1$  and  $k = 0$  correspond to the noise distribution and the clean action trajectory, respectively. Since both variables fundamentally represent the progression of the denoising process, we unify the notation by using  $k$  for both frameworks to avoid redundant symbolic definitions and potential ambiguity. Flow matching models the transition of distribution as an Ordinary Differential Equation (ODE):

$$\frac{d\mathbf{a}_{t:t+T}^k}{dt} = v(\mathbf{a}_{t:t+T}^k, o, l, k). \quad (3)$$

#### C. Problem Formulation

Given a pre-trained **base policy**  $\pi^*$ , our goal is to enable it to adapt to OOD scenarios  $\{o, l\}_{OOD} \notin \mathcal{D}_{expert}$  at inference time without fine-tuning. To enable  $\pi^*$  adapt to the new condition  $(o, l)_{OOD}$ , we leverage Classifier Guidance [13] to steer the sampling process of the base policy. The core idea is to find a guidance function and use the gradient  $g = \nabla_{\mathbf{a}_{t:t+T}^k} \log p((o, l)_{OOD} | \mathbf{a}_{t:t+T}^k)$  which represent the score of joint distribution of action proposal  $\mathbf{a}_{t:t+T}^k$  and OOD condition  $(o, l)_{OOD}$ , to guide the direction of denoising. We provide a detailed derivation of gradient-based steering for diffusion and flow-matching policies in Appendix.

For diffusion models, the modified the noise prediction is:

$$\hat{\epsilon} = \epsilon(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD}, k) - \lambda \cdot \sqrt{1 - \bar{\alpha}_k} \cdot g(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD}), \quad (4)$$

where  $\lambda$  is the guidance scale hypermeter to control the guidance strength. Similar to diffusion model, a flow matching policy can be steered to accommodate the condition  $y = (o, l)_{OOD}$  by controlling the predicted velocity field:

$$\hat{v} = v(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD}, k) + \lambda \cdot g(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD}). \quad (5)$$

The main challenge lies in modeling the gradient guidance function  $g(\cdot)$ . In real-world OOD deployment, the conditioning

input  $(o, l)_{OOD}$  is not a simple class label, but a structured specification that implicitly encodes spatial and semantic constraints. A valid guidance function must therefore flexibly and accurately model  $\log p((o, l)_{OOD} | \mathbf{a}_{t:t+T}^k)$ , while satisfying two requirements: (1) it must correctly interpret the geometry and logical structure induced by the OOD condition, and (2) it must provide dense, informative gradients with respect to the proposed action trajectory.

#### IV. OUR APPROACH: VLS

The core of Vision–Language Steering (VLS) is to approximate the guidance signal  $g \triangleq \nabla_{\mathbf{a}_{t:t+T}^k} \log p((o, l)_{OOD} | \mathbf{a}_{t:t+T}^k)$  for inference-time adaptation under OOD inputs. Since the likelihood term is not directly available in real-world deployment, VLS constructs a differentiable surrogate score  $\mathcal{R}$  that maps an action proposal to how well it satisfies the spatial and semantic constraints implied by the OOD input pair  $(o, l)_{OOD}$ :

$$\mathcal{R}(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD}) \approx \log p((o, l)_{OOD} | \mathbf{a}_{t:t+T}^k). \quad (6)$$

By design,  $\mathcal{R}$  is differentiable with respect to  $\mathbf{a}_{t:t+T}^k$ , enabling gradient guidance  $g \approx \nabla_{\mathbf{a}_{t:t+T}^k} \mathcal{R}(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD})$  to steer the denoising trajectory of the frozen base policy  $\pi^*$  without fine-tuning. As illustrated in Figure 2, VLS instantiates this pipeline with three components:

- 1) **OOD input grounding and VLM-generated differentiable scoring.** Ground the OOD input pair  $(o, l)_{OOD}$  into a compact geometric scaffold  $\mathcal{P}$  of task-relevant 3D keypoints, and use a vision–language model to synthesize stage-wise, programmatic reward functions  $\mathcal{R}(\cdot, \mathcal{P})$  that provide differentiable scores over action proposals (Sec. IV-A).
- 2) **Inference-time denoising guidance.** Inject  $\nabla_{\mathbf{a}} \mathcal{R}_s$  into the diffusion or flow-matching denoising updates, combining gradient-based refinement with particle-level diversity and resampling to steer the action distribution under OOD inputs (Sec. IV-B).
- 3) **Closed-loop execution control and stage switching.** Use execution feedback to adaptively regulate the guidance strength across action chunks and to determine transitions between stage-specific reward functions, enabling stable multi-stage execution under physical uncertainty (Sec. IV-C).

Full implementation details corresponding to each component of VLS, including the VLM prompt design and reward function structure, are provided in Appendix.

##### A. OOD Input Grounding and Reward Generation

To construct the surrogate score  $\mathcal{R}(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD})$ , VLS must map the high-dimensional OOD input pair  $(o, l)_{OOD}$  to a differentiable function over the action space. We achieve this by (i) grounding the OOD input into a compact set of task-relevant spatial variables, and (ii) synthesizing programmatic, differentiable reward functions over these variables.

1) *OOD Input Grounding:* Given an OOD input pair  $(o, l)_{OOD}$ , VLS first identifies the objects and regions relevant to the manipulation task using a vision–language model (VLM). For each identified object, we apply the Segment Anything Model (SAM [29]) to obtain a set of object masks  $\mathcal{M}$ . Following [23], we extract semantically aligned dense visual features using DINOv2 [8], producing a patch-wise feature map  $\Phi \in \mathbb{R}^{H \times W \times d}$ , which is filtered using the corresponding object masks.

To recover physical structure, masked pixels are reprojected into a 3D point cloud using depth information. Each point is represented by the concatenation of its DINO feature ( $d$  dimensions) and its 3D spatial coordinates, yielding a  $(d + 3)$ -dimensional representation. We cluster these object-centric point clouds to obtain a set of task-relevant keypoints  $\mathcal{P} = \{p_i\}_{i=1}^n$ , where each  $p_i \in \mathbb{R}^3$  anchors a physically meaningful spatial constraint in the environment.

Through this process, the OOD input pair  $(o, l)_{OOD}$  is deterministically compressed into a geometric scaffold  $\mathcal{P}$ , which exposes the spatial variables required for downstream, differentiable reward evaluation.

2) *Programmatic Reward Generation:* Given the grounded keypoint set  $\mathcal{P}$ , VLS leverages the cross-modal reasoning capability of VLMs to synthesize stage-aware, programmatic reward functions. Specifically, the VLM is queried to: (i) decompose the task implied by  $(o, l)_{OOD}$  into  $S$  sequential stages, and (ii) for each stage  $s \in \{1, \dots, S\}$ , generate a differentiable reward function that evaluates how well an action proposal satisfies the corresponding spatial constraints.

Formally, for each stage  $s$ , the VLM produces a reward function  $\mathcal{R}_s(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD}) = f_{VLM}(\mathbf{a}_{t:t+T}^k, \mathcal{P}, s)$ , where  $\mathcal{R}_s$  defines a stage-specific potential field over the action space, returning a scalar value that is differentiable with respect to the action proposal  $\mathbf{a}_{t:t+T}^k$ . Intuitively,  $\mathcal{R}_s$  measures the degree to which the proposed action trajectory respects the spatial relationships encoded by  $\mathcal{P}$  at stage  $s$ . To ensure differentiability, we constrain the VLM to output programmatic reward definitions implemented as PyTorch [39] functions composed of differentiable tensor operations (e.g., distances, dot products, soft constraints). During inference, gradients are backpropagated through the instantiated reward function, while the VLM itself remains a non-differentiable, off-graph component.

This construction directly instantiates the gradient guidance signal required for inference-time steering:

$$g_s \triangleq \nabla_{\mathbf{a}_{t:t+T}^k} \mathcal{R}_s(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD}), \quad (7)$$

providing a dense, action-space gradient that approximates  $\nabla_{\mathbf{a}_{t:t+T}^k} \log p((o, l)_{OOD} | \mathbf{a}_{t:t+T}^k)$ .

##### B. Action Denoising Process Guidance

Given the stage-wise reward functions  $\{\mathcal{R}_s\}$  constructed in Sec. IV-A, we next describe how VLS injects their gradients into the denoising process of the frozen base policy  $\pi^*$ . Our goal is to steer inference-time sampling toward action trajectories that satisfy the constraints implied by the OOD

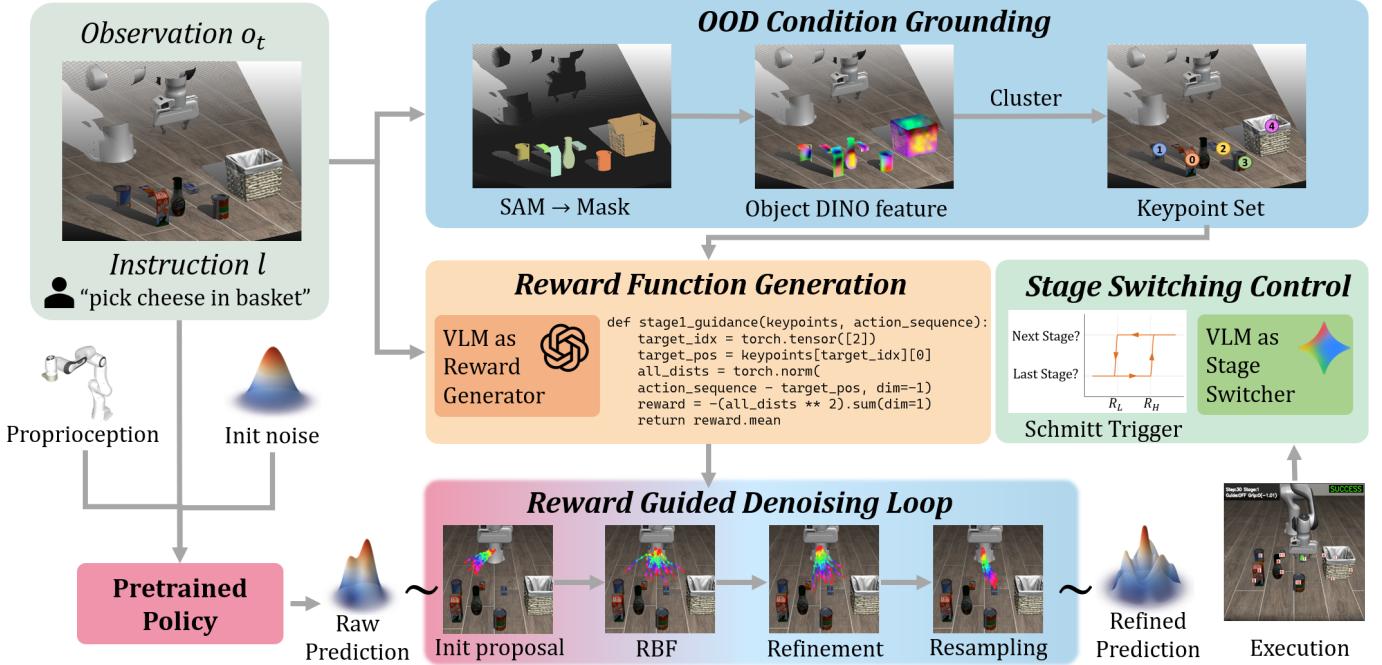


Fig. 2. **VLS pipeline overview.** At environment time step  $t$ , given RGB-D observation  $o_t$  and language instruction  $l$ , VLS firstly utilize the Segment Anything Model (SAM [29]) and DINOv2 [8] feature to ground condition into a set of spatial keypoints  $\mathcal{P}$ . Subsequently, a Vision-Language Model will be queried to generates a series of stage-aware differentiable programmatic reward functions  $\{\mathcal{R}_s\}_{s=1}^S$ , based on observation, task instruction and keypoints, which are used to guide the action generation process of the frozen base policy  $\pi^*$ : during the denoising sampling loop, the system precisely corrects action trajectories by injecting reward gradients, incorporating RBF [24] repulsion terms and a Feynman–Kac [44] based resampling mechanism to rapidly converge to high-reward regions while maintaining sampling diversity. Finally, VLS constructs a closed-loop stage switching system based on reward feedback, utilizing adaptive guidance strength and Schmitt-trigger [43] switching logic to monitor execution progress, thereby automatically triggering phase transitions or retry strategies when facing physical uncertainties (such as object displacement or manipulation failures), ensuring robust completion of long-horizon manipulation tasks in OOD environments.

input pair  $(o, l)_{OOD}$ , while preserving diversity and robustness under complex, multi-modal landscapes. To this end, we combine gradient-based refinement with particle-level diversity and gradient-free resampling.

### 1) Diverse Proposal Initialization with Repulsive Forces:

At each environment timestep  $t$ , denoising begins by sampling a batch of  $B$  action proposals  $\{\mathbf{a}_{t:t+T}^k[i] \sim \mathcal{N}(\mathbf{0}, \mathbf{I})\}_{i=1}^B$  independently. To prevent the batch from collapsing prematurely to a narrow mode of the base policy, we introduce a diversity-promoting repulsive force during the early denoising steps. Inspired by [24, 10], we define a repulsive gradient based on pairwise distances:

$$g_{RBF}^k[i] = \nabla_{\mathbf{a}_{t:t+T}^k[i]} \sum_{j \neq i} \frac{1}{\|\mathbf{a}_{t:t+T}^k[i] - \mathbf{a}_{t:t+T}^k[j]\|_2 + \epsilon}. \quad (8)$$

This term encourages action proposals within the batch to remain separated, ensuring broad coverage of the action manifold and providing diverse candidates for subsequent reward-based guidance.

2) Gradient-Based Refinement: To bias the denoising trajectory toward actions that satisfy the OOD-induced constraints, we inject the stage-specific reward gradient  $g_s = \nabla_{\mathbf{a}_{t:t+T}^k} \mathcal{R}_s(\mathbf{a}_{t:t+T}^k, (o, l)_{OOD})$  into the denoising updates. Specifically, following Eq. (4) for diffusion policies and Eq. (5) for flow-matching policies, the reward gradient is

added to the noise or velocity prediction at each denoising step  $k$ , steering samples toward regions of higher reward. To improve stability under noisy gradients, we adopt stochastic refinement with multiple inner updates per denoising step, analogous to MCMC-based guidance [16, 49, 17]. This procedure enables smoother exploration of the reward landscape and mitigates sensitivity to local gradient artifacts.

3) Gradient-Free Resampling via Feynman–Kac Steering: In addition to gradient-based refinement, VLS employs a gradient-free resampling mechanism based on Feynman–Kac (FK) steering [12, 14, 44]. We interpret the batch of action proposals as an interacting particle system and periodically resample particles according to reward-based potentials. For the  $i$ -th particle at denoising step  $k$ , the potential is defined as

$$G_i^k = \exp(\mathcal{R}_s(\mathbf{a}_{t:t+T}^k[i], (o, l)_{OOD})). \quad (9)$$

Normalized weights  $w_i^k = G_i^k / \sum_{j=1}^B G_j^k$  are computed, and multinomial resampling is applied to the particle set. This procedure effectively tilts the transition kernel of the generative process toward the target distribution  $p(\mathbf{a} | (o, l)_{OOD})$ , allowing high-reward particles to replicate while pruning proposals that violate the OOD-induced constraints.

By combining continuous gradient-based steering with discrete, reward-weighted resampling, VLS enables the frozen base policy  $\pi^*$  to navigate complex, multi-modal constraint

landscapes efficiently, avoiding the sample inefficiency and brittleness of purely selection-based methods.

### C. Closed-Loop Execution Control and Stage Switching

To handle physical uncertainty (e.g., object slippage, partial execution) and to robustly coordinate multi-stage tasks, VLS incorporates a closed-loop execution control mechanism. This mechanism uses execution feedback to (i) adaptively regulate the guidance strength  $\lambda$  at the level of action chunks, and (ii) determine when to switch between stage-specific reward functions  $\{\mathcal{R}_s\}$  during task execution.

1) *Adaptive Guidance Strength*: Within a fixed task stage  $s$ , multiple action chunks  $\{\mathbf{a}_{t:t+T}\}$  may be generated sequentially. We adapt the guidance strength  $\lambda_t$  for each action chunk  $t$  based on the relative reward achieved under the current stage-specific reward function  $\mathcal{R}_s$ .

Let  $\mathcal{R}_s^t$  denote the reward value of the final denoising step for the action chunk generated at chunk index  $t$  under stage  $s$ , and let  $\mathcal{R}_s^{base}$  denote the corresponding reward obtained from the first action chunk generated for this stage. The guidance strength is computed as:

$$\lambda_t = \lambda_{\max} \cdot \text{sigmoid}\left(1 - \frac{\mathcal{R}_s^t}{\mathcal{R}_s^{base}}\right). \quad (10)$$

This adaptive schedule increases guidance when the current action chunk deviates from the constraints encoded by  $\mathcal{R}_s$ , and progressively reduces guidance as execution improves. As a result, strong steering is applied when coarse correction is required, while the frozen base policy  $\pi^*$  is allowed to dominate during fine-grained manipulation near stage completion.

2) *Schmitt-Trigger-Based Stage Switching*: To robustly determine when to transition between stages and to avoid oscillatory behavior near stage boundaries, we employ a hysteresis-based switching mechanism inspired by the Schmitt trigger [43]. For the current stage  $s$ , we define two reward thresholds  $R_{high}$  and  $R_{low}$ , and compute a switching signal  $Q_t$  based on the evolution of  $\mathcal{R}_s^t$ :

$$Q_t = \begin{cases} \text{Advance stage,} & \mathcal{R}_s^t > R_{high}, \\ \text{Maintain stage,} & R_{low} \leq \mathcal{R}_s^t \leq R_{high}, \\ \text{Reinforce stage,} & \mathcal{R}_s^t < R_{low}. \end{cases} \quad (11)$$

When a switching event is triggered, a vision–language model is queried to interpret the execution outcome and select the appropriate next-stage reward function  $\mathcal{R}_{s+1}$ , or to continue applying the current stage reward  $\mathcal{R}_s$  with updated guidance strength. By introducing hysteresis into stage switching, VLS avoids premature transitions and repeated oscillations, enabling stable coordination across stages under physical uncertainty and complex OOD execution dynamics. The whole algorithm of VLS can be found at Algorithm 1.

## V. EXPERIMENTS

We evaluate (VLS) in both simulation and real-world settings. Simulation experiments are conducted on two widely

---

### Algorithm 1: VLS Algorithm

---

```

Input: Base policy  $\pi^*$ ; Initial observation  $o_0$ ; language
instruction  $l$ ; chunk horizon  $T$ ; sample batch size  $B$ 
Output: Action chunk  $\mathbf{a}_{t:t+T}$ 
1 // Condition grounding and reward
   generation;
2  $\mathcal{P} = \{p_i\}_{i=1}^n \leftarrow LVM(o_0, l)$ 
3  $\{\mathcal{R}_s(\mathbf{a}_{t:t+T}, \mathcal{P})\}_{s=1}^S \leftarrow f_{VLM}(o_0, l, \mathcal{P})$ 
4 // Initialize parameters;
5  $s \leftarrow 1$ ;
6  $MCMC \leftarrow 4$  if  $\pi^*$  is diffusion else 1;
7 // Denoising loop at action chunk index  $t$ ;
8 Sample initial proposals:  $\{\mathbf{a}_{t:t+T}^k[i] \sim \mathcal{N}(0, I)\}_{i=1}^B$ ;
9 for  $k = K \rightarrow 0$  do
10   // Diversity initialization;
11    $g_{RBF}^k[i] = \nabla_{\mathbf{a}_{t:t+T}^k[i]} \sum_{j \neq i} \frac{1}{\|\mathbf{a}_{t:t+T}^k[i] - \mathbf{a}_{t:t+T}^k[j]\|_2 + \epsilon}$ 
12   Use  $g_{RBF}^k$  as  $g$  in Eq. (4) or Eq. (5)
13   // Gradient-based refinement;
14    $g_{reward}^k = \nabla_{\mathbf{a}_{t:t+T}^k} \mathcal{R}_s(\mathbf{a}_{t:t+T}^k, \mathcal{P})$ ;
15   for  $m = 1 \rightarrow MCMC$  do
16     Use  $g_{reward}^k$  as  $g$  in Eq. (4) or Eq. (5)
17   // Gradient-free resampling;
18   for  $i = 1 \rightarrow B$  do
19      $G_i^k \leftarrow \exp(\mathcal{R}_s(\mathbf{a}_{t:t+T}^k[i], \mathcal{P}))$ ;
20      $w_i^k \leftarrow G_i^k / \sum_{j=1}^B G_j^k$ ;
21   Resample  $\{\mathbf{a}_{t:t+T}^k[i]\}_{i=1}^B$  according to  $\{w_i^k\}$ 
22 // Closed-loop execution control;
23 Adapt  $\lambda_t$  via Eq. (10)
24 Update stage  $s$  via Eq. (11)
25 return  $\mathbf{a}_{t:t+T}[0]$ 

```

---

used manipulation benchmarks, CALVIN [35] and LIBERO-PRO [54], while real-world deployment is performed on a Franka Emika robot. We systematically test generalization under both spatial and semantic shifts. To explicitly model inference-time out-of-distribution (OOD) conditions, we introduce controlled perturbations at test time along two axes:

**Observation perturbations.** We modify the environment state by (i) adding previously unseen objects as distractors, (ii) changing objects’ attribute during testing, and (iii) changing the positions or orientations of task-relevant objects and support surfaces.

**Language perturbations.** We alter task instructions by changing target objects and goal behaviors. More details on the perturbation and task description are provided in Appendix.

Our evaluation answers the following questions:

- **Q1. Is inference-time steering necessary to handle observation and language shifts at test time?**
- **Q2. Does VLS provide stronger adaptation than existing inference-time steering approaches?**
- **Q3. What is the contribution of each component in the VLS framework?**
- **Q4. Can VLS adapt policies in the real world with minimal computational overhead?**

Method	Task Perturbation					Position Perturbation					Overall Avg.
	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
$\pi_0$	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	<b>33.50</b>	<b>54.00</b>	<b>25.50</b>	<b>41.00</b>	<b>38.50</b>	<b>38.00</b>	<b>42.00</b>	<b>15.50</b>	<b>45.00</b>	<b>35.13</b>	<b>36.81</b>

Table I. LIBERO-PRO results. We test VLA baselines and a frozen  $\pi_{0.5}$  policy with/without VLS. The experimental environment consists of LIBERO-PRO [54]’s task and position perturbation, applied to LIBERO [34]’s four suites: Goal, Spatial, 10 (Long) and Object, with each suite containing 10 tasks. For each task in each suite, we test 20 episodes and report the average success rates (%). “Overall” reports the mean across all columns.

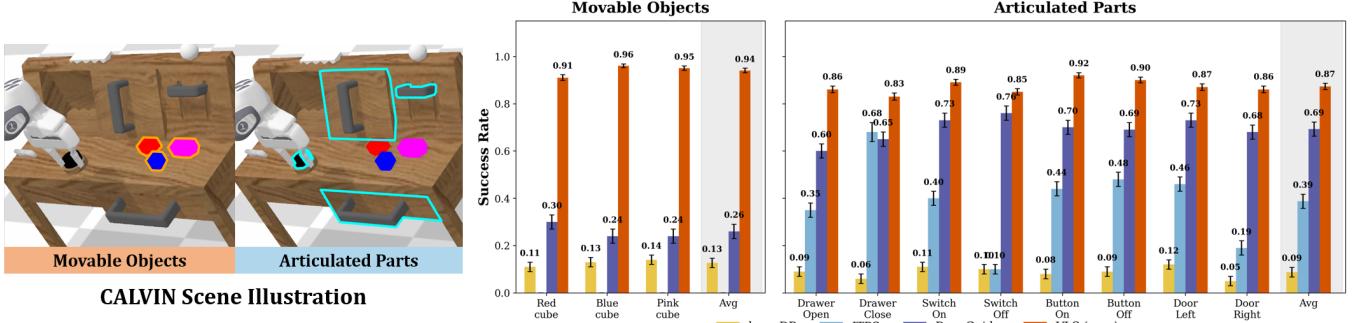


Fig. 3. **Steering methods comparison on CALVIN.** Success rates for VLS (ours), DynaGuide, ITPS, and the base diffusion policy across movable objects (cubes) and articulated parts (drawer, switch, button, door). VLS achieves 94% average on movable objects ( $7.4 \times$  over base policy) and 87% on articulated parts ( $9.6 \times$  boost), outperforming prior steering methods by 15–25 percentage points. Error bars show standard deviation over 600 episodes per task.

### A. Baselines

We compare VLS against seven baselines, grouped into VLA models and inference-time steering methods. (Detailed Implementation in Appendix.)

**VLA models:** To answer Q1, we evaluate four leading VLA models that rank highly on the LIBERO-PRO leaderboard: OpenVLA [28],  $\pi_0$  [3],  $\pi_{0.5}$  [4], and  $\pi_{0.5}$  LeRobot finetuned version [31]. All models use a VLM backbone to jointly reason over observations and language instructions and are evaluated without any fine-tuning on our OOD test scenarios.

**DP Steering policies:** To answer Q2, we compare against two popular inference-time steering approaches on the same frozen base policy: i) DynaGuide [16], which steers denoising using distances in pretrained DINO feature space as heuristic guidance; and ii) ITPS [49], which selects from a predefined set of guidance functions based on the detected OOD condition.

**Ablation:** To answer Q3, we evaluate three ablated variants of VLS that remove one component at a time: i) w/o gradient guidance, ii) w/o Feynman–Kac (FK) resampling, and iii) w/o RBF-based diversity initialization.

### B. Results

**Inference-Time Steering Is Necessary.** We choose LIBERO-PRO [54], a simulation benchmark, as our testing platform. This is an OOD test suite developed based on LIBERO [34], which primarily includes comprehensive perturbations across five aspects of the original LIBERO’s

four task suites: object, position, semantic, task, and environment. Among these, the **position** and **task perturbations** best align with the description of OOD scenarios in this paper. The position perturbation refers to relocating objects ( $o_{OOD}$ ) while keeping language instructions unchanged. The task perturbation refers to redefining task logic and target states, where visual observations remain in-distribution while language instructions are completely changed ( $l_{OOD}$ ). For each perturbation for tasks in each suites, we test 20 episodes. We choose Success Rate (SR) as metric. We evaluate four leading VLA models that rank highly on the LIBERO-PRO leaderboard: OpenVLA [28],  $\pi_0$  [3],  $\pi_{0.5}$  [4], and  $\pi_{0.5}$  LeRobot finetuned version [31].

As shown in Table I, these pre-trained VLAs, despite leveraging pretrained VLM backbones for perception and instruction understanding, struggle to adapt to joint observation and language shifts at test time. VLS consistently outperforms all evaluated VLA models under joint observation and language perturbations (Table I). While VLAs exhibit strong in-distribution performance, their success rates drop sharply under OOD conditions. This failure persists despite the use of pretrained VLM backbones. We attribute this to the fact that post-training on robot data entangles spatial reasoning with specific training contexts, effectively degrading the VLM’s generalization ability when the execution environment deviates from the training manifold. These results shows inference-time steering is necessary for pretrained policy adaptation.

**VLS Outperforms Existing Steering Methods.** We compared with two leading steering methods, DynaGuide [16] and

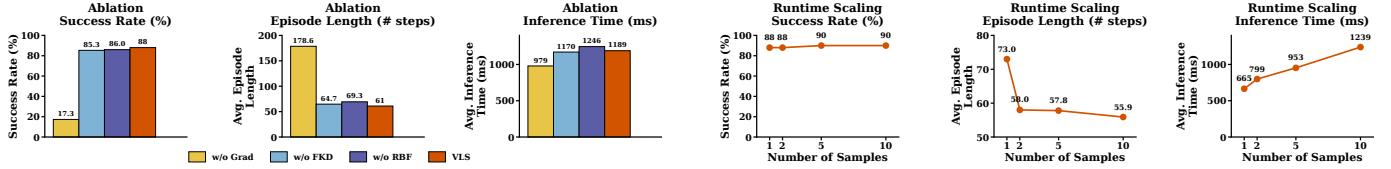


Fig. 4. (left) Ablation of VLS components (50 episodes per task). We compare Full VLS (gradient guidance + FK steering + RBF diversity, with  $K = 10$ ) against variants that remove FK steering (w/o FKD), remove RBF diversity (w/o RBF), or remove gradient guidance (w/o grad). (right) Scaling with sample batch size  $K$  on `door_left` (50 episodes). Larger  $K$  improves performance but increases inference time, illustrating a compute–performance tradeoff.

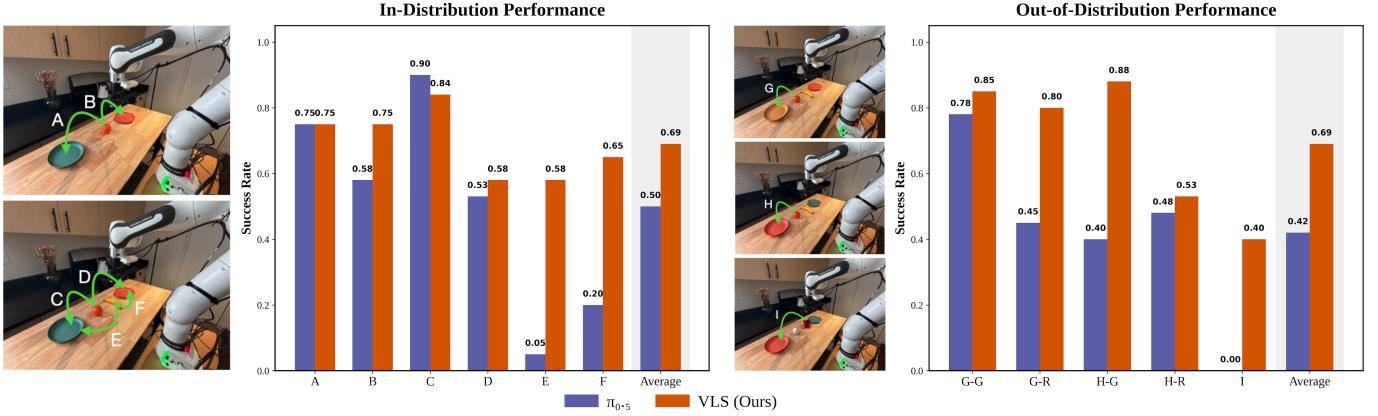


Fig. 5. **Real-world Deployment on a Franka robot.** (Left: In-distribution tasks) Task layouts, language instructions, and success rates for in-distribution real-world manipulation. Level 1 (top) requires placing an orange onto a specified plate (red or green) based on the instruction. Level 2 (bottom) introduces an additional object (banana), requiring sequential selection of both the target object and the target plate. Bar plots report per-task and average success rates for the frozen  $\pi$ -0.5 baseline and VLS. (Right: Out-of-distribution tasks) Task layouts, instructions, and results under test-time distribution shifts. We evaluate three OOD variants: (1) Appearance shift (top), replacing the red/green plate with a previously unseen yellow plate; (2) Position shift (middle), swapping the locations of the two plates while keeping the instruction unchanged; (3) Object shift (bottom), replacing the banana with a never-before-seen mug and instructing the robot to place the mug on the green plate. Each task is evaluated over 20 trials. Grasping the correct object contributes 50% success, and full task completion contributes 100%. VLS consistently outperforms the baseline and maintains robust execution under real-world OOD conditions.

ITPS [49] on CALVIN [35]. As shown in left part of Figure 3, where a Franka Panda robot interacts with a tabletop scene containing articulated objects (door, drawer, button, switch) and three randomly placed colored cubes (red, blue, pink).

As illustrated in Figure 3, all steering methods improve over the unsteered base diffusion policy, confirming the necessity of inference-time steering. On *MovableObjects* (cube manipulation), VLS achieves a 94% average success rate, corresponding to a  $7.4\times$  improvement over the base policy. On *ArticulatedParts* (drawer, switch, button, door), VLS reaches 87% average success, a  $9.6\times$  gain. ITPS performs reasonably on articulated tasks with fixed target states, but fails on movable-object tasks where object positions vary across episodes. DynaGuide improves performance across both task groups, but its DINO-feature-based heuristic lacks the expressiveness to capture task-specific spatial requirements. In contrast, VLS conditions its guidance directly on the current observation–language input, enabling precise steering under spatial variability and task-dependent constraints that heuristic guidance cannot reliably handle.

**Ablation Study of components.** We evaluate three ablated variants of VLS that remove one component at a time: i) w/o gradient guidance, ii) w/o Feynman–Kac (FK) resampling, and iii) w/o RBF-based diversity initialization.

**Effect of gradient-based guidance.** Removing gradient guidance causes a severe performance collapse across all tasks, with success rates dropping to near-failure and episode lengths increasing substantially (Figure 4 (left)). This confirms that dense, trajectory-differentiable guidance is the primary driver of VLS’s effectiveness.

**Role of FK resampling and RBF diversity.** Removing FK resampling or RBF-based diversity has a smaller impact on success rate but consistently degrades efficiency and stability. These components improve sample efficiency by preventing premature collapse to suboptimal modes and by maintaining global coverage early in denoising.

**Scaling with sample batch size.** As shown in Figure 4 (right), increasing the batch size improves success rates and reduces episode length, at the cost of higher inference latency. This exposes a practical compute–performance trade-off that can be tuned for deployment.

Together, these results show that both gradient-free exploration and gradient-based refinement are necessary for robust inference-time control. Robust inference-time adaptation requires both gradient-free global exploration (to avoid poor initial modes) and gradient-based local refinement (to satisfy fine-grained constraints during execution).

**VLS Enables Efficient Real-World Deployment.** We eval-

uate VLS on a Franka Emika robot to test whether inference-time steering can reliably adapt a frozen VLA policy under real-world test-time variation. (Detailed Implementation in Appendix.)

As shown in Figure 5, VLS consistently improves real-world task success over the frozen  $\pi$ -0.5 baseline across both in-distribution and out-of-distribution settings. In in-distribution tasks requiring object selection and placement, VLS achieves a 69% average success rate, outperforming the baseline by 19%. Under out-of-distribution conditions involving appearance changes, object repositioning, and novel object substitutions, the baseline performance degrades sharply, while VLS maintains stable execution and substantially higher success rates. In the most challenging object-level OOD case, where the target object is replaced by a previously unseen mug, the baseline fails entirely, whereas VLS succeeds in 40% of trials. These results show that VLS can be deployed efficiently in real robotic systems and enables pretrained policies to adapt to test-time spatial and semantic variation through inference-time steering alone.

## VI. CONCLUSION & LIMITATION

We propose VLS, a training-free framework that guides pretrained robotic policies using differentiable rewards generated by Vision-Language Models, addressing the challenge of policy deployment in OOD scenarios. Experiments demonstrate that VLS significantly outperforms existing methods in both simulation and real-world tasks. Limitations of VLS include computational latency: batch sampling, MCMC runs, and FK resampling introduce high inference overhead. Future work may explore progress-aware reward signal generation and optimizing computational efficiency during inference.

## REFERENCES

- [1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.
- [2] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kumimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025.
- [3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolò 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, Haohuan Wang, and Ury Zhilinsky.  $\pi_0$ : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [4] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolò 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, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025.
- [5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.
- [6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montserrat Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.
- [7] Jiahang Cao, Yize Huang, Hanzhong Guo, Rui Zhang, Mu Nan, Weijian Mai, Jiaxu Wang, Hao Cheng, Jingkai Sun, Gang Han, Wen Zhao, Qiang Zhang, Yijie Guo, Qihao Zheng, Chunfeng Song, Xiao Li, Ping Luo, and Andrew F. Luo. Compose your policies! improving diffusion-based or flow-based robot policies via test-time distribution-level composition. *arXiv preprint arXiv:2510.01068*, 2025.

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, October 2021.

[9] 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. *arXiv preprint arXiv:2303.04137*, 2023.

[10] Gabriele Corso, Yilun Xu, Valentin De Bortoli, Regina Barzilay, and T. Jaakkola. Particle guidance: non-i.i.d. diverse sampling with diffusion models. *ArXiv*, abs/2310.13102, 2023. URL <https://api.semanticscholar.org/CorpusID:264405842>.

[11] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[12] Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, 2004.

[13] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

[14] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[15] Danny Driess, Fei Xia, Alexander Sax, Brian Ichter, Keerthana Gopalakrishnan, Jeannette Bohg, Andy Zeng, Chelsea Finn, Sergey Levine, Karol Hausman, et al. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[16] Maximilian Du and Shuran Song. Dynaguide: Steering diffusion policies with active dynamic guidance. *arXiv preprint arXiv:2506.13922*, 2025.

[17] Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Narain Sohl-Dickstein, A. Doucet, and Will Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. *ArXiv*, abs/2302.11552, 2023. URL <https://api.semanticscholar.org/CorpusID:257078922>.

[18] Jiafei Duan, Samson Yu Bai Jian, and Cheston Tan. Space: A simulator for physical interactions and causal learning in 3d environments, 2021. URL <https://arxiv.org/abs/2108.06180>.

[19] Jiafei Duan, Arijit Dasgupta, Jason Fischer, and Cheston Tan. A survey on machine learning approaches for modelling intuitive physics. *arXiv preprint arXiv:2202.06481*, 2022.

[20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. DDPM: Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[22] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[23] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[24] Hyeonseong Jeon, Cheolhong Min, and Jaesik Park. Tree-guided diffusion planner. 2025. URL <https://api.semanticscholar.org/CorpusID:280985003>.

[25] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.

[26] 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.

[27] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. *arXiv preprint arXiv:2110.02711*, 2022.

[28] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag 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. *arXiv preprint arXiv:2406.09246*, 2024.

[29] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, October 2023.

[30] Nishanth Kumar, William Shen, Fabio Ramos, Dieter Fox, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Caelan Reed Garrett. Open-world task and motion planning via vision-language model inferred constraints. *arXiv preprint arXiv:2411.08253*, 2024.

[31] LeRobot Team.  $\pi_{0.5}$  (pi05 libero) (lerobot). [https://huggingface.co/lerobot/pi05\\_libero\\_finetuned](https://huggingface.co/lerobot/pi05_libero_finetuned), 2025. Model checkpoint + documentation (accessed 2026-01-31).

[32] 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. *arXiv preprint arXiv:2511.14178*, 2025.

[33] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *arXiv preprint arXiv:2210.02747*, 2022.

[34] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[35] Oier Mees, Lukáš Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7:7327–7334, 2021. URL <https://api.semanticscholar.org/CorpusID:244908821>.

[36] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

[37] Mitsuhiro Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024.

[38] 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)*, pages 6892–6903. IEEE, 2024.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[40] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

[41] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Manon Gimenez, Yevgenii Sulsky, Jack Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[42] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[43] O. H. Schmitt. A thermionic trigger. *Journal of Scientific Instruments*, 15(1):24–26, 1938.

[44] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ran-

ganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

[45] Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. *arXiv preprint arXiv:2508.05941*, 2025.

[46] Narek Tumanyan, Michael Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. *arXiv preprint arXiv:2211.12572*, 2023.

[47] Andrew Wagenmaker, Mitsuhiro Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[48] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[49] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Perez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. *arXiv preprint arXiv:2411.16627*, 2024.

[50] Yilin Wu, Anqi Li, Tucker Hermans, Fabio Ramos, Andrea Bajcsy, and Claudia Pérez-D'Arpino. Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification. *arXiv preprint arXiv:2510.16281*, 2025.

[51] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: VLM-in-the-loop policy steering via latent alignment. *arXiv preprint arXiv:2502.01828*, 2025.

[52] Hanming Ye. Steering diffusion policies with value-guided denoising. In *OpenReview (Forum Paper)*, 2025. URL <https://openreview.net/forum?id=dtMBW9W5jo>.

[53] Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[54] 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.

[55] Zhengbang Zhu, Ziyan Li, Xiu Yuan, Hanbo Zhang, Yuxiao Liu, Chongjie Zhang, Yong Yu, Weinan Zhang, and Minghuan Liu. Unified latent steering and residual refinement for online improvement of diffusion policy models. In *ICLR 2026 Conference Submission (Open-Review)*, 2025. URL <https://openreview.net/forum?id=DbBD2aT1OG>.