Observing and Controlling Features in Vision-Language-Action Models

Hugo Buurmeijer [1] **Carmen Amo Alonso** [1] **Aiden Swann** [2] **Marco Pavone** [1 3]

Abstract

Vision-Language-Action Models (VLAs) have shown remarkable progress towards embodied intelligence. While their architecture partially resembles that of Large Language Models (LLMs), VLAs exhibit higher complexity due to their multi-modal inputs/outputs and often hybrid nature of transformer and diffusion heads. This is part of the reason why insights from mechanistic interpretability in LLMs, which explain how the internal model representations relate to their output behavior, do not trivially transfer to VLA counterparts. In this work, we study the linear separability of action-relevant features across VLA architectures, and show that observability varies significantly by design: in the transformer-based OpenVLA, action-relevant features are poorly linearly separable, whereas in the hybrid architecture $\pi_{0.5}$, such features are accurately recovered from internal representations. Building on this, we show that lightweight linear interventions grounded in optimal control can reliably steer $\pi_{0.5}$'s behavior while preserving closed-loop capabilities, enabling alignment with user preferences and task requirements without fine-tuning.

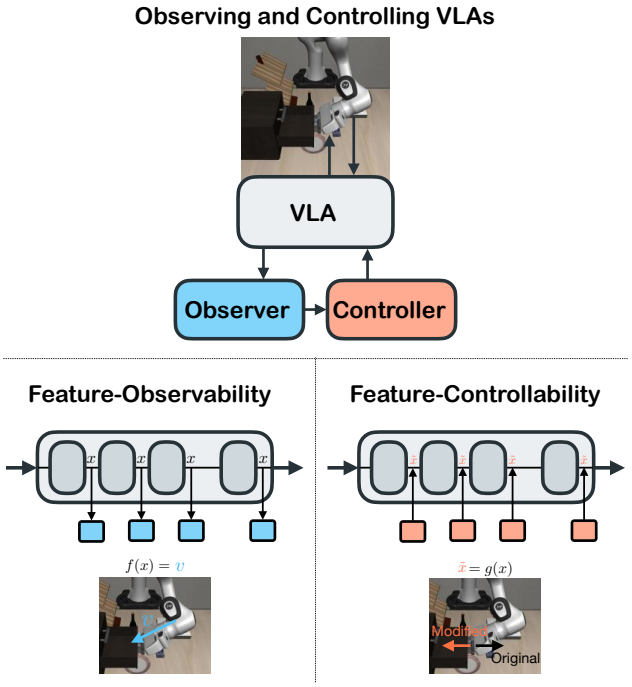

Figure 1. We present the combination of *feature-observability* and *feature-controllability*: a framework to observe and control robot behavior through accessing and modifying internal VLA representations. A linear observer extracts features from the Transformer's internal representations (left), while a minimal linear controller steers these representations to align outputs with desired constraints (right), enabling policy steering without fine-tuning.

1. Introduction

Vision-Language-Action models (VLAs) represent a significant step towards embodied intelligence. By processing multi-modal inputs consisting of images, language instructions, and proprioceptive signals, VLAs combine perception, reasoning, and action generation in a single model. Similar to other disciplines where generative models have unlocked unprecedented degrees of generalization, VLAs enable robots to interpret natural language commands in rich visual contexts and execute complex behaviors, exhibiting improved generalization across tasks and environments (Kim et al., 2025; Intelligence et al., 2025). However, VLAs suffer from similar limitations as other generative models: their behavior can be unpredictable, difficult to correct real-time, or misaligned with user preferences and safety requirements (Fei et al., 2025).

One promising approach to overcoming these limitations is the ability to steer generative models toward desired behaviors while suppressing undesirable ones. In the context of Large Language Models (LLMs), intervening internal model representations to influence outputs has become a prominent research direction, referred to as "activation steering" (Turner et al., 2023). Achieving a similar level of control is arguably even more critical for VLAs, where steering directly affects physical behavior in the real world. Fur-

[1]Department of Aeronautics and Astronautics, Stanford University, Stanford, USA [2]Department of Mechanical Engineering, Stanford University, Stanford, USA [3]NVIDIA Research, Santa Clara, USA. Correspondence to: Hugo Buurmeijer <hbuurmei@stanford.edu>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

thermore, since VLAs often share architectural components with LLMs, (most notably transformer-based representations (Vaswani et al., 2017)), methods developed for LLMs could in theory transfer to VLAs without significant overhead. However, VLAs operate over multiple modalities, including proprioceptive inputs, and must function in closed-loop interaction with the physical world where compounding errors are notorious, thereby posing challenges absent in LLMs. Additionally, robotic tasks such as manipulation naturally demand smooth, continuous actions, which has driven frontier architectures toward hybrid transformer-diffusion designs (Kawaharazuka et al., 2025), further complicating the transfer of interpretability insights from LLMs to VLAs.

To close this gap, several works have explored porting concepts from mechanistic interpretability in LLMs over to the domain of VLAs (Häon et al., 2025; Mitra et al., 2025; Lu et al., 2025). For instance, (Häon et al., 2025) overwrites VLA model activations at inference time to modulate basic motion features such as speed and direction. Overall, the effectiveness of existing methods is still limited. Crucially, while the focus of activation steering methods in LLMs is to control the behavior of the model while preserving its original merits, such as natural-sounding generations, in the field of VLAs interventions that preserve natural behaviors have not been studied. Moreover, the effect of architectural design choices, such as purely transformer-based versus hybrid transformer-diffusion architectures, on the interpretability and steerability of VLA representations remains unexplored.

In this paper, we address these challenges by formalizing concepts that are often implicit in work on steering LLMs, with the goal of making them applicable to VLAs. We introduce a unified framework for analyzing and influencing VLA behavior through their internal representations, and use them to systematically compare common VLA architectures. We find that purely transformer-based architectures such as OpenVLA exhibit poor feature-observability, whereas hybrid transformer-diffusion architectures such as $\pi_{0.5}$ support linear decoding of action-relevant features. We then show how these features can be manipulated in representation space via lightweight linear interventions, with a particular focus on preserving closed-loop success rate.

Concretely, our **contributions** are as follows:

- We formalize the concepts of *feature-observability* and *feature-controllability* for VLAs, grounded in classical notions of observability and controllability, providing a framework to analyze and influence behavior through internal representations. (Section 3)
- We apply this framework to study the linear representation hypothesis in VLAs, and show that feature-observability varies significantly by architecture: purely transformer-based models (OpenVLA) exhibit poor linear separability, while hybrid transformer-diffusion architectures ($\pi_{0.5}$) support approximate linear decoding of action-relevant features. (Section 5.1)
- We propose a linear observer and a minimal linear controller for hybrid VLAs, and show that steering along observed directions robustly affects the model outputs. (Sections 4.1, 4.2)
- We present an online algorithm that integrates the observer and controller in closed-loop operation without requiring fine-tuning or retraining, and validate it in closed-loop simulation on $\pi_{0.5}$. (Sections 4.3, 5)

2. Related Work

2.1. Mechanistic Interpretability in VLAs

The intersection of mechanistic interpretability and robotics, specifically VLAs, is significantly underexplored. Nonetheless, a number of recent works have investigated the use of those tools in the context of end-to-end robot control. Notably, it has been shown that steering vectors can be extracted from the feed-forward layers of the Transformer backbone to causally affect model behaviors such as speed and height (Häon et al., 2025). More abstract concepts, such as spatial relationships between objects and movement targets, were extracted from VLA models in (Lu et al., 2025) by means of linear probes.

Beyond interpretability and behavior steering, recent work has explored leveraging similar tools to improve performance on new tasks. For instance, (Li, 2025) extract task-specific "text latent" vectors from π_0's hidden states that can be used to reconstruct demonstrated behaviors or blended to compose novel skills, improving extrapolation performance from 9% to 83% on out-of-distribution tasks. Furthermore, VLA models can be adapted to new tasks by mechanistically identifying and fine-tuning the task relevant attention heads (Mitra et al., 2025).

2.2. Activation Steering in LLMs

A growing body of work has shown that high-level semantic and behavioral features in LLMs are often linearly represented in their activation spaces, a phenomenon commonly referred to as the *linear separability hypothesis* (Park et al., 2024; 2025). It has been widely observed that simple linear classifiers can reliably extract attributes such as sentiment, topic, or personality traits from intermediate transformer layers (Cheng & Alonso, 2024; Rodriguez et al., 2025; Chen et al., 2025), suggesting that these representations encode disentangled, behaviorally meaningful features. Building on this insight, *activation steering* methods modify internal activations at inference time to influence model outputs without fine-tuning (Wu et al., 2024; Sun et al., 2025). Techniques such as activation addition (Turner et al., 2023; Li et al.,

2023) have been applied to guide generation toward desired attributes, mitigate harmful behaviors, and improve safety and alignment. Crucially, these methods emphasize preserving the naturalness and coherence of generated text by applying low-rank or minimal perturbations to the model's internal states.

3. Problem Statement

In this section, we present the problem of how the internal representations in VLAs can be controlled directly to align their outputs with user preferences. To that end, we describe the internal structure of two different families of VLA architectures, as captured by OpenVLA (Kim et al., 2025) and $\pi_{0.5}$ (Intelligence et al., 2025). We then introduce two concepts that enable the formalization of the problem in mathematical terms.

3.1. Visual-Language-Action Models (VLAs)

The goal is to formalize the study of internal representations in VLAs. Here we present two different types of VLA architectures: (a) transformer-based, such as OpenVLA (Kim et al., 2025), and (b) transformer-flow-matching hybrids, such as $\pi_{0.5}$ (Intelligence et al., 2025).

Transformer-based VLAs This architecture consists of a Vision-Language-Model autoregressive transformer architecture, where visual observations (processed through vision encoders such as DINOv2 and SigLIP) and language instructions are encoded into a shared token space, and a language model backbone (e.g., Llama 2) predicts tokenized output actions autoregressively, which are then decoded into robot commands. Examples of this architecture are OpenVLA (Kim et al., 2025) and RT2 (Zitkovich et al., 2023).

Transformer-Flow-Matching hybrid VLAs This architecture consists of an autorregressive transformer architecture, where a pretrained Vision-Language Model (VLM) processes visual and language inputs, while a separate "action expert" transformer uses conditional flow matching to generate continuous, high-frequency action trajectories. An example of this architecture is the family of π models, such as π_0 (Black et al., 2026) and $\pi_{0.5}$ (Intelligence et al., 2025).

We note that since the focus of this work is to port existing tools and insights from LLMs into VLAs, we focus on the transformer architectural component part present in VLA architectures. To that end, the remainder of the paper is focus on *internal representations inside the transformer block*. Yet, we show that leveraging these internals directly relates to relevant action features of the VLA output.

We formalize the Transformer architecture (Vaswani et al., 2017) as a sequence of transformations (layers) acting on a

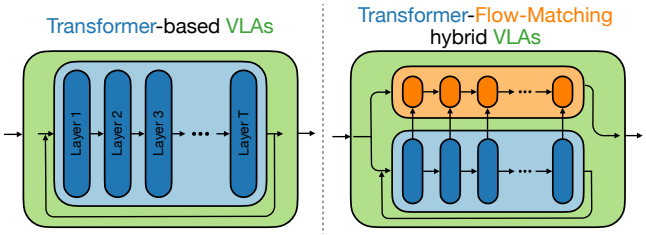

Figure 2. Schematic representation of two VLA architectures. On the left, a *Transformer-based VLA* features an autoregressive Transformer architecture. On the right, a *Transformer-Flow-Matching hybrid VLA* is composed of a Transformer architecture and a Flow Matching block (referred to as the "action expert"), where the Flow Matching layers attend to their paired counterparts in the Transformer.

latent representation of an input sequence, denoted as $\mathcal{I}$ (e.g., images, text and proprioceptive inputs). The model first maps this input sequence into a continuous representation via an embedding map E, producing an initial hidden state

$$x_0 = E(\mathcal{I}),\ x_0 \in \mathbb{R}^d. \tag{1}$$

The model then applies a sequence of T Transformer layers. Each layer $L_\ell : \mathbb{R}^d \to \mathbb{R}^d$ transforms the hidden state from the previous layer, yielding the recurrence

$$x_{\ell+1} = L_{\ell+1}(x_\ell), \qquad \ell = 0, \ldots, T-1, \tag{2}$$

where $x_\ell \in \mathbb{R}^d$ is the latent representation after the ℓ-th layer.

We work under an assumption on the VLA representation space that is analogous to common assumptions made for LLM representations. Namely, we assume that the latent representation x_ℓ encodes information about a set of important features of the robot's behavior. That is, there exists an (unknown) mapping from the hidden state to these features, reflecting the common assumption in LLMs that different dimensions or subspaces of the representation capture distinct aspects of the input. Examples of features include, for instance, the actions and states of the robot. We denote features as $\zeta \in \mathbb{R}^n$.

Since the Transformer architecture is embedded into a VLA model, the action output a is computed using the internal representations, i.e., $a = \phi(x_1, \ldots, x_T)$, where the map ϕ depends on the choice of architecture: for the transformer-based architecture ϕ depends only on the final layer representation, x_T, while for the transformer-flow-matching hybrid architecture ϕ includes the flow matching process that can be conditioned on intermediate layer representations $x_1, \ldots, x_T$.

3.2. Observability and Controllability of VLA Features

In this paper, we explore how the structure of the representation space in VLAs can be leveraged to (i) observe

features, such as states or actions, and (ii) steer features into a desired region by controlling their representations. To that end, we introduce the concepts of *feature-observability* and *feature-controllability*.

Definition 3.1 (Feature-Observability)**.** Given the activation $x_\ell \in \mathbb{R}^d$ in layer ℓ of a Transformer architecture, a *feature* $\zeta \in \mathbb{R}^n$ is *observable at layer* ℓ if there exists a map (referred to as the *observer*) $f_\ell : \mathbb{R}^d \to \mathbb{R}^n;\ x \mapsto \zeta$ such that $f_\ell(x_\ell) = \zeta$.

Definition 3.2 (Feature-Controllability)**.** Given a desired set $\mathcal{D} \in \mathbb{R}^d$, a *feature* $\zeta \in \mathbb{R}^n$ is *controllable at layer* ℓ of a Transformer architecture if there exists a map (referred to as the *controller*) $g_\ell : \mathbb{R}^d \to \mathbb{R}^d;\ x \mapsto \tilde{x}$ such that when the modified intervention $\tilde{x}_\ell \in \mathbb{R}^d$ is propagated through layers $\ell, \dots, T$ via Equations (1) and (2), it leads to $\zeta \in \mathcal{D}$.

Given the definitions above, the open question is how to design observer, $f(\cdot)$, and controller maps, $g(\cdot)$, that enable feature-observability and feature-controllability for relevant features in representation space, and how to use them effectively during VLA generations. In what follows, we propose that linear maps are sufficient to effectively observe features and align VLA's outputs towards user preferences.

4. Approach

In this section, we formalize our approach through the two central concepts: *feature-observability* and *feature-controllability* previously introduced. Specifically, we propose a linear observer structure that allows for information about interpretable features in the model to be extracted from its internal state (feature observer); and a linear controller structure that steers features in the model's internal representations through interventions to its internal state (feature controller). We then show how these proposals enable efficient feature-observability and -controllability at inference.

4.1. Designing the Feature Observer

We design the observer map at layer ℓ, $f_\ell(\cdot)$, to be a linear function. This is, given a representation $x_\ell \in \mathbb{R}^d$, a feature of interest $\zeta \in \mathbb{R}^n$ can be found as $\zeta = f_\ell(x_\ell)$ with

$$f_\ell(x) := W_\ell x + b_\ell, \tag{3}$$

where $W_\ell \in \mathbb{R}^{d\times n}$ and $b_\ell \in \mathbb{R}^n$ are learned parameters.

The weights $W_\ell \in \mathbb{R}^{d\times n}$ and the bias $b_\ell \in \mathbb{R}^n$ are learned from data. In particular, we leverage pairs of inputs ($\mathcal{I}$ containing images, prompt and potentially robot states) to a feature of interest (denoted as $\zeta \in \mathbb{R}^n$). Given a set of input-feature pairs $\{\mathcal{I}^{(i)}, \zeta^{(i)}\}_{i=1}^N$, a set of activation-feature pairs $\{x_\ell^{(i)}, \zeta^{(i)}\}_{i=1}^N$ can be collected at each layer ℓ by simply propagating each input sequence $\mathcal{I}^{(i)}$ up until layer ℓ through the transformer architecture via Equations (1) and (2), whose output is $x_\ell^{(i)}$. Then, W_ℓ and b_ℓ are learned by minimizing a task-appropriate loss over $\{x_\ell^{(i)}, \zeta^{(i)}\}_{i=1}^N$:

$$W_\ell, b_\ell = \underset{W_\ell, b_\ell}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\big(f_\ell(x_\ell^{(i)}), \zeta^{(i)}\big), \tag{4}$$

where $\mathcal{L}$ is the mean squared error loss for continuous features ($\zeta \in \mathbb{R}^n$) and the cross-entropy loss for discrete features ($\zeta \in \{0,1\}^n$). Algorithm 1 summarizes the steps to train the observer at layer ℓ, $f_\ell(\cdot)$.

Algorithm 1 Learning linear observer f_ℓ (Eq. (3)) at layer ℓ

Input: Labeled dataset $\{(\mathcal{I}^{(i)}, \zeta^{(i)})\}_{i=1}^N$
Output: $W_\ell \in \mathbb{R}^{d\times n}$ and $b_\ell \in \mathbb{R}^n$
for $i = 0, \dots, N$ **do** ▷ In parallel for all $i = 1, \dots, N$
  $x_0^{(i)} \leftarrow$ Eq. (1) ▷ Compute initial embedding
  for $\ell = 0, \dots, T-1$ **do** ▷ Until layer T
    $x_{\ell+1}^{(i)} \leftarrow$ Eq. (2) ▷ Get layer ℓ representation
  end for
end for
$(W_\ell, b_\ell) \leftarrow$ Eq. (4) ▷ Train linear observer at layer ℓ
return W_ℓ, b_ℓ

The choice of a linear map is motivated by the linear separability hypothesis, widely explored in LLMs (Park et al., 2024). We note that the presented method holds for generalized versions of the linear observer, such as $f_\ell(x) = \nu(W_\ell x + b_\ell)$, where ν is some known monotonic nonlinearity of choice.

Remark 4.1. We highlight that finding an observer f_ℓ for a given layer ℓ via Algorithm 1 does not guarantee robustness of such observer. This is, in theory, an epsilon perturbation to internal representation x_ℓ could lead to an arbitrarily large change in the estimated feature ζ. For this reason, after learning the observer f_ℓ via Algorithm 1, we empirically verify that given $\epsilon > 0$, then $\|f_\ell(x+\epsilon) - f_\ell(x)\| < \delta$ for some $\delta > 0$.

4.2. Designing the Feature Controller

We design the controller map at layer ℓ, $g_\ell(\cdot)$, to be a linear intervention on the layer representation. This is, given a representation $x_\ell \in \mathbb{R}^d$, the modified representation is computed as $\tilde{x}_\ell = g_\ell(x_\ell)$ with

$$g_\ell(x) := x + u_\ell, \tag{5}$$

where u_ℓ is an additive perturbation to the representation. Similar to prior work in LLMs (Cheng & Alonso, 2024), we propose to compute u_ℓ as the solution to the optimization

problem:

$$u_\ell = \underset{u \in \mathbb{R}^d}{\text{argmin}} \quad \|u\|_2^2 \tag{6a}$$

$$s.t. \quad f_\ell(x_\ell + u) \in \mathcal{D}, \tag{6b}$$

where $\mathcal{D} \subset \mathbb{R}^n$ is the desired set of feature values, and $f_\ell(\cdot)$ is the observer map at layer ℓ. In words, u_ℓ is the minimal additive intervention such that the resulting modified representation $x_\ell + u_\ell$ lies in the preimage of the desired feature set under the observer. Intuitively, u_ℓ is the smallest control input in representation space that steers the observed features into a target region.

In general, the constraint in (6b) may be nonconvex, depending on the form of f_ℓ and the geometry of $\mathcal{D}$. To obtain a tractable solution, we use a linear observer as per Equation (3), and we assume that $\mathcal{D} = [\zeta_{\min}, \zeta_{\max}] \subset \mathbb{R}$, i.e., some lower and upper bound for one-dimensional ζ values. As shown in (Cheng & Alonso, 2024) (Theorem 4.1), under these assumptions the solution to 6 can be computed in closed form as:

$$u_\ell = (\zeta_{\max} - \zeta_\ell)\frac{W_\ell}{\|W_\ell\|_2^2}, \qquad \text{if } \zeta_\ell > \zeta_{\max}, \tag{7a}$$

$$u_\ell = (\zeta_{\min} - \zeta_\ell)\frac{W_\ell}{\|W_\ell\|_2^2}, \qquad \text{if } \zeta_\ell < \zeta_{\min}, \tag{7b}$$

$$u_\ell = 0, \qquad \text{otherwise}, \tag{7c}$$

where $\zeta_\ell = f(x_\ell) = W_\ell^\top x_\ell + b_\ell$ is the layer ℓ observation.

We note that the proposed controller leverages the observer. Although feature-observability and feature-controllability are independent properties, i.e. a feature can be observable but not controllable, and vice-versa (as is standard in control theory (Kalman, 1960)), more effective control can be achieved when a feature is both observable and controllable. In the LLM literature, while some activation steering proposals inject additive interventions into internal representations without explicit observers (Turner et al., 2023), their performance is surpassed by those which rely on past observations using labeled data (Rodriguez et al., 2025; Cheng & Alonso, 2024). For this reason, the proposed feature controller relies on observations of the feature.

4.3. Observing and Controlling Features in VLAs

We illustrate how to leverage the proposed feature observer and feature controller in practice to align VLA outputs with user preferences. Given

- a *feature observer*, $f_\ell(\cdot)$, with linear structure as defined in Equation (3), where linear parameters W_ℓ and b_ℓ are computed before inference time via Algorithm 1; and
- a *feature controller*, $g_\ell(\cdot)$, with linear structure as defined in Equation (5), where the linear parameter u_ℓ is computed by evaluating the representation x_ℓ according to Equation 7,

the goal is to integrate them into the inference time computations at layer ℓ to achieve precise representation control and downstream alignment of the VLA output.

Let us consider two set of layers: (i) $\mathcal{L}_\mathcal{O} \subseteq [0, \dots, T]$ where feature-observability is desired, and (ii) $\mathcal{L}_\mathcal{C} \subseteq [0, \dots, T]$ where feature-controllability is desired. We note that, given that the proposed controller requires observations, $\mathcal{L}_\mathcal{C} \subseteq \mathcal{L}_\mathcal{O}$. The inference-time computation of a Transformer architecture's forward pass equipped with observer and controller for feature ζ is described by Algorithm 2.

Algorithm 2 Inference forward-pass in a VLA-embedded Transformer architecture with ζ-observer and ζ-controller

1: **Input:** $\mathcal{I}$, $\mathcal{L}_\mathcal{O}$, $\mathcal{L}_\mathcal{C}$, $W_{\ell \in \mathcal{L}_\mathcal{O}}$, $b_{\ell \in \mathcal{L}_\mathcal{O}}$, $\zeta_{\min}$, $\zeta_{\max}$
2: **Output:** $x_1, \dots, x_T$
3: $x_0 \leftarrow$ Eq. (1) ▷ Compute initial embedding
4: **for** $\ell = 1, \dots, T$ **do** ▷ **Forward-pass through layers**
5: $\quad x_\ell \leftarrow$ Eq. (2) ▷ Compute layer ℓ representation
6: $\quad$ **if** $\ell \in \mathcal{L}_\mathcal{O}$ **then** ▷ **Feature-observability**
7: $\quad\quad \zeta_\ell \leftarrow$ Eq. (3) ▷ Compute observation
8: $\quad\quad$ **if** $\ell \in \mathcal{L}_\mathcal{C}$ **then** ▷ **Feature-controllability**
9: $\quad\quad\quad u_\ell \leftarrow$ Eq. (7) ▷ Compute control input
10: $\quad\quad\quad x_\ell \leftarrow$ Eq. (5) ▷ Apply intervention
11: $\quad\quad$ **end if**
12: $\quad$ **end if**
13: **end for**
14: **return** $x_1, \dots, x_T$

The forward-pass presented in Algorithm 2 only differs from a standard un-observed and un-intervened Transformer forward-pass in steps 6-12. A crucial advantage of choosing a linear observer and a linear closed-form controller is that computations in step 7, 8 and 9 introduce minimal overhead, which results in a negligible increase in runtime while equipping the model with effective steering capabilities.

5. Experiments

In this section, we present the experiments that explore the concepts of feature-observability and feature-controllability in two frontier VLA policies: the transformer-based OpenVLA and the transformer-flow-matching hybrid model $\pi_{0.5}$. We first compare how accurately robot states and actions can be observed in representation space using a linear observer for these two architectures. Then, for the hybrid architecture, we show that these observations are robust to perturbations in the representation space. Lastly, we study how minimal control interventions to the representation space lead to ef-

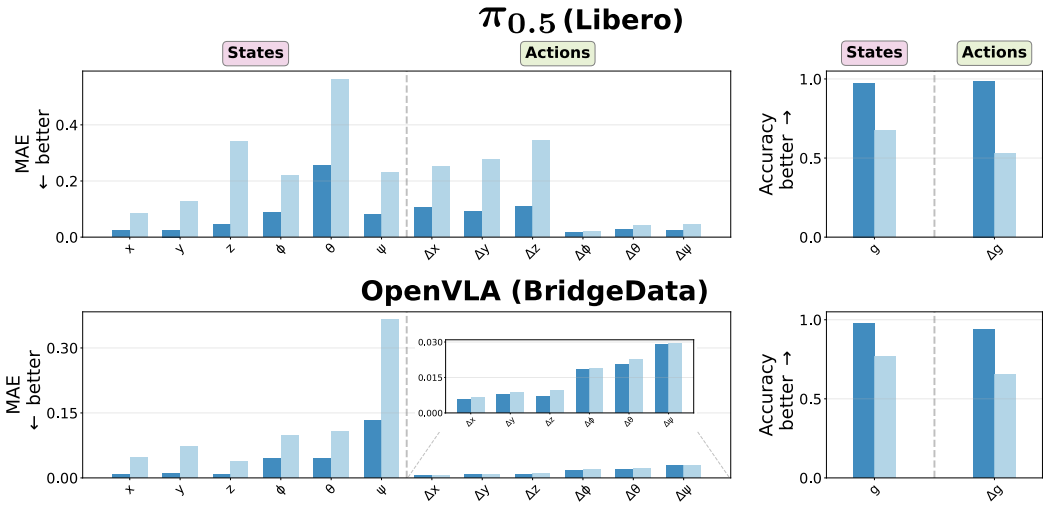

Figure 3. On the top, results from training a linear classifier via Algorithm 1 on the $\pi_{0.5}$ transformer layers on test data from Libero dataset. Left: the maximum absolute error (MAE) of the trained classifier compared with the MAE from mean training prediction, used as a baseline. Right: accuracy of the trained classifier compared with accuracy from majority class prediction, used as a baseline. On the bottom, identical representation, using OpenVLA model on the BridgeData V2 dataset. In all cases, only results for the best performing layer are displayed.

fective steering of the VLA output and better alignment with user preferences and safety requirements.

5.1. Feature-Observability

The goal is to asses (i) whether robot-relevant features can be observed with a linear classifier as defined in Equation (3); and (ii) whether these observations are robust, i.e., given $\epsilon > 0$, then $\|f_\ell(x+\epsilon) - f_\ell(x)\| < \delta$ for some $\delta > 0$.

We perform these experiments by testing $\pi_{0.5}$ (Intelligence et al., 2025) on the Libero dataset (Liu et al., 2023), and OpenVLA (Kim et al., 2025) on the BridgeData V2 dataset (Walke et al., 2023). For this first set of experiments, we focus on features of interest being robot states and actions, and provide a case study on a more abstract feature in Section 5.3. This choice is well-motivated: robot states and actions constitute the fundamental observation and control space for robotic systems, are directly measurable and actuatable, and generalize naturally across diverse tasks. Practically, these features are also readily available in common robotic datasets. Formally, we define the robot state space as $s = (x, y, z, \phi, \theta, \psi, g)$, where $(x, y, z) \in \mathbb{R}^3$ represent Cartesian position, $(\phi, \theta, \psi) \in [0, 2\pi)^3$ encode roll, pitch, and yaw orientations, and $g \in [0, 1]$ represents normalized gripper aperture. Robot actions $a = \Delta s \in \mathcal{A}$ correspond to relative displacements in this state space.

To learn the linear classifier $f(\cdot)$ in Equation (3), we train a probe via Algorithm 1 on activations collected from each dataset. Specifically, an input sequence $\mathcal{I}$ consists of images capturing the scene, the task instruction prompt, and proprioceptive inputs; for OpenVLA, only a single image observation is used with no proprioceptive measurements. Features correspond to robot states and actions, for instance $\zeta = \Delta x \in \mathbb{R}$, yielding a dataset of input-feature pairs $\{\mathcal{I}^{(i)}, \zeta^{(i)}\}_{i=1}^N$. Inputs are passed through the frozen model and activations are collected at each layer ℓ, providing activation-feature pairs $\{x_\ell^{(i)}, \zeta^{(i)}\}_{i=1}^N$ for probe training. For $\pi_{0.5}$, activations are collected from the VLM backbone (prefix) only, excluding the action expert. For OpenVLA, activations are collected at the first forward pass, prior to autoregressive action generation. Cartesian positions and orientations are probed with a regression probe using continuous labels, while gripper state and action are probed with a binary classifier. Probes are trained at every layer $\ell \in [1, \ldots, T]$, and Figure 3 reports the best-performing probe across layers for each state and action feature.

Figure 3 reveals a consistent pattern across both architectures: proprioceptive states and gripper actions are approximately linearly decodable from transformer representations in both $\pi_{0.5}$ and OpenVLA. However, a divergence emerges for continuous actions: $\pi_{0.5}$ probes substantially outperform the constant mean predictor for Cartesian position and orientation actions, whereas OpenVLA probes fail to improve over this baseline, indicating that linear structure for these features is effectively absent in OpenVLA representations. We attribute this difference primarily to the architectural distinction between the two models. In $\pi_{0.5}$, the VLM backbone conditions the action expert, which must extract and refine action-relevant information from these representations into continuous outputs. This creates an inductive bias toward action-relevant features being explicitly organized in the VLM's representation space. In OpenVLA, by contrast,

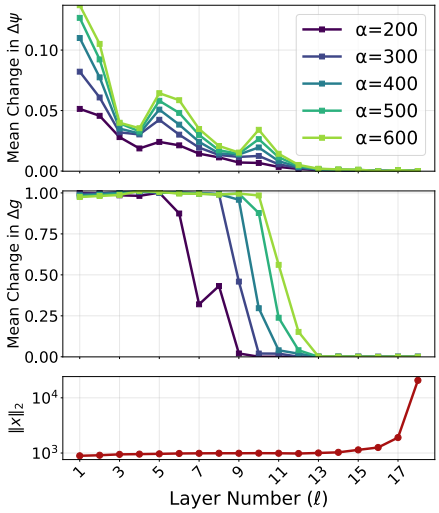

Figure 4. Effect of linear interventions on $\pi_{0.5}$ representations at different layers, using LIBERO episodes. Top: mean change in delta yaw action across episodes as a function of intervention layer, for varying perturbation strengths α. Middle: same for delta gripper action. Bottom: L_2-norm of the representation vector across layers, showing that representation magnitude increases with depth, explaining the diminishing effect of a fixed perturbation at deeper layers.

the transformer directly generates discrete action tokens autoregressively, with no separate module that needs to decode action-relevant structure from intermediate representations, hence providing little incentive for such structure to emerge in a linearly separable form. We note, however, that $\pi_{0.5}$ and OpenVLA differ not only in architecture but also in training data and recipe, which constitute confounding factors that preclude a strictly controlled architectural comparison.

Having established feature-observability for $\pi_{0.5}$, we further analyze the robustness of its linear observers by perturbing the representation at layer ℓ as $x_\ell \leftarrow x_\ell + \alpha W_\ell$, where W_ℓ is the linear observer direction and α controls the perturbation strength. Only a single layer is perturbed per generation, and we study the mean change in the observed action feature across episodes. As shown in Figure 4, increasing α yields a smooth, monotonic change in the output action for both observers, and this pattern holds consistently across all action features. This confirms that the linear observer directions are actionable: perturbing representations along W_ℓ reliably modifies the corresponding output feature, which we exploit for control in the next subsection. Notably, perturbations are more effective in earlier layers, as the L_2-norm of the representation grows with depth, diminishing the relative effect of a fixed α.

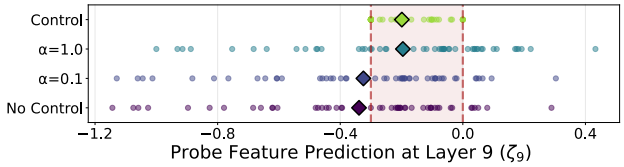

Figure 5. Representations projected into the classifier image space at layer 9 of $\pi_{0.5}$, under different interventions. Individual samples are shown with means indicated by solid diamonds. The proposed controller (Eq. (7)) constrains the intervened representations to lie within the target bounds $[\zeta_{\min}, \zeta_{\max}]$, while baseline interventions and un-intervened representations fall outside.

5.2. Feature-Controllability

We now ask whether targeted linear interventions can reliably steer the observable features at inference time. Specifically, we assess (i) whether the internal representations of the VLA can be controlled via a linear intervention as defined in Equation (5), and (ii) whether such interventions result in accurate steering of relevant robotic features.

To test the effectiveness of the minimal controller proposed in Equation (7), we observe (via the observer in Equation (3)) where the representation lies in the classifier image. In particular, without intervention the projected representations are spread out, and a fixed additive perturbation αW_ℓ, as used in several activation steering methods (Turner et al., 2023), preserves this spread rather than concentrating representations toward the target. However, the proposed intervention in Equation (7) *guarantees* that the image of the intervened representation lies within the desired bounds $[\zeta_{min}, \zeta_{max}]$, since it is a hard constraint in optimization (6). Figure 5 demonstrates this empirically.

While Figure 5 confirms that the controller satisfies the representational constraint by construction, the more critical question is whether this translates to reliable closed-loop steering without sacrificing task success. To evaluate this, all experiments are based on the LIBERO simulator (Liu et al., 2023), a standardized benchmark for evaluating vision-language-action models on robotic manipulation tasks. Inference is performed on a single NVIDIA RTX 5090 GPU. Specifically, we evaluate on ten tasks from the spatial task suite, performing ten rollouts per task for each method.

For all features we analyze both the absolute steering performance, and the trade off between constraint satisfaction and closed-loop success rate. While previous work has primarily focused on constraint satisfaction or steerability in isolation (Häon et al., 2025), we strongly argue that steering performance is only meaningful when considered in conjunction with closed-loop task success rate—analogous to how language model steering methods must preserve response naturalness and coherence.

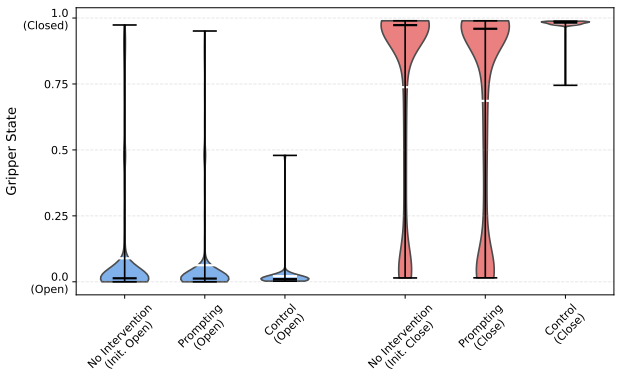

Figure 6. Violin plots of normalized gripper state during the constraint window, comparing no intervention, prompting, and control under open-initialized (blue) and closed-initialized (red) conditions. Values are normalized to [0,1] where 0 denotes open and 1 denotes closed; means (white) and medians (black) are shown.

We study three low-level action features: gripper, end-effector height, and end-effector speed; full results are provided in Appendix A. Here we highlight the gripper state, which we aim to force to be opened or closed, as a representative example. Figure 6 shows strong steerability for both "open" and "closed" target states, compared against no intervention and prompting baselines. Our method achieves near-perfect steering while maintaining a task success rate above 90%. In the next section, we show that the framework extends to more abstract, semantically meaningful features.

5.3. Case Study: Steering Grasping Modalities

We consider the simple task of "pick up the wine bottle," which we execute in the LIBERO simulator with $\pi_{0.5}$-LIBERO as the base policy. In the LIBERO dataset, this task is well represented as a subtask of 83 expert demonstrations. These demonstrations consist of two distinct modalities to solve this task: 1) a stable grasp at the base of the bottle where it is approached from the side, and 2) an unstable grasp at the top, approached from above. Given that the robot is equipped with a two-fingered gripper, the second grasp modality is undesirable, and the user will want to suppress this behavior at deployment (without full retraining).

Interestingly, while the grasp modalities are nearly equally represented in the training data (57% unstable grasps), the base policy strongly prefers the unstable mode, as shown in Table 1. In typical usage, one might decide to adjust the task description to bias the policy towards the user-aligned modality, which we refer to as *prompting*. We try two variations of prompts: "pick up the wine bottle at the base", which emphasizes the location of the grasp, and "pick up the wine bottle from the side", which emphasizes the grasp approach. This approach improves the ratio of stable grasps, yet fails to make it the dominant modality, as reflected in Table 1 (rows 2 and 3). Instead, if we learn simple, linear observers to detect this modality (with close to 100% test accuracy), and control the policy's representations, we render the stable gasp as the primary modality (while maintaining perfect task success rate). We note that using, for instance, safety filters to steer the robot's behavior would be unable to alter such a behavior modality effectively without excessive handcrafting.

Table 1. Grasp modality distribution for wine bottle manipulation across 100 sampled initial conditions in the `LIBERO GOAL` scene.

	Stable (Base) Grasp	Unstable (Top) Grasp
Base policy ($\pi_{0.5}$)	28%	72%
Prompting ("base")	40%	60%
Prompting ("side")	34%	66%
Ours	**73%**	**27%**

6. Conclusion and Outlook

This paper introduced a principled framework for understanding and controlling Vision-Language-Action models through their internal representations. We used this framework to study how architectural design choices affect the interpretability and steerability of VLA internal representations, and leveraged these insights to enable reliable inference-time steering. By comparing a transformer-only architecture (OpenVLA) with a hybrid transformer-diffusion architecture ($\pi_{0.5}$), we found that action-relevant features are approximately linearly decodable in the latter but not the former, which we attribute to the functional role of the VLM backbone in hybrid designs. Building on this, we proposed lightweight linear observers and controllers that reliably steer robot behavior while preserving closed-loop task performance (without fine-tuning or retraining), and with negligible computational overhead. We validated this approach in closed loop using the LIBERO benchmark, demonstrating real-time alignment of VLA outputs with user preferences and task constraints.

Limitations and Future Work. While our framework successfully demonstrates feature observability and controllability, several directions warrant further investigation. First, our current approach requires labeled data to train linear observers, which may not always be available in large-scale robotics datasets. Future work could explore self-supervised or unsupervised methods for feature discovery, such as leveraging Sparse Autoencoders to identify interpretable features without explicit labels. Second, we focused primarily on the transformer components of VLA architectures; extending our framework to the diffusion or flow-matching heads would enable end-to-end interpretability and control across hybrid architectures. Finally, establishing principled safety guarantees and bounds on the effects of representation-space interventions would be crucial for deploying these techniques in safety-critical robotic applications.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L. X., Tanner, J., Vuong, Q., Walling, A., Wang, H., and Zhilinsky, U. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Chen, R., Arditi, A., Sleight, H., Evans, O., and Lindsey, J. Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint arXiv:2507.21509*, 2025.

Cheng, E. and Alonso, C. A. Linearly controlled language generation with performative guarantees. *arXiv preprint arXiv:2405.15454*, 2024.

Fei, S., Wang, S., Shi, J., Dai, Z., Cai, J., Qian, P., Ji, L., He, X., Zhang, S., Fei, Z., et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Häon, B., Stocking, K., Chuang, I., and Tomlin, C. Mechanistic interpretability for steering vision-language-action models. *arXiv preprint arXiv:2509.00328*, 2025.

Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Kalman, R. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Kawaharazuka, K., Oh, J., Yamada, J., Posner, I., and Zhu, Y. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 13:162467–162504, 2025. doi: 10.1109/ACCESS.2025.3609980.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E. P., Sanketi, P. R., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., and Finn, C. Openvla: An open-source vision-language-action model. In Agrawal, P., Kroemer, O., and Burgard, W. (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Li, K., Patel, O., Viégas, F., Pfister, H., and Wattenberg, M. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Li, Q. Task reconstruction and extrapolation for π_0 using text latent. *arXiv preprint arXiv:2505.03500*, 2025.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Lu, H., Li, H., Shahani, P. S., Herbers, S., and Scheutz, M. Probing a vision-language-action model for symbolic states and integration into a cognitive architecture. *arXiv preprint arXiv:2502.04558*, 2025.

Mitra, C., Luo, Y., Saravanan, R., Niu, D., Pai, A., Thomason, J., Darrell, T., Anwar, A., Ramanan, D., and Herzig, R. Mechanistic finetuning of vision-language-action models via few-shot demonstrations. *arXiv preprint arXiv:2511.22697*, 2025.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. In *International Conference on Machine Learning*, pp. 39643–39666. PMLR, 2024.

Park, K., Choe, Y. J., Jiang, Y., and Veitch, V. The geometry of categorical and hierarchical concepts in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Rodriguez, P., Blaas, A., Klein, M., Zappella, L., Apostoloff, N., Suau, X., et al. Controlling language and diffusion models by transporting activations. In *The Thirteenth International Conference on Learning Representations*, 2025.

Sun, J., Baskaran, S., Wu, Z., Sklar, M., Potts, C., and Geiger, A. Hypersteer: Activation steering at scale with hypernetworks. *arXiv preprint arXiv:2506.03292*, 2025.

Turner, A. M., Thiergart, L., Udell, D., Leech, G., Mini, U., and MacDiarmid, M. Activation addition: Steering language models without optimization. *CoRR*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Walke, H., Black, K., Lee, A., Kim, M. J., Du, M., Zheng, C., Zhao, T., Hansen-Estruch, P., Vuong, Q., He, A., Myers, V., Fang, K., Finn, C., and Levine, S. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Wu, Z., Arora, A., Wang, Z., Geiger, A., Jurafsky, D., Manning, C. D., and Potts, C. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., Vuong, Q., Vanhoucke, V., Tran, H., Soricut, R., Singh, A., Singh, J., Sermanet, P., Sanketi, P. R., Salazar, G., Ryoo, M. S., Reymann, K., Rao, K., Pertsch, K., Mordatch, I., Michalewski, H., Lu, Y., Levine, S., Lee, L., Lee, T.-W. E., Leal, I., Kuang, Y., Kalashnikov, D., Julian, R., Joshi, N. J., Irpan, A., Ichter, B., Hsu, J., Herzog, A., Hausman, K., Gopalakrishnan, K., Fu, C., Florence, P., Finn, C., Dubey, K. A., Driess, D., Ding, T., Choromanski, K. M., Chen, X., Chebotar, Y., Carbajal, J., Brown, N., Brohan, A., Arenas, M. G., and Han, K. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Tan, J., Toussaint, M., and Darvish, K. (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A. Closed-Loop Results for Additional Features

We present closed-loop steering results for two additional low-level features: end-effector height and end-effector speed. Together with the gripper state results in Figure 6, these experiments validate the proposed framework across a diverse set of essential robotic features.

End-effector height. Figure 7 shows that end-effector height can be reliably steered toward both low and high target conditions, with near-perfect constraint satisfaction. As shown in Figure 8, a modest drop in task success rate is observed relative to the unconstrained baseline, which is expected since the constrained task is strictly harder. We hypothesize that for a sufficiently robust base policy with strong recovery behaviors, this reduction could be further mitigated.

End-effector speed. Speed is a derived feature, computed as $v = \|\Delta x, \Delta y, \Delta z\|/dt$, and is not directly predicted by the model. As shown in Figure 9, the proposed method reliably steers the robot toward lower speeds, but is less accurate in inducing higher speeds. We attribute this asymmetry to limited training data in the fast-speed regime, consistent with observations in (Häon et al., 2025). Notably, Figure 10 shows that task success rates are maintained nearly perfectly when controlling speed, suggesting that this feature can be steered with minimal disruption to closed-loop behavior.

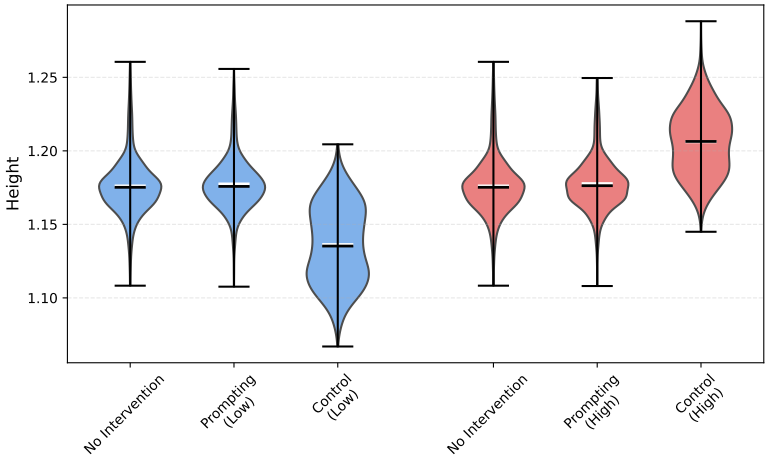

Figure 7. Violin plots of normalized end-effector height over the first 15 constrained steps, comparing no intervention, prompting, and control under low (blue) and high (red) target conditions. Means (white) and medians (black) are overlaid.

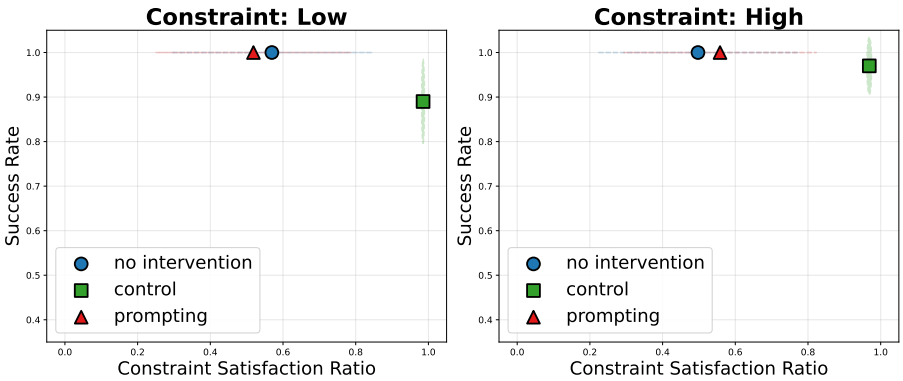

Figure 8. Task success rate vs. constraint satisfaction for end-effector height, comparing no intervention, prompting, and the proposed method. Left: constrain trajectory below initial height. Right: constrain trajectory above initial height. Constraints are active for the first 15 timesteps. Ellipses denote standard deviations across tasks and episodes.

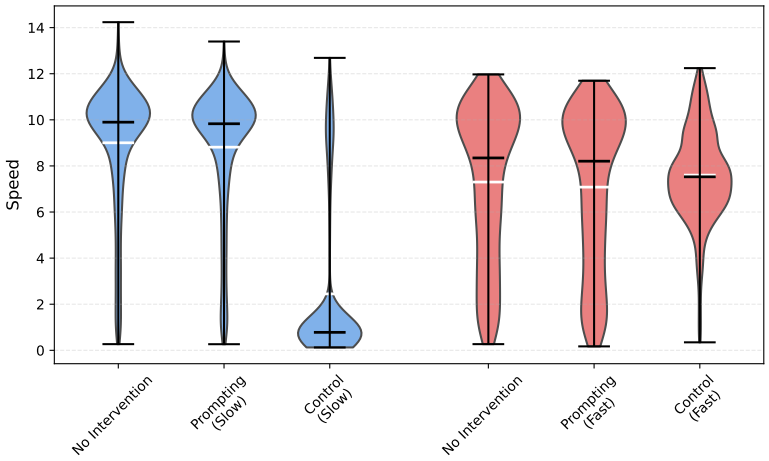

Figure 9. Violin plots of normalized end-effector speed over the first 25 constrained steps, comparing no intervention, prompting, and control under low (blue) and high (red) target conditions. Means (white) and medians (black) are overlaid.

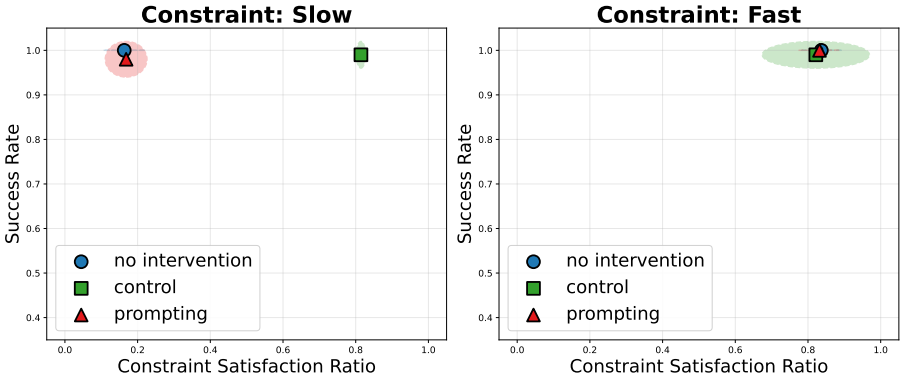

Figure 10. Task success rate vs. constraint satisfaction for end-effector speed, comparing no intervention, prompting, and the proposed method. Left: constrain trajectory below median training speed. Right: constrain trajectory above median training speed. Constraints are active for the first 25 timesteps. Ellipses denote standard deviations across tasks and episodes.