THE MISSING TEMPORAL DIMENSION IN FLOW-BASED ACTION CHUNKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Action chunking has emerged as a cornerstone technique in robotic control and imitation learning. Recent robotic control policies predominantly employ action-chunked policies trained with flow matching objectives, achieving substantial performance gains. Despite this success, existing approaches represent action chunks merely as unstructured vectors during flow-based generation, overlooking the physical-time dependencies among actions within each chunk and potentially leading to ineffective policy learning. In this paper, we show that additionally supervising the first-order temporal derivatives of actions within a chunk provides a surprisingly effective way to incorporate physical-time dependencies into flow-based policies. We further establish that this supervision scheme corresponds to a special case of path-independent flow matching, in which action chunk generation evolves jointly along two dimensions: denoising time and physical time. By capturing the joint effects of multi-step denoising and the physical-time evolution of actions, our method enables fully exploiting the rich temporal structure of action chunks while requiring only one additional flow-based loss term. Across a broad range of simulation benchmarks and real-robot experiments, our method consistently improves both performance and sample efficiency over the base policy, with particularly pronounced gains on real-robot tasks. These results highlight the importance of explicitly modeling the temporal structure inherent in action chunking.

1 INTRODUCTION

Action chunking has emerged as a default design choice in modern robot learning, spanning imitation learning policies (Zhao et al., 2023; Haldar et al., 2024; Chi et al., 2025) and generalist vision-language-action models (Black et al., 2024; Intelligence et al., 2025; Park et al., 2025a; Bjorck et al., 2025; Liu et al., 2026; Zheng et al., 2026). Rather than predicting one action at a time, action-chunked policies predict and execute an open-loop sequence of future actions conditioned on the current observation, providing denser supervision during training and promoting temporally coherent behavior at execution time. Recent approaches increasingly combine action chunking with expressive generative models, such as diffusion models and particularly flow matching (Ho et al., 2020; Lipman et al., 2023), to capture complex action distributions. This combination has become a remarkably effective and increasingly dominant recipe for challenging robotic manipulation tasks (Chi et al., 2025; Chisari et al., 2025; Black et al., 2024; Intelligence et al., 2025; Bjorck et al., 2025; Liu et al., 2026; Zheng et al., 2026). Yet, despite its widespread adoption, the underlying formulation of flow-based action-chunk generation has received surprisingly little scrutiny.

Almost all existing approaches represent action chunks merely as unstructured vectors during diffusion/flow-based generation, overlooking the physical-time dependencies among actions within each chunk. This leads to a key question: *Are we modeling action chunks in the right way that fully exploits their internal temporal structure?* Recent analyses have begun to explain the benefits of action chunking through its expressivity, compounding error reduction, control-theoretic stability, and implicit ensembling effects (Zhang et al., 2026; Lazzati et al., 2026; Liu et al., 2025). While these investigations help to explain why predicting sequences of actions can be beneficial, none of them touches upon the interplay between the denoising-based generation process and the internal temporal structure inside action chunks.

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

In fact, action chunks contain rich structure for policy learning, not only how action sequences are jointly distributed, but also how actions evolve over physical time. When treating action chunk $A_t = [a_t, a_{t+1}, \cdots, a_{t+H-1}]^T$ as an unstructured vector in flow matching, it leaves its internal physical-time organization implicit, effectively collapsing all generation variation onto a single flow-time axis. However, this is clearly problematic, as we naturally face two distinct temporal dimensions when dealing with flow-based action chunking : 1) *denoising time* $k \in [0, 1]$, which parameterized the transformation from noise ($k = 0$) to action chunk data ($k = 1$) (Lipman et al., 2023); and 2) *physical time* t, which indexes the evolution of the actions internally inside a chunk, typically bounded by external task/physical constraints rather than randomly varied. This actually requires the learned velocity field in flow matching to capture the coupled evolution in action chunk generation along both temporal dimensions.

To explicitly capture the physical-time dependencies within action chunks, we find that additionally supervising the policy with the physical-time derivative of chunked actions (i.e., $\dot{A}_t$) can be particularly effective. Specifically, we differentiate the flow-based policy along physical time through a Jacobian–vector product transformation, yielding an extra first-order flow-matching objective alongside the standard training loss. We further show that this seemingly simple scheme admits a principled interpretation as a special case of path-independent flow matching (PiFM) (Téllez et al., 2026), which models multi-parameter generative dynamics using velocity fields associated with each temporal axis while ensuring path-independent distributional transport. In our setting, action-chunk generation evolves jointly along denoising time and physical time dimensions. While conventional PiFM learns separate velocity fields for different temporal axes, we show that physical-time consistency of flow-based action chunking can be enforced by reusing the original denoising velocity field, requiring neither an additional prediction head nor architectural modifications.

Our method can be easily applied as a drop-in training objective to any flow-based action-chunked policy by adding only one first-order flow-matching loss, without modifying the policy architecture or inference procedure. Extensive experiments on LIBERO (Liu et al., 2023a), LIBERO-Plus (Fei et al., 2025), OGBench (Park et al., 2025b), and real-world robotic tasks demonstrate consistent improvements in both task performance and sample efficiency over the corresponding base policies, with particularly pronounced gains in real-robot tasks. Together, our theoretical and empirical results show that the standard flow-matching objective can be biased when applied to temporally structured action chunks, highlighting the importance of explicitly modeling their physical-time structure.

2 Related Work

Action chunking. Action chunking, popularized by ACT (Zhao et al., 2023), predicts a sequence of future actions from a single observation. It has since become a standard design in imitation learning and generative robot policies (Zhao et al., 2023; Chi et al., 2025; Black et al., 2024; Octo Model Team et al., 2024; Intelligence et al., 2025; Bjorck et al., 2025), and has also been extended to reinforcement learning (RL), where multi-step action prediction can facilitate value learning (Li et al., 2025; 2026). A large body of follow-up work studies how action chunks should be executed and represented. For example, the prediction and execution horizons of action chunks need not coincide: longer execution reduces replanning frequency but delays access to new observations (Liu et al., 2025; Black et al., 2026), motivating methods that dynamically adapt the execution horizon to the task or manipulation phase (Zhao et al., 2026; Nie et al., 2026). Related work in RL decouples policy and critic horizons to balance reactive control with multi-step value learning (Li et al., 2026). Other approaches redesign the chunked action space to promote generation smoothness, such as representing action sequences in the frequency domain with the discrete cosine transform (DCT) (Guo et al., 2026), or using B-spline control points (Yang et al., 2026). Streaming Flow Policy instead aligns flow generation with incremental action execution (Jiang et al., 2025).

Despite its empirical success, the mechanisms underlying action chunking remain incompletely understood. Existing analyses attribute its benefits to reduced compounding error, non-Markovian expressivity, control-theoretic stability, and implicit ensembling (Zhang et al., 2026; Lazzati et al., 2026). Meanwhile, recent studies of temporal consistency have focused primarily on coordinating successive action chunks through guided sampling or conditioning on previously generated actions (Liu et al., 2025; Malhotra et al., 2025; Park et al., 2025a; Black et al., 2025; Liu et al., 2026). These approaches either constrain relationships across chunks or regularize the action-space

representation. In contrast, how a conditional generative policy should model physical-time dependencies among actions within a single chunk—particularly their interaction with denoising-time dynamics—remains poorly understood.

Flow matching (FM). Our work focuses on flow matching for action-chunk generation, a framework increasingly adopted in recent robotic control and RL policies. Flow matching constructs a transport from a noise distribution $\mathcal{N}(0, I)$ to a target distribution $p(\cdot \mid s_t)$ through simulation-free regression of a time-dependent velocity field along a prescribed conditional probability path (Lipman et al., 2023; Tong et al., 2024). While effective at matching distributions, the standard FM objective does not explicitly constrain the learned velocity field to recover the intrinsic dynamics of the data or remain consistent across multiple directions of evolution. Recent work has therefore incorporated additional data structure into flow matching, including geometric and dynamical properties (Kapuśniak et al., 2024; Petrović et al., 2025), temporal dependencies (Zhang et al., 2024), and population-level context (Atanackovic et al., 2025). Path-independent flow matching (PiFM) (Téllez et al., 2026) generalizes FM to multi-parameter generative processes by learning compatible velocity fields whose induced transport is independent of the path taken across the parameter space. Its general formulation, however, requires a separate velocity field for each evolution axis and the construction of multi-parameter conditional probability paths, introducing additional architectural and path-design complexity. We identify flow-based action chunking as a structured instance of PiFM, with denoising time and within-chunk physical time forming two coupled evolution axes. Exploiting this structure allows us to enforce physical-time consistency using the original denoising velocity field, without introducing additional prediction heads or redesigning the underlying probability path.

3 What's Missing in Flow-Based Action Chunking?

In this section, we revisit the standard formulation of flow-based action chunking and identify its issue of missing physical-time dependency considerations. To explicitly enforce within-chunk physical-time consistency, we introduce an intuitive approach that additionally supervises the physical-time derivatives of actions, yielding an auxiliary first-order flow-matching objective. Despite its apparent simplicity, we later show that this objective is precisely a special case of PiFM.

3.1 Revisiting Flow Matching for Action Chunk Modeling

Conventional treatment. Let $\tau = \{(s_t, a_t)\}_{t=0}^{L-1}$ be a data trajectory consisting of states $s_t \in \mathbb{R}^{d_s}$ and actions $a_t \in \mathbb{R}^{d_a}$. Denote $A_{k,t} = [a_{k,t}, \dots, a_{k,t+H-1}]^T$ as the action chunk starting at physical time t at denoising time $k \in [0, 1]$. In particular, $A_{1,t} = [a_t, \dots, a_{t+H-1}]^T$ is the target action chunk of length H at $k = 1$. Conventional flow policies model action chunk by learning a denoising time k dependent velocity field $v_k : [0, 1] \times \mathbb{R}^{Hd_a} \to \mathbb{R}^{Hd_a}$, which induces a continuous flow $\phi_k : [0, 1] \times \mathbb{R}^{Hd_a} \to \mathbb{R}^{Hd_a}$ between a simple distribution to the target action chunk distribution. Following the standard linear interpolation path in conditional flow matching (Lipman et al., 2023; Liu et al., 2023b), we define the interpolation process for denoising time $k \in [0, 1]$ as

$$A_{k,t} = (1 - k)A_0 + kA_{1,t}, \tag{1}$$

where $A_0 \sim \mathcal{N}(0, I)$ is a noise sample and $A_{1,t}$ is the target action chunk. The velocity field $v_k(A_{k,t}, k, s_t)$ is learned by regressing it toward the target velocity $A_{1,t} - A_0$, i.e., minimizing

$$\mathbb{E}_{k\sim\mathcal{U}(0,1),\,(s_t,A_{1,t})\sim\mathcal{D},\,A_0\sim\mathcal{N}(0,I)} \|v_k(A_{k,t}, k, s_t) - (A_{1,t} - A_0)\|_2^2 . \tag{2}$$

The overlooked physical-time dimension. Crucially, stacking the actions into a single vector preserves their ordering but does not explicitly model their evolution along physical time. The conventional flow-matching objective constrains how $A_{k,t}$ evolves with the denoising time k, but imposes no constraint on how either the samples or their velocity field should evolve as the physical time t advances. A flow-based action chunk should therefore be viewed not as a one-dimensional generation process indexed only by k, but as a two-dimensional process indexed jointly by denoising time k and physical time t. Accordingly, we consider a two-parameter flow $\phi_{k,t} \in [0, 1]^2 \times \mathbb{R}^{Hd_a} \to \mathbb{R}^{Hd_a}$ and an associated velocity field $v_{k,t} \in [0, 1]^2 \times \mathbb{R}^{Hd_a} \to \mathbb{R}^{Hd_a}$ (Téllez et al., 2026). Under this formulation, the flow-matching objective is applied pointwise at each physical time:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{k\sim\mathcal{U}(0,1),\,(s_t,A_{1,t})\sim\mathcal{D},\,A_0\sim\mathcal{N}(0,I)} \|v_{k,t}(A_{k,t}, k, s_t) - (A_{1,t} - A_0)\|_2^2 . \tag{3}$$

3.2 Enforcing Physical-Time Consistency via First-Order Action Supervision

To incorporate physical-time dependencies within an action chunk, we examine how the predicted endpoint of the generative flow evolves along physical time. Given an intermediate sample $A_{k,t}$, we define the one-step endpoint prediction

$$\hat{A}_{1,t} := A_{k,t} + (1-k)\, v_{k,t}(A_{k,t}, k, s_t) \tag{4}$$

Eq. (4) is analogous to x_0-prediction trick in diffusion models (Karras et al., 2022; Dhariwal & Nichol, 2021), which recovers the target action chunk whenever $v_{k,t}$ equals the target velocity $A_{1,t} - A_0$. Differentiating $\hat{A}_{1,t}$ with respect to physical-time t yields a first-order endpoint prediction,

$$\begin{aligned}\dot{\hat{A}}_{1,t} &= \dot{A}_{k,t} + (1-k)\left[\frac{\partial v^i_{k,t}(A_{k,t},k,s_t)}{\partial s_t}\dot{s}_t + \sum_{j=0}^{H-1}\frac{\partial v^i_{k,t}(A_{k,t},k,s_t)}{\partial a_{k,t+j}}\dot{a}_{k,t+j}\right]_{i=0}^{H-1} \\ &= k\dot{A}_{1,t} + (1-k)\left[\frac{\partial v^i_{k,t}(A_{k,t},k,s_t)}{\partial s_t}\dot{s}_t + k\sum_{j=0}^{H-1}\frac{\partial v^i_{k,t}(A_{k,t},k,s_t)}{\partial a_{k,t+j}}\dot{a}_{t+j}\right]_{i=0}^{H-1}\end{aligned} \tag{5}$$

where $v^i_{k,t}$ denotes the i-th component of $v_{k,t}(A_{k,t}, k, s_t)$, and $i, j \in \{0, \dots, H-1\}$ index positions within the action chunk. We use $\dot{x} := dx/dt$ to denote differentiation with respect to physical time. The second equality follows by taking the physical-time derivative of Eq. (1). Because A_0 is random noise and independent of physical time, $\dot{A}_0 = 0$, yielding $\dot{A}_{k,t} = k\dot{A}_{1,t}$ and $\dot{a}_{k,t+j} = k\dot{a}_{t+j}$.

Note that $\dot{A}_{1,t} = [\dot{a}_t, \dots, \dot{a}_{t+H-1}]^T$ is the chunk formed by first-order actions, which can be empirically estimated from data trajectories, e.g., simply calculating $\dot{a}_t = a_{t+1} - a_t$ or adopting more advanced time-derivative estimation methods. This provides an additional first-order supervision signal for flow-based action chunking, which requires the first-order endpoint prediction $\dot{\hat{A}}_{1,t}$ to match its data-derived ground truth target $\dot{A}_{1,t}$:

$$\mathcal{L}_{1^{\text{st}}} = \mathbb{E}_{\substack{k\sim\mathcal{U}(0,1),\\ (s_t,A_{1,t})\sim\mathcal{D},\\ A_0\sim\mathcal{N}(0,I)}}\left[(1-k)^2\left\|\left[\frac{\partial v^i_{k,t}(A_{k,t},k,s_t)}{\partial s_t}\dot{s}_t + k\sum_{j=0}^{H-1}\frac{\partial v^i_{k,t}(A_{k,t},k,s_t)}{\partial a_{k,t+j}}\dot{a}_{t+j} - \dot{a}_{t+i}\right]_{i=0}^{H-1}\right\|_2^2\right]. \tag{6}$$

The role of $\mathcal{L}_{1^{\text{st}}}$ is not immediately obvious. Although derived from the physical-time derivative of a one-step endpoint prediction, it does not directly match the derivative of the final generated chunk, which is obtained by integrating the learned velocity field from $k = 0$ to 1. Computing that derivative exactly would require differentiating through the entire flow, as in adjoint matching (Domingo-Enrich et al., 2024), and incur substantial computational cost. In contrast, $\mathcal{L}_{1^{\text{st}}}$ is local and simulation-free. We next show that it actually admits an equivalent formulation of the path-independence equation in path-independent flow matching.

4 Reinterpreting Flow-Based Action Chunking as PiFM

The noised action chunk $A_{k,t}$ actually evolves along both denoising time k and physical time t. For these evolutions to form a consistent two-parameter flow, advancing along the two directions in either order should yield the same resulting distribution. This property is known as *path independence* (Téllez et al., 2026). To show how the previous first-order loss connects to the path independence consistency, we begin by introducing the general two-parameter PiFM formulation.

Two-parameter path-independent flow. Generally, consider a two-parameter generative process indexed by $k, t \in [0, 1]$, with continuously differentiable velocity fields $v_{k,t}, u_{k,t} : [0, 1]^2 \times \mathbb{R}^d \to \mathbb{R}^d$. Evolution along each direction is described by

$$\partial_k \phi_{k,t}(x) = v_{k,t}\big(\phi_{k,t}(x)\big), \qquad \partial_t \psi_{k,t}(x) = u_{k,t}\big(\psi_{k,t}(x)\big), \tag{7}$$

where we denote $\partial_k \phi_{k,t} := \partial_k \phi_{k,t}/\partial k$ and $\partial_t \psi_{k,t} := \partial_k \psi_{k,t}/\partial t$ for convenience. Here, $\phi_{k,t}(x)$ and $\psi_{k,t}(x)$ are flows along k and t dimension, respectively, with the other parameter fixed. We denote

the corresponding flow transport maps by $\Phi_{k,t\to k',t}$ and $\Psi_{k,t\to k,t'}$. Let $p_{k,t}$ denote the probability density path at (k,t). Path independence requires the two orders of transport to agree:

$$\left(\Phi_{k,t'\to k',t'} \circ \Psi_{k,t\to k,t'}\right)_{\#} p_{k,t} = \left(\Psi_{k',t\to k',t'} \circ \Phi_{k,t\to k',t}\right)_{\#} p_{k,t} \tag{8}$$

where $\#$ denotes the push-forward operator. Following PiFM (Téllez et al., 2026), obtaining a path-independent two-parameter flow requires two conditions. The first is to satisfy the property in Eq. (8). A necessary and sufficient condition for the existence of a unique smooth function $\phi_{k,t}$ under the path-independent transport is given as (Proposition 19.29 of Lee (2013)):

$$\partial_t v_{k,t}(x) + \nabla_x v_{k,t}(x) u_{k,t}(x) = \partial_k u_{k,t}(x) + \nabla_x v_{k,t}(x) u_{k,t}(x), \quad \forall x \in \mathbb{R}^d, \tag{9}$$

This gives the path-independence regularizer in Eq. (10)[1]. Moreover, velocity fields $v_{k,t}$ and $u_{k,t}$ should match their respective target velocities, which can be achieved through the conditional flow matching-style losses $\mathcal{L}_v$ and $\mathcal{L}_u$ similar to $\mathcal{L}_{FM}$ in Eq. (3). Combining these terms gives the overall PiFM objective $\mathcal{L}_{\text{PiFM}}$ in Eq.(11).

$$\mathcal{L}_{\text{PI}} = \mathbb{E}\Big[\big\|[\partial_t v_{k,t}(x) + \nabla_x v_{k,t}(x) u_{k,t}(x)] - [\partial_k u_{k,t}(x) + \nabla_x u_{k,t}(x) v_{k,t}(x)]\big\|_2^2\Big], \tag{10}$$

$$\mathcal{L}_{\text{PiFM}} = \mathcal{L}_v + \mathcal{L}_u + \mathcal{L}_{\text{PI}}. \tag{11}$$

General PiFM separately learns two velocity fields for different temporal directions. In flow-based action chunking, k is the denoising time, and we have $\mathcal{L}_v = \mathcal{L}_{\text{FM}}$. The remaining mystery lies in the physical-time objective $\mathcal{L}_u$ and the compatibility objective $\mathcal{L}_{\text{PI}}$. Since actions within a chunk are temporally ordered, their temporal transition provides an external data-derived signal that may allow both objectives to be addressed through the denoising velocity field $v_{k,t}$.

4.1 Enforcing Path Independence via First-Order Supervision

To unravel the remaining mystery, we first examine $\mathcal{L}_{\text{PI}}$ in Eq. (10). In our flow-based action chunking setting, $v_{k,t}(A_{k,t}, k, s_t)$ is the denoising-time conditional velocity, while $u_{k,t}$ denotes the conditional velocity that induces the target marginal transport of $A_{k,t}$ along physical time t. We can show that the target physical-time velocity can be identified as $u_{k,t} = k\dot{A}_{1,t}$ following the linear interpolation path. This allows us to substitute it into $\mathcal{L}_{\text{PI}}$ without learning a second velocity field. The following theorem establishes this identification. The proof is provided in Appendix C.1.

Theorem 1. *Fix $k \in [0,1)$ and consider the conditional physical-time velocity field $u_{k,t} = k\dot{A}_{1,t}$. If its induced marginal field is continuous in t and uniformly Lipschitz in its action input, then it generates the marginal evolution of $A_{k,t}$ along physical time. In particular, the resulting marginal flow satisfies $(\Psi_{k,t\to k,t'})_{\#} p_{k,t} = p_{k,t'}$, where $p_{k,t}$ is the probability density path of $A_{k,t}$.*

Considering the terms in $\mathcal{L}_{\text{PI}}$, with $u_{k,t} = k\dot{A}_{1,t}$ identified, differentiating the physical-time velocity $u_{k,t}$ with respect to k is straightforward, yielding $\partial_k u_{k,t} = \dot{A}_{1,t}$. Moreover, as the target velocity of $u_{k,t} = k\dot{A}_{1,t}$ is independent of noised intermediate sample $A_{k,t}$, hence $\nabla_{A_{k,t}} u_{k,t} = 0$. On the other hand, differentiating the denoising-time velocity $v_{k,t}$ with respect to t must account for the temporal variation of both the conditioning state s_t and all action components in $A_{k,t}$. Substituting the above terms into Eq. (10) and leverage the relationship $\dot{a}_{k,t+j} = k\dot{a}_{t+j}$ that we have derived in Section 3.2, we can obtain the following form of $\mathcal{L}_{\text{PI}}$ under the action chunking setting:

$$\mathcal{L}_{\text{PI}} = \mathbb{E}\left\| \left[\frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial s_t} \dot{s}_t + k \sum_{j=0}^{H-1} \frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial a_{k,t+j}} \dot{a}_{t+j} - \dot{a}_{t+i} \right]_{i=0}^{H-1} \right\|_2^2. \tag{12}$$

The detailed derivation is provided in Appendix B.1. A keen reader will immediately notice that the above objective has exactly the same form as the bracketed term in our previously derived first-order loss $\mathcal{L}_{1^{\text{st}}}$ in Eq. (6). Therefore, $\mathcal{L}_{1^{\text{st}}}$ introduced in Section 3.2 is precisely a weighted squared regularization on the path independence property, revealing that the seemingly straightforward first-order supervision enforces path independence between denoising generation and physical-time evolution. Unlike Téllez et al. (2026), which learns two independently parameterized velocity fields, we prescribe the physical-time direction with the inherent temporal structure in action chunks and the linear conditional path construction, and therefore require no additional parameterized velocity field.

[1]Note that the form of $\mathcal{L}_{\text{PI}}$ in the PiFM paper (Téllez et al., 2026) is flawed, which misses a minus sign. We provide the correct version by following its original form in Lee (2013). See Appendix B.1 for more discussion.

4.2 Unified Generative- and Physical-Time Flow Matching in Action Chunking

We then turn to the last remaining objective, $\mathcal{L}_u$. Specifically, we consider the linear approximation of the physical-time derivative. Recall from Theorem 1 that the target physical-time conditional velocity is $u_{k,t} = k\dot{A}_{1,t}$. Since $A_{1,t} \in \mathbb{R}^{H\times d_a}$, each action dimension forms a discrete-time trajectory. We reconstruct each action dimension as a continuous trajectory using a differentiable basis expansion, such as B-splines or a discrete cosine basis (de Boor, 1978; Ahmed et al., 1974) (implementation details are provided in Section 5). This expansion provides a linear operator $G \in \mathbb{R}^{H\times H}$ for approximating its time derivative, as shown in the left part of Eq. (13). Meanwhile, the endpoint identity also exhibits a linear structure (see Eq. (4)), as given in the same equation below:

$$\dot{A}_{1,t} \approx GA_{1,t}, \qquad A_{1,t} = A_{k,t} + (1-k)v_{k,t}. \tag{13}$$

Combining these two linear structures allows $u_{k,t}$ to be approximated through $v_{k,t}$ as,

$$u_{k,t} = k\dot{A}_{1,t} \approx kGA_{1,t} = kG\big[A_{k,t} + (1-k)v_{k,t}\big], \tag{14}$$

Also, under the linear interpolation path, recall that the target velocity $v_{k,t} = A_{1,t} - A_0$, the right-hand-side term hence has a target value $kG\big[A_{k,t} + (1-k)(A_{1,t} - A_0)\big)\big]$. This provides a tractable supervision signal for $u_{k,t}$ through $v_{k,t}$ by matching its form in Eq. (14) and the previous target, which results in the following objective for $\mathcal{L}_u$:

$$\mathcal{L}_u = \mathbb{E}_{k\sim\mathcal{U}(0,1),(s_t,A_{1,t})\sim\mathcal{D},A_0\sim\mathcal{N}(0,I)}\left[\Big\|k(1-k)G\Big(v_{k,t}(A_{k,t},k,s_t) - (A_{1,t}-A_0)\Big)\Big\|_2^2\right]. \tag{15}$$

Recall that $\mathcal{L}_v$ is the denoising-time velocity-learning objective, which is exactly the standard flow-matching objective $\mathcal{L}_{\mathrm{FM}}$. Moreover, the following theorem indicates that the joint optimization of $\mathcal{L}_u + \mathcal{L}_v$ is equivalent, in terms of population optima, to optimizing $\mathcal{L}_v$ (i.e., $\mathcal{L}_{\mathrm{FM}}$) alone.

Theorem 2. *Under the linear transformations in Eq. (13) and some mild assumption on G, the physical-time velocity learning objective $\mathcal{L}_u$ can be written as a variant form of the flow-matching objective of $\mathcal{L}_v$ under a different Bregman divergence, thereby corresonding to the same optimal marginal velocity solution. Moreover, any population minimizer of $\mathcal{L}_v$ also minimizes $\mathcal{L}_u$.*

The proof is provided in Appendix C.2. Together, the results in Sections 4.1 and 4.2 give a unified view of the two temporal directions. At the population optimum, the velocity-matching losses $\mathcal{L}_v + \mathcal{L}_u$ are captured by the zeroth-order objective $\mathcal{L}_{\mathrm{FM}}$. The path-independence loss $\mathcal{L}_{\mathrm{PI}}$ corresponds to the first-order supervision $\mathcal{L}_{1^{\mathrm{st}}}$. With an additional hyperparameter λ, the resulting overall training objective for path-independent flow under action chunking is therefore given as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{FM}} + \lambda\mathcal{L}_{1^{\mathrm{st}}}. \tag{16}$$

5 Implementation Details

Improved physical-time derivative estimation. Estimating the time derivative of an action sequence is challenging because only discrete-time action labels are available in the dataset. We adopt the natural prior that action chunks are dominated by low-frequency components under typical robot dynamics. A basis representation can reconstruct a high-fidelity continuous action trajectory from these discrete labels, yielding analytical derivatives that better preserve the underlying temporal structure than forward differences, i.e., $\dot{a}_t \approx a_{t+1} - a_t$. Specifically, we fit a degree-p B-spline $\hat{a}_t(x) = \sum_{m=0}^{M-1} B_m(x)c_{t,m}$ to $A_{1,t} \in \mathbb{R}^{H\times d_a}$, where $x_h = h\Delta$ denotes the physical time of the h-th action where Δ denotes the sampling time. Let $\mathbf{B} \in \mathbb{R}^{H\times M}$ denote the B-spline basis matrix evaluated at these action times, with $\mathbf{B}_{h,m} = B_m(x_h)$, and let $\dot{\mathbf{B}}_{h,m} = \partial_x B_m(x_h)$ denote the corresponding derivative matrix. The fitted control points and action derivatives are

$$C_t^* = \mathbf{B}^\dagger A_{1,t}, \qquad \dot{A}_{1,t} \approx \dot{\mathbf{B}}C_t^* = \underbrace{\dot{\mathbf{B}}\mathbf{B}^\dagger}_{G} A_{1,t}. \tag{17}$$

Here, $\mathbf{B}^\dagger$ denotes the Moore–Penrose pseudoinverse, and G is a fixed differentiation matrix that is precomputed once and applied independently to each action dimension. We provide more detail discussion on Appendix D.1.

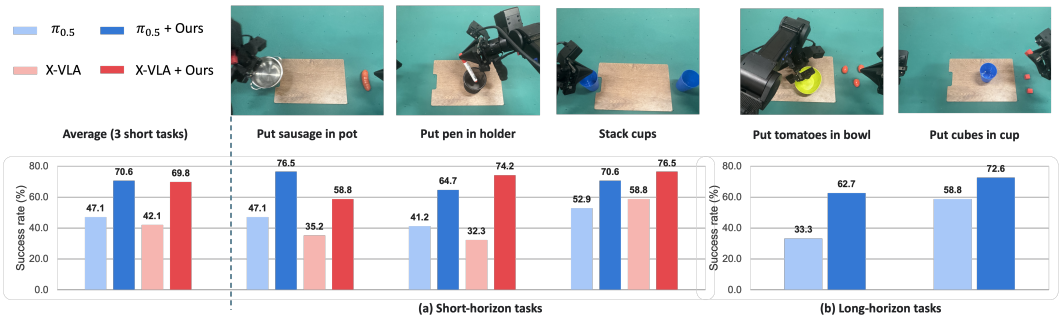

Figure 1: Real-robot evaluation on the dual-arm ARX platform. (a) Success rates on three short-horizon tasks for $\pi_{0.5}$ and X-VLA, with their averages shown on the left. (b) Success rates on two additonal long-horizon tasks for $\pi_{0.5}$. Task images appear above the corresponding results.

Unlike action trajectories, states are high-dimensional and heterogeneous, may be noisy, and lack a shared low-frequency prior for spline fitting. We therefore estimate the state tangent by central difference, $\dot{s}_t \approx (s_{t+1} - s_{t-1})/(2\Delta)$. The action and state tangents $(k\dot{A}_{1,t}, \dot{s}_t)$ are then evaluated through a single Jacobian–vector product for Eq.(6), without forming the full Jacobian. Physical-time t dependence enters through $A_{k,t}$ and s_t, so the auxiliary loss needs neither an explicit t input nor a prediction head. It is applied only during training, leaving inference unchanged.

Extension to offline reinforcement learning. Our method extends to Q-Chunking (Li et al., 2025) and its IFQL variant, QC-IFQL. The first-order loss in Eq. (6) augments the flow-matching objective, while value-learning and Q-guided policy extraction remain unchanged. The IQL-style Q/value updates and Q-based selection among flow-generated candidate chunks are retained. The specific implementation details are presented in Appendix D.5.

6 Experiment

To further assess the effectiveness of first-order action supervision, we conduct a comprehensive empirical evaluation. We first evaluate lightweight flow-based policies and large-scale VLA models on LIBERO (Liu et al., 2023a), LIBERO-Plus (Fei et al., 2025), and real-world dual-arm manipulation tasks to examine whether our method brings benefits across model scales and deployment conditions. We then incorporate the same objective into Q-Chunking (Li et al., 2025) and its variant QC-IFQL on OGBench (Park et al., 2025b) to investigate whether its benefits extend beyond imitation learning to offline reinforcement learning. Finally, we ablate the derivative estimator and loss weight to analyze the key design choices of our method.

6.1 Imitation Learning Setting

Simulation benchmarks and evaluation. We evaluate 40 LIBERO tasks (Liu et al., 2023a), 10 each from Spatial, Object, Goal, and long-horizon LIBERO-10. We assess policy robustness on the more challenging LIBERO-Plus (Fei et al., 2025) benchmark. We compare lightweight policies and pretrained X-VLA on 4,293 lighting, sensor-noise, and robot-initialization cases with fixed object layouts and task goals. We reserve background, camera, object, and language perturbations for X-VLA because they also test visual or semantic generalization beyond lightweight policies' fixed-scene training. Each variant is evaluated once per case across three seeds (12,879 rollouts); we report mean success. Appendix E reports X-VLA results for all seven categories.

Policy backbones. In simulation, we use LeRobot (Cadene et al., 2024) implementations of Multi-task DiT (TRI LBM Team et al., 2025), Diffusion Policy (Chi et al., 2025), and X-VLA (Zheng et al., 2026). We train the Diffusion Policy architecture with flow matching in place of its diffusion objective. On the real robot, we evaluate $\pi_{0.5}$ and X-VLA. For each policy, +Ours adds the first-order objective during training while retaining the baseline architecture and inference procedure. We present the implementation details of the backbones in Appendix D.3.

Table 1: **Evaluation results on LIBERO and LIBERO-Plus.** Suite-wise success rates (%) on the four LIBERO task suites under the clean setting and three LIBERO-Plus perturbations. Each cell shows Base → +Ours, with results averaged over three random seeds. Scores that improve over their corresponding baselines are bolded, and red ↑ labels give gains in score points.

Policy	Setting	Spatial	Object	Goal	LIBERO-10
Multi-task DiT	Clean	90.50 → **92.10** ↑1.60	97.20 → 97.20	94.10 → **94.20** ↑0.10	81.53 → **84.20** ↑2.67
	Lighting	75.51 → **82.88** ↑7.37	83.16 → **84.34** ↑1.18	54.66 → **58.60** ↑3.94	55.23 → **69.10** ↑13.87
	Sensor	52.99 → **63.53** ↑10.54	47.87 → **54.03** ↑6.16	45.65 → **45.78** ↑0.13	34.45 → **43.73** ↑9.28
	Robot	45.14 → 43.14	32.16 → **34.80** ↑2.64	37.41 → **40.22** ↑2.81	42.83 → **47.41** ↑4.58
Diffusion Policy	Clean	83.70 → 80.60	96.70 → 96.50	10.10 → **10.13** ↑0.03	69.80 → **71.53** ↑1.73
	Lighting	83.90 → 79.62	91.25 → **93.27** ↑2.02	16.67 → 14.93	71.90 → **77.14** ↑5.24
	Sensor	55.13 → 51.09	42.54 → **47.51** ↑4.97	12.14 → 11.52	26.58 → **28.86** ↑2.28
	Robot	45.71 → **50.00** ↑4.29	32.29 → **46.86** ↑14.57	7.46 → **8.23** ↑0.77	40.38 → **47.34** ↑6.96
X-VLA	Clean	92.10 → 91.40	98.90 → 98.50	94.70 → **96.40** ↑1.70	83.70 → **90.30** ↑6.60
	Lighting	92.30 → 89.38	99.67 → 99.16	80.11 → **91.76** ↑11.65	86.68 → **93.43** ↑6.75
	Sensor	62.40 → 59.97	64.81 → 60.79	50.27 → 47.76	53.79 → **55.79** ↑2.00
	Robot	86.00 → **87.00** ↑1.00	94.60 → **97.36** ↑2.76	81.05 → **86.80** ↑5.75	73.79 → **83.84** ↑10.05

Simulation results. The clearest result in Table 1 is on LIBERO-10, where +Ours improves all three policies under every evaluated condition. These long-horizon tasks require multiple dependent manipulation stages, so inconsistencies between generative-time flow and physical-time action evolution would disrupt subsequent actions. First-order supervision promotes compatibility between these two temporal directions. Its consistent gains on LIBERO-10 align with the prediction that this property becomes more consequential as tasks demand sustained, coherent action execution.

Real-robot evaluation. On dual-arm ARX platform, we evaluate $\pi_{0.5}$ and X-VLA on three short-horizon tasks (*Put Sausage in Pot*, *Put Pen in Holder*, and *Stack Cups*); $\pi_{0.5}$ is additionally evaluated on two long-horizon tasks (*Put Tomatoes in Bowl* and *Put Cubes in Cup*). First-order supervision improves all evaluated policy–task pairs (Fig. 1), extending simulation gains to real-world bimanual manipulation. One explanation for these gains is that +Ours reduces within-chunk action inconsistency as shown in Figure 2, the mean within-chunk action standard deviation and mean adjacent-action absolute difference decrease by 37.6% and 35.5%, respectively. We present the implementation details in Appendix D.4.

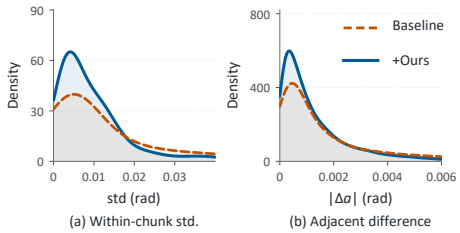

Figure 2: X-VLA joint-action variation on *Put Pen in Holder* task. (a) We compute each joint's standard deviation within a chunk. (b) The absolute difference $|a_{t+1} - a_t|$ between consecutive actions of each joint.

6.2 Reinforcement Learning Setting

We add first-order supervision to the flow generators of QC (Li et al., 2025) and QC-IFQL on OGBench (Park et al., 2025b), leaving value learning and Q-guided chunk selection unchanged. Table 2 reports the complete task-level results, with all scores rounded to the nearest integer. The QC and QC-IFQL baseline scores are taken directly from the offline 1M-step results reported in Q-Chunking (Li et al., 2025), rather than being rerun in our implementation. Adding first-order supervision improves both methods across all three task groups. The overall average score increases from 52 to 58 for QC and from 32 to 37 for QC-IFQL, corresponding to gains of 6 and 5 points, respectively. The largest group-level improvements occur on cube-double. Gains are also observed on scene-sparse and cube-triple, although performance on the latter remains close to the task floor. These consistent improvements suggest that first-order supervision complements the value-learning and Q-guided policy-extraction mechanisms of offline RL by improving the generated action candidates without modifying the underlying RL procedure.

Table 2: **Evaluation scores on OGBench tasks in the offline setting.** Scores that improve over the corresponding base policy are bolded, and red ↑ labels give gains in score points. The QC and QC-IFQL baseline scores are taken from the offline results reported by Q-Chunking (Li et al., 2025).

	scene-sparse					cube-double					cube-triple					Overall
Method	task1	task2	task3	task4	task5	task1	task2	task3	task4	task5	task1	task2	task3	task4	task5	Avg.
QC	100	98	90	93	43	84	79	68	22	74	13	1	7	0	0	52
QC + Ours	100	**100** ↑2	**100** ↑10	**98** ↑5	**53** ↑10	**92** ↑8	**87** ↑8	**84** ↑16	**35** ↑13	**85** ↑11	**28** ↑15	**2** ↑1	**10** ↑3	0	0	**58** ↑6
QC-IFQL	93	68	65	93	93	17	15	8	5	17	0	0	0	0	0	32
QC-IFQL + Ours	**99** ↑6	**73** ↑5	**74** ↑9	**98** ↑5	**100** ↑7	**33** ↑16	**22** ↑7	**16** ↑8	**6** ↑1	**26** ↑9	**5** ↑5	0	0	0	0	**37** ↑5

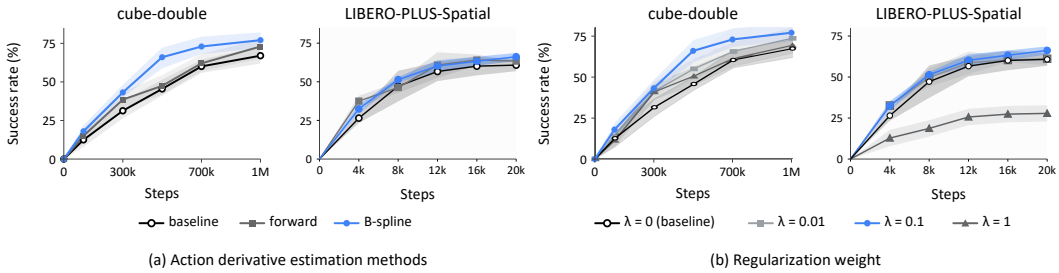

Figure 3: **Ablation studies. (a) Left:** Action derivative estimation, with $\lambda = 0.1$ for the first-order variants. **(b) Right:** First-order regularization weight λ, using B-spline derivatives for nonzero weights. Within each group, the left plot shows OGBench cube-double and the right plot shows LIBERO-PLUS-Spatial; $\lambda = 0$ denotes the base policy.

6.3 Ablation Experiments

Action derivative estimation. At fixed $\lambda = 0.1$, the estimator changes $\dot{A}_{1,t}$ both as the JVP direction and as the target of the first-order loss. Forward differences already outperform the baseline, while B-splines develop a much larger mid-training lead on cube-double. Thus, the first-order signal is useful without splines, but constructing a continuous tangent can improve learning efficiency.

First-order regularization weight. With B-spline derivatives fixed, $\lambda = 0.1$ reaches the cube-double baseline's final success in roughly half the training steps. At $\lambda = 1$, cube-double remains near its baseline, whereas LIBERO-PLUS-Spatial plateaus around 28%, versus about 60% for its baseline. This contrast may reflect differences in model capacity and training-data scale, which affect how well a policy can fit the flow-matching and tangent constraints jointly.

7 Conclusion

In this work, we revisit the prevailing treatment of action chunks as unstructured vectors during flow-based generation, which overlooks the physical-time dependencies among actions within each chunk. We show that additionally supervising the first-order temporal derivatives of actions within a chunk provides a simple and effective way to incorporate these dependencies into flow-based policies. We further establish that this supervision corresponds to a special case of path-independent flow matching, in which action-chunk generation evolves jointly along denoising time and physical time. The resulting method requires one additional training loss, without modifying the policy architecture or inference procedure.

By incorporating first-order supervision into lightweight policies and large-scale VLA models, we observe performance gains on LIBERO and LIBERO-Plus, as well as improved performance and sample efficiency in real-robot manipulation. Beyond imitation learning, applying the same supervision to Q-Chunking and QC-IFQL also improves their performance on OGBench, further demonstrating the applicability of our method. Overall, these results highlight the importance of explicitly modeling the temporal structure inherent in action chunking and establish first-order supervision as a simple and broadly applicable way to exploit it.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing, grammar checking, and improving the readability of the manuscript. We did not use generative AI tools for any research task requiring disclosure; such tasks were either performed entirely by the authors or were not applicable to this work. In particular, generative AI tools were not used to develop the theoretical or conceptual framework, formulate mathematical claims or proofs, propose hypotheses, design the methodology or experiments, implement the method, generate or process data, conduct analyses, or interpret the results. All AI-assisted language edits were carefully reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Nasir Ahmed, T. Natarajan, and Kamisetty R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

Lazar Atanackovic, Xi Zhang, Brandon Amos, Mathieu Blanchette, Leo J. Lee, Yoshua Bengio, Alexander Tong, and Kirill Neklyudov. Meta flow matching: Integrating vector fields on the Wasserstein manifold. In *International Conference on Learning Representations*, 2025.

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

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

Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-art machine learning for real-world robotics in PyTorch, 2024.

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

Eugenio Chisari, Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. In *Conference on Robot Learning*, 2025.

Carl de Boor. *A Practical Guide to Splines*, volume 27. Springer, 1978.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *Advances in Neural Information Processing Systems*, 2024.

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

Jianing Guo, Fangzheng Chen, Zihao Mao, Lik Hang Kenny Wong, Zhenhong Wu, Yu Li, Yishuai Cai, Yuanpei Chen, Yikun Ban, Kai Chen, Qi Dou, Yaodong Yang, Xianglong Liu, Huijie Zhao, and Simin Li. Frequency-aware flow matching for continuous and consistent robotic action generation. *arXiv preprint arXiv:2606.20135*, 2026.

Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. BAKU: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. In *Conference on Robot Learning*, 2025.

Kacper Kapuśniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael Bronstein, Avishek Joey Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why does action chunking improve behavioral cloning performance in robotic control? *arXiv preprint arXiv:2608.02547*, 2026.

John M Lee. *Introduction to smooth manifolds*. Springer, 2013.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled q-chunking. In *International Conference on Learning Representations*, volume 2026, pp. 145315–145389, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

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

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023b.

Yuejiang Liu, Jubayer Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *International Conference on Learning Representations*, volume 2025, pp. 4594–4627, 2025.

Yufeng Liu, Hang Yu, Juntu Zhao, Bocheng Li, Di Zhang, Mingzhu Li, Wenxuan Wu, Yingdong Hu, Junyuan Xie, Junliang Guo, Dequan Wang, and Yang Gao. Learning native continuation for action chunking flow policies. *arXiv preprint arXiv:2602.12978*, 2026.

Rhea Malhotra, Yuejiang Liu, and Chelsea Finn. Self-guided action diffusion. *arXiv preprint arXiv:2508.12189*, 2025. URL https://arxiv.org/abs/2508.12189.

Junnan Nie, Jiayi Li, Jiachen Zhang, Junyi Lao, Chenghao Liu, Tianle Zhang, Liang Lin, and Songfang Huang. Pace: Phase-aware chunk execution for robot policies with action chunking. *arXiv preprint arXiv:2606.00537*, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. URL https://octo-models.github.io/.

Minho Park, Kinam Kim, Junha Hyung, Hyojin Jang, Hoiyeong Jin, Jooyeol Yun, Hojoon Lee, and Jaegul Choo. ACG: Action coherence guidance for flow-based VLA models. *arXiv preprint arXiv:2510.22201*, 2025a.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025b.

Katarina Petrović, Lazar Atanackovic, Viggo Moro, Kacper Kapuśniak, İsmail İlkan Ceylan, Michael Bronstein, Avishek Joey Bose, and Alexander Tong. Curly flow matching for learning non-gradient field dynamics. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8748–8763, 2021.

Francisco Téllez, AmirHossein Zamani, Philippe Martin, Shuang Ni, Guy Wolf, Eugene Belilovsky, Sina Sanjari, and Yanlei Zhang. Path-independent flow matching for multi-parameter generative dynamics. *arXiv preprint arXiv:2605.13487*, 2026.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

TRI LBM Team, Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025. URL https://arxiv.org/abs/2507.05331.

Fan Yang, Peiguang Jing, Kaihua Qu, Ningyuan Zhao, and Yuting Su. ABPolicy: Asynchronous B-spline flow policy for real-time and smooth robotic manipulation. *arXiv preprint arXiv:2602.23901*, 2026.

Thomas T Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and data augmentation yield exponential improvements in behavior cloning for continuous spaces. In *International Conference on Learning Representations*, volume 2026, pp. 126689–126742, 2026.

Xi Zhang, Yuan Pu, Yuki Kawamura, Andrew Loza, Yoshua Bengio, Dennis L. Shung, and Alexander Tong. Trajectory flow matching with applications to clinical time series modeling. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Yuchi Zhao, Miroslav Bogdanovic, Arjun Sohal, Liyu Tao, Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. Dynamic execution horizon prediction for chunk-based robot policies. *arXiv preprint arXiv:2606.11408*, 2026.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026.

A LIMITAION

Although our first-order action supervision improves performance in both simulation and real-robot experiments, computing JVPs and backpropagating through them can incur substantial computation and memory costs for large networks. In particular, optimizing the auxiliary loss involves second-order mixed derivatives with respect to network parameters and inputs, which may increase sensitivity to numerical errors and noisy derivative targets, potentially requiring additional gradient regularization for stable training. Approximating JVPs with finite differences offers a potential alternative that avoids explicit mixed-derivative computation, at the cost of approximation error and sensitivity to the finite-difference step size.

Furthermore, our formulation does not directly apply to discrete state variables, for which the physical-time derivative $\dot{s}_t$ is not defined in the usual sense. Such variables require a suitable continuous representation with meaningful temporal derivatives, or must be excluded from the differentiated state components while remaining available as conditioning inputs to the policy.

B DERIVATIONS DETAILED

B.1 PATH INDEPENDENCE AND FIRST ORDER ENDPOINT PREDICTION

In this section, We show that, under the action-chunking setting, the general path-independence objective

$$\mathcal{L}_{\mathrm{PI}} = \mathbb{E}\Big[\Big\| \big[\partial_t v_{k,t}(x) + \nabla_x v_{k,t}(x) u_{k,t}(x)\big] - \big[\partial_k u_{k,t}(x) + \nabla_x u_{k,t}(x) v_{k,t}(x)\big] \Big\|_2^2\Big]$$

is equivalent to

$$\mathcal{L}_{\mathrm{PI}} = \mathbb{E}\left[\left\| \left[\frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial s_t} \dot{s}_t + k \sum_{j=0}^{H-1} \frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial a_{k,t+j}} \dot{a}_{t+j} - \dot{a}_{t+i} \right]_{i=0}^{H-1} \right\|_2^2 \right].$$

In the action-chunking setting, let

$$x = A_{k,t},$$

and parameterize the generative-time velocity field as

$$v_{k,t}(x) = v_\theta(A_{k,t}, k, s_t).$$

Here, x denotes only the action-chunk variable, while s_t is the physical-time-dependent conditioning state.

Since $v_\theta(A_{k,t}, k, s_t)$ is a physical-time-ignorant network, we separately account for the effects of the action chunk $A_{k,t}$ and the conditioning state s_t on its physical-time variation. Accordingly, when evaluating $\partial_t v_{k,t}(x)$, we hold $x = A_{k,t}$ fixed, such that the remaining dependence on t arises only through s_t. Hence, for the i-th component,

$$\partial_t v^i_{k,t}(x) = \frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial s_t} \dot{s}_t.$$

For the second term, since $x = A_{k,t}$,

$$\begin{aligned} \big[\nabla_x v_{k,t}(x) u_{k,t}\big]^i &= \sum_{j=0}^{H-1} \frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial a_{k,t+j}} \, u^j_{k,t} \\ &= k \sum_{j=0}^{H-1} \frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial a_{k,t+j}} \dot{a}_{t+j}. \end{aligned}$$

Equivalently, this follows from the conditional path

$$A_{k,t} = (1-k)A_0 + kA_{1,t},$$

for which A_0 remains fixed with respect to physical time, yielding

$$\dot{A}_{k,t} = k\dot{A}_{1,t}.$$

For the third term,

$$\partial_k u_{k,t} = \partial_k\Big(k\dot{A}_{1,t}\Big) = \dot{A}_{1,t},$$

and hence

$$\big[\partial_k u_{k,t}\big]^i = \dot{a}_{t+i}.$$

Finally, $\dot{A}_{1,t}$ is specified by the trajectory distribution in the dataset and is independent of the interpolated action-chunk variable $x = A_{k,t}$. Therefore,

$$\nabla_x u_{k,t} = \nabla_x\Big(k\dot{A}_{1,t}\Big) = 0,$$

and thus

$$\nabla_x u_{k,t}\, v_{k,t}(x) = 0.$$

Combining the four terms gives

$$\begin{aligned}&\big[\partial_t v_{k,t} + \nabla_x v_{k,t} u_{k,t} - \partial_k u_{k,t} - \nabla_x u_{k,t} v_{k,t}\big]^i \\ &\quad= \frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial s_t}\dot{s}_t + k\sum_{j=0}^{H-1}\frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial a_{k,t+j}}\dot{a}_{t+j} - \dot{a}_{t+i}.\end{aligned}$$

Hence,

$$\boxed{\mathcal{L}_{\mathrm{PI}} = \mathbb{E}\left[\left\|\left[\frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial s_t}\dot{s}_t + k\sum_{j=0}^{H-1}\frac{\partial v^i_{k,t}(A_{k,t}, k, s_t)}{\partial a_{k,t+j}}\dot{a}_{t+j} - \dot{a}_{t+i}\right]_{i=0}^{H-1}\right\|_2^2\right].}$$

B.2 First order endpoint prediction

Fix k and A_0 when differentiating along physical time. Since $A_{k,t} = (1-k)A_0 + kA_{1,t}$, we have $\dot{a}_{k,t+j} = k\dot{a}_{t+j}$. The learned field has no explicit t input; its time dependence comes through $A_{k,t}$ and s_t. Thus, for each output position i,

$$\begin{aligned}\partial_t v^i_{k,t}(A_{k,t}, k, s_t) &= \frac{\partial v^i_{k,t}}{\partial s_t}\dot{s}_t + \sum_{j=0}^{H-1}\frac{\partial v^i_{k,t}}{\partial a_{k,t+j}}\dot{a}_{k,t+j} \\ &= \frac{\partial v^i_{k,t}}{\partial s_t}\dot{s}_t + k\sum_{j=0}^{H-1}\frac{\partial v^i_{k,t}}{\partial a_{k,t+j}}\dot{a}_{t+j}. \qquad (18)\end{aligned}$$

The partial derivatives on the right are evaluated at $(A_{k,t}, k, s_t)$. Substituting this into $\hat{A}_{1,t} = A_{k,t} + (1-k)v_{k,t}(A_{k,t}, k, s_t)$ gives

$$\hat{\dot{a}}_{t+i} - \dot{a}_{t+i} = (1-k)\left[\partial_t v^i_{k,t}(A_{k,t}, k, s_t) - \dot{a}_{t+i}\right]. \qquad (19)$$

Therefore,

$$\mathcal{L}_{1^{\mathrm{st}}} = \mathbb{E}\left[(1-k)^2\sum_{i=0}^{H-1}\left\|\partial_t v^i_{k,t}(A_{k,t}, k, s_t) - \dot{a}_{t+i}\right\|_2^2\right]. \qquad (20)$$

B.3 Unified two-parameter Flow Matching in Action Chunking

We now expand the physical-time velocity objective using the linear operator G. The conditional physical-time velocity satisfies

$$u_{k,t} = k\dot{A}_{1,t} \approx kGA_{1,t}. \tag{21}$$

To express this target through the denoising-time velocity, start from the interpolation $A_{k,t} = (1-k)A_0 + kA_{1,t}$. Direct substitution gives

$$\begin{aligned} A_{k,t} + (1-k)(A_{1,t} - A_0) &= (1-k)A_0 + kA_{1,t} + (1-k)A_{1,t} - (1-k)A_0 \\ &= A_{1,t}. \end{aligned} \tag{22}$$

Applying kG to both sides and distributing the linear operator yields

$$\begin{aligned} kGA_{1,t} &= kG\big[A_{k,t} + (1-k)(A_{1,t} - A_0)\big] \\ &= kGA_{k,t} + k(1-k)G(A_{1,t} - A_0). \end{aligned} \tag{23}$$

Thus the operator-based physical-time target has one term determined by $A_{k,t}$ and one term determined by the conditional denoising-time velocity $A_{1,t} - A_0$.

Apply the same transformation to the one-step endpoint prediction $\hat{A}_{1,t} = A_{k,t} + (1-k)v_{k,t}(A_{k,t}, k, s_t)$. Its operator-based physical-time prediction is

$$\begin{aligned} kG\hat{A}_{1,t} &= kG\big[A_{k,t} + (1-k)v_{k,t}(A_{k,t}, k, s_t)\big] \\ &= kGA_{k,t} + k(1-k)Gv_{k,t}(A_{k,t}, k, s_t). \end{aligned} \tag{24}$$

Subtracting Eq. (23) from Eq. (24) shows each cancellation:

$$\begin{aligned} kG\hat{A}_{1,t} - kGA_{1,t} &= \big[kGA_{k,t} + k(1-k)Gv_{k,t}(A_{k,t}, k, s_t)\big] \\ &\quad - \big[kGA_{k,t} + k(1-k)G(A_{1,t} - A_0)\big] \\ &= k(1-k)G\big[v_{k,t}(A_{k,t}, k, s_t) - (A_{1,t} - A_0)\big]. \end{aligned} \tag{25}$$

Squaring this physical-time residual and taking the expectation gives

$$\begin{aligned} \mathcal{L}_u &= \mathbb{E}\left[\left\| kG\hat{A}_{1,t} - kGA_{1,t} \right\|_2^2\right] \\ &= \mathbb{E}\left[\left\| k(1-k)G\big[v_{k,t}(A_{k,t}, k, s_t) - (A_{1,t} - A_0)\big] \right\|_2^2\right]. \end{aligned} \tag{26}$$

For comparison, the generative-time objective is

$$\mathcal{L}_v = \mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\left\| v_{k,t}(A_{k,t}, k, s_t) - (A_{1,t} - A_0) \right\|_2^2\right]. \tag{27}$$

The expectations use $k \sim \mathcal{U}(0,1)$, $(s_t, A_{1,t}) \sim \mathcal{D}$, and $A_0 \sim \mathcal{N}(0, I)$. Equations (26) and (27) therefore contain the same velocity residual; the physical-time objective applies the linear transformation $k(1-k)G$ to it. The relation between their population optima is proved in the Theorem 1.

C Proof of Theorem

C.1 Proof of Theorem 1

Theorem 1. *Fix $k \in [0,1)$ and consider the conditional physical-time velocity field $u_{k,t} = k\dot{A}_{1,t}$. If its induced marginal field is continuous in t and uniformly Lipschitz in its action input, then it generates the marginal evolution of $A_{k,t}$ along physical time. In particular, the resulting marginal flow satisfies $(\Psi_{k,t\to k,t'})_{\#}p_{k,t} = p_{k,t'}$, where $p_{k,t}$ is the probability density path of $A_{k,t}$.*

Proof. Fix $k \in [0,1)$. The probability density path specified by Eq. (1) is

$$p_{k,t}(A_{k,t}) = \mathbb{E}\big[\mathcal{N}\big(A_{k,t};\, kA_{1,t}, (1-k)^2 I\big)\big].$$

To determine its evolution along physical time, keep the noise sample A_0 fixed as t varies and independent of the entire action trajectory. Equation (1) then gives

$$\partial_t A_{k,t} = k\dot{A}_{1,t} = u_{k,t}.$$

Holding the density argument $A_{k,t}$ fixed, differentiation along the coupled action trajectory yields

$$\begin{aligned}\partial_t p_{k,t}(A_{k,t}) &= \mathbb{E}\Big[\big(\nabla_{A_{1,t}}\mathcal{N}\big(A_{k,t};\, kA_{1,t}, (1-k)^2 I\big)\big)^\top \dot{A}_{1,t}\Big] \\ &= -\nabla_{A_{k,t}} \cdot \mathbb{E}\Big[k\dot{A}_{1,t}\,\mathcal{N}\big(A_{k,t};\, kA_{1,t}, (1-k)^2 I\big)\Big] \\ &= -\nabla_{A_{k,t}} \cdot \left[p_{k,t}(A_{k,t})\,\mathbb{E}[u_{k,t} \mid A_{k,t}]\right].\end{aligned}$$

Thus $p_{k,t}$ satisfies the continuity equation with induced marginal field $\mathbb{E}[u_{k,t} \mid A_{k,t}]$. Under the stated continuity and uniform Lipschitz conditions, the flow generated by this marginal field transports the density path:

$$(\Psi_{k,t\to k,t'})_{\#} p_{k,t} = p_{k,t'}.$$

Therefore the conditional physical-time velocity $u_{k,t} = k\dot{A}_{1,t}$ induces the claimed marginal evolution. □

C.2 Proof of Theorem 2

Theorem 2. *Under the linear transformations in Eq. (13) and some mild assumption on G, the physical-time velocity learning objective $\mathcal{L}_u$ can be written as a variant form of the flow-matching objective of $\mathcal{L}_v$ under a different Bregman divergence, thereby corresonding to the same optimal marginal velocity solution. Moreover, any population minimizer of $\mathcal{L}_v$ also minimizes $\mathcal{L}_u$.*

Proof. The result follows from the conditional-to-marginal equivalence of flow matching under Bregman divergences (Lipman et al., 2024, Sec. 4.5). For each fixed k, consider the convex quadratic potential

$$\Phi_k(x) := \|k(1-k)Gx\|_2^2. \tag{28}$$

Its associated, possibly degenerate, Bregman divergence is

$$\begin{aligned}D_{\Phi_k}(x, y) &:= \Phi_k(x) - \Phi_k(y) - \langle x - y, \nabla\Phi_k(y)\rangle \\ &= \|k(1-k)G(x-y)\|_2^2.\end{aligned} \tag{29}$$

The physical-direction objective in Eq.(15) can therefore be written as

$$\mathcal{L}_u = \mathbb{E}[D_{\Phi_k}(A_{1,t} - A_0,\, v_{k,t}(A_{k,t}, k, s_t))]. \tag{30}$$

Conditioning on $(A_{k,t}, k, s_t)$ and applying the Bregman conditional-to-marginal identity gives

$$\begin{aligned}&\mathbb{E}[D_{\Phi_k}(A_{1,t} - A_0,\, v_{k,t}(A_{k,t}, k, s_t)) \mid A_{k,t}, k, s_t] \\ &= D_{\Phi_k}(\mathbb{E}[A_{1,t} - A_0 \mid A_{k,t}, k, s_t],\, v_{k,t}(A_{k,t}, k, s_t)) \\ &\quad + \mathbb{E}[\Phi_k(A_{1,t} - A_0) \mid A_{k,t}, k, s_t] - \Phi_k(\mathbb{E}[A_{1,t} - A_0 \mid A_{k,t}, k, s_t]).\end{aligned} \tag{31}$$

The last two terms are independent of the learned velocity field. Hence, $\mathcal{L}_u$ is minimized whenever

$$k(1-k)G\left(v_{k,t}(A_{k,t}, k, s_t) - \mathbb{E}[A_{1,t} - A_0 \mid A_{k,t}, k, s_t]\right) = 0 \tag{32}$$

almost surely. In particular,

$$v_{k,t}(A_{k,t}, k, s_t) = \mathbb{E}[A_{1,t} - A_0 \mid A_{k,t}, k, s_t] \tag{33}$$

is a population minimizer of $\mathcal{L}_u$.

The squared Euclidean objective $\mathcal{L}_v = \mathcal{L}_{\mathrm{FM}}$ is itself a Bregman flow-matching loss and has Eq. (33) as its unique unrestricted population minimizer. Therefore, every population minimizer of $\mathcal{L}_v$ also minimizes $\mathcal{L}_u$. Since G may have a nontrivial null space, $\mathcal{L}_u$ may admit additional minimizers. □

D EXPERIMENTAL DETAILS

D.1 CHOICE OF INTERPOLATION TECHNIQUE

Finite differences. The simplest estimate of $\dot{A}_{1,t}$ takes differences between adjacent actions in $A_{1,t}$ and divides by the sampling interval Δ. This is inexpensive and requires no fitted trajectory model. However, differencing amplifies sample-to-sample noise, and one-sided estimates at the ends of a short chunk can be particularly sensitive to individual action labels. It also gives derivatives only at sampled times, without a continuous trajectory from which to evaluate them. Since these estimates serve as targets for the first-order objective, their variability can directly affect its supervision.

B-spline reconstruction. We instead fit the degree-p B-spline described in Section 5 to each action dimension of $A_{1,t}$ and differentiate the fitted curve. The compact support of its basis functions provides local shape control: a change in one part of a chunk can be represented without requiring a global change in the continuous curve. The fit yields the fixed linear differentiation operator G used in Eq.(17). Choosing fewer control points than action samples ($M < H$) can additionally suppress high-frequency variation, but we found $M = H$ sufficient in both simulation and real-robot experiments. Thus, our choice of B-splines does not rely on reducing the number of control points to filter the action labels. The spline degree and number of control points still determine the reconstruction bias; overly coarse fits may blur abrupt but meaningful action changes. We use B-splines because they provide a continuous, analytically differentiable trajectory while retaining flexibility for local changes in the demonstrated actions.

Discrete cosine transform. A discrete cosine transform (DCT) offers another differentiable basis representation of $A_{1,t}$. Retaining low-frequency coefficients can give a compact, smooth reconstruction, and its derivative can likewise be computed with a fixed linear operator. However, cosine basis functions span the entire chunk, so a localized transition affects the reconstruction across the chunk. Truncating the spectrum may smooth away brief, task-relevant changes, whereas retaining more high-frequency components reduces that smoothing and can make the derivative more sensitive to noise. The cosine extension also imposes boundary behavior that need not match the beginning or end of a demonstrated chunk. For our first-order targets, we therefore prefer the local shape control of B-splines over a globally supported DCT basis.

D.2 CHOICE OF STATE

Differentiable conditioning. The state s_t in our derivation denotes the policy conditioning along which a physical-time tangent is taken, rather than every field in a raw observation. We use continuous proprioceptive quantities (e.g., joint positions and end-effector pose) and per-frame visual features produced by the policy's existing encoder, so the tangent is expressed in the numerical space seen by its action head. For a continuous feature z_t, its tangent is estimated from episode-local neighboring frames; with both neighbors available, the central estimate is $(z_{t+1} - z_{t-1})/(2\Delta)$. A fixed task instruction has zero physical-time tangent. Categorical variables may still condition the policy. We set their tangent to zero in the Jacobian–vector product because differences between category codes or token IDs do not define derivatives in the model's input space. Thus, selecting a state tangent does not require removing useful conditioning information from the ordinary policy forward pass.

Gripper channels. In particular, a binary open/close gripper command is not a continuously varying state or action coordinate. Its finite difference is zero between switches and produces a sampling-rate-dependent spike at a switch; fitting a smooth curve would instead invent intermediate commands. We therefore exclude discrete gripper commands from both the differentiated conditioning components and the first-order action target, while retaining their standard action-learning objective. This distinction applies to the command representation. A measured, continuous gripper joint position or aperture can be treated as proprioception, whereas a binarized gripper readout must be held fixed in the state tangent.

Choices for the evaluated policies. For Multi-task DiT and the flow-matching Diffusion Policy on LIBERO, we take the tangent of their existing per-step conditioning vector, which contains robot

proprioception and encoded camera observations; the instruction embedding in Multi-task DiT is held fixed. The LIBERO robot-state vector includes measured gripper joint positions, whereas the gripper action is an open/close command; the latter is excluded from the auxiliary action loss. For X-VLA in both LIBERO and real-robot experiments, we use its VLM and auxiliary visual features for the conditioning tangent and set the proprioceptive tangent to zero in the visual-only JVP, while retaining proprioception in the forward pass. On LIBERO, its first-order action loss covers the end-effector translation and 6D rotation coordinates, excluding the gripper and padded arm channels. On the dual-arm robot, it selects the twelve arm-joint coordinates and excludes the two gripper coordinates. In $\pi_{0.5}$, proprioception is quantized into the language prefix; we therefore set the raw-state tangent to zero and take the conditioning tangent only through the visual prefix embeddings, leaving the task and quantized-state tokens fixed. Its first-order action loss likewise selects the twelve arm-joint coordinates and omits the two gripper coordinates. Finally, for the OGBench policies, we use the benchmark's continuous low-dimensional observation as the state and estimate its tangent by central differences. Across all backbones, these choices affect only the auxiliary first-order objective; the policy's original conditioning inputs and inference procedure are unchanged.

D.3 LIBERO and LIBERO-Plus Experimental Details

Training and evaluation protocol. We train the imitation-learning policies on demonstrations from the four LIBERO suites (Liu et al., 2023a), with ten tasks each in Spatial, Object, Goal, and LIBERO-10. LIBERO-Plus (Fei et al., 2025) is used only for evaluation, so none of its perturbations enter policy training. Multi-task DiT (TRI LBM Team et al., 2025) and the flow-matching Diffusion Policy (Chi et al., 2025) use the LeRobot (Cadene et al., 2024) LIBERO data and its seven-dimensional relative end-effector actions. X-VLA (Zheng et al., 2026) uses the official X-VLA-formatted LIBERO demonstrations and its native absolute end-effector action representation. Because the action representations and pretrained components differ across backbones, each Base and +Ours comparison uses the same data format, initialization scheme, and inference procedure within that backbone. We report success on clean LIBERO and evaluate robustness under LIBERO-Plus lighting, sensor-noise, and robot-initialization changes for all three policies. Background, camera, object, and language perturbations are additionally evaluated for X-VLA, as described in Appendix E.

Multi-task DiT. We train one language-conditioned Multi-task DiT policy jointly on all 40 LIBERO tasks. Its visual and text encoders are initialized from pretrained CLIP ViT-B/16 (Radford et al., 2021) weights, while the action-generating DiT is trained on LIBERO demonstrations without a pretrained robot-policy checkpoint. The policy conditions on two observation steps, predicts a 48-step action chunk, and executes 24 actions before replanning. Both variants use the same flow-matching backbone; +Ours adds the first-order loss during training. Thus, the use of pretrained CLIP encoders should not be confused with pretraining the complete DiT policy on robot trajectories.

Diffusion Policy architecture. We use the LeRobot conditional one-dimensional U-Net architecture of Diffusion Policy, but train it with flow matching (Lipman et al., 2023) instead of the original diffusion objective. Its ResNet-18 (He et al., 2016) visual encoder starts from ImageNet-pretrained weights (Deng et al., 2009), whereas the action generator is trained on LIBERO rather than initialized from a pretrained robot policy. We train a separate policy for each LIBERO suite. As with Multi-task DiT, it conditions on two observation steps, predicts 48 actions, and executes the first 24 before replanning. The Base and +Ours variants share this architecture and differ by the additional first-order training objective.

X-VLA. Unlike the two lightweight policies, X-VLA (Zheng et al., 2026) is initialized from the pretrained `2toINF/X-VLA-Pt` checkpoint and then fine-tuned on LIBERO. We fine-tune it across the four suites using the official X-VLA action format, with 30-step action chunks and 30 executed actions per prediction. Both the Base and +Ours runs load the same pretrained checkpoint. The latter adds first-order supervision to the continuous end-effector translation and 6D rotation channels, while the gripper remains under the original action-learning objective. The model architecture and inference procedure are unchanged.

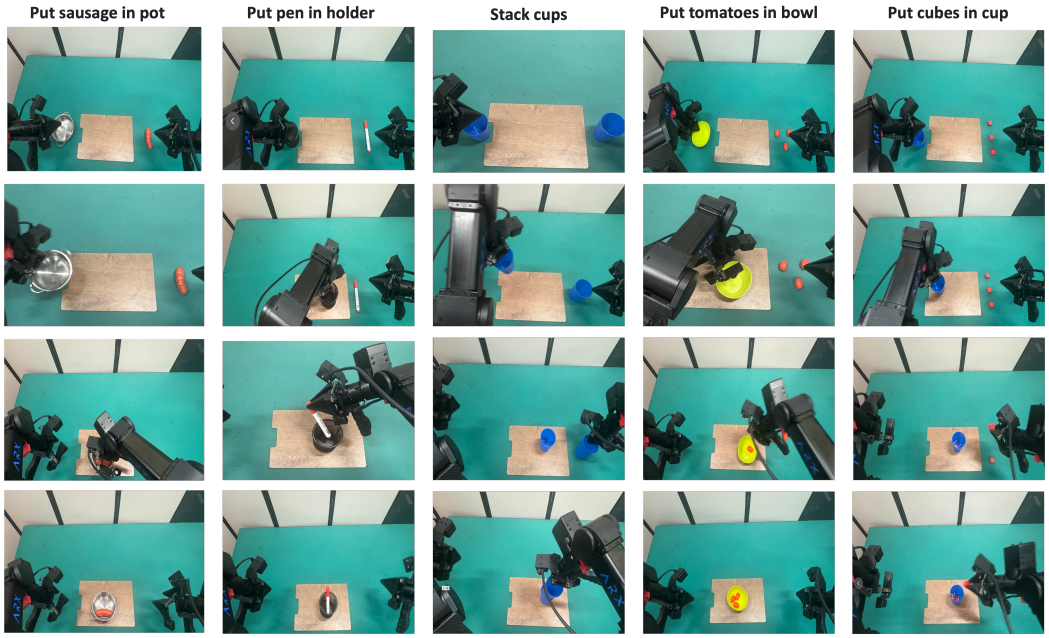

Figure 4: Demonstration of the real-world robot control tasks.

D.4 Real-Robot Experimental Details

Platform and tasks. We evaluate on the dual-arm ARX A5 Beta platform[2], with six arm joints and one gripper per arm. Each observation contains a front RGB image, two wrist RGB images, measured robot state, and a task instruction. Policies predict 14-dimensional absolute joint-space actions: six joint targets and one gripper target for each arm. Images and demonstrations are recorded at 30 Hz. Both $\pi_{0.5}$ and X-VLA are evaluated on *Put Sausage in Pot*, *Put Pen in Holder*, and *Stack Cups*; $\pi_{0.5}$ is additionally evaluated on *Put Tomatoes in Bowl* and *Put Cubes in Cup*. The ARX deployment uses a 50-step action horizon for $\pi_{0.5}$ and a 30-step horizon for X-VLA. The figure 4 shows the tasks demonstration.

Initialization and fine-tuning. We initialize $\pi_{0.5}$ from the released OpenPI `pi05_base` checkpoint (Intelligence et al., 2025) and X-VLA from the released `2toINF/X-VLA-Pt` checkpoint (Zheng et al., 2026). We retain their pretrained model architectures and adapt their action interfaces to the ARX joint space: the 14 robot-action coordinates are padded to the respective 32- and 20-dimensional model action spaces. We fine-tune a separate policy for each task, using 50 demonstration trajectories and 8,000 training steps for $\pi_{0.5}$, and 100 trajectories and 20,000 steps for X-VLA. Within each backbone and task, Base and +Ours use the same initialization, demonstrations, action representation, training schedule, and deployment procedure; +Ours adds only the first-order training objective. These are ARX-specific fine-tuning settings built on the released OpenPI and X-VLA models, rather than official A5 configurations supplied by their original authors.

State tangent and Jacobian–vector product. For the auxiliary objective, we differentiate the conditioning variables that have a meaningful continuous physical-time tangent. In $\pi_{0.5}$, measured proprioception is quantized into the language prefix. We therefore hold the task and quantized-state tokens fixed and estimate the tangent of the visual prefix embeddings by a central difference between neighboring observations. A single `jax.jvp` differentiates the action-generating velocity map with respect to these prefix embeddings and the noised action chunk; it does not take a JVP through the image encoder. In X-VLA, the differentiated conditioning consists of the Florence-2 VLM features and auxiliary visual features, with tangents estimated by a forward difference between adjacent observations. The task instruction and proprioception remain available to the ordinary forward pass

[2] https://github.com/ARXroboticsX/A5_beta

but are held fixed in this JVP. A single `torch.func.jvp` is applied to X-VLA's action-generating Transformer with respect to the visual features and noised action chunk, rather than through the Florence-2 encoder. For both policies, we obtain the action tangent by analytically differentiating a quadratic B-spline fitted along each demonstrated action chunk. The first-order loss supervises the twelve arm-joint coordinates and excludes the two gripper coordinates; the standard action-learning loss still covers the grippers. The auxiliary objective is used only during training, leaving policy inference unchanged.

D.5 OGBench Experimental Details

We evaluate our method in the offline RL setting on 15 single-task environments from OGBench (Park et al., 2025b), comprising five tasks each from "scene-sparse", "cube-double", and "cube-triple". We incorporate our auxiliary first-order objective into both Q-Chunking (QC) (Li et al., 2025) and its IFQL variant, QC-IFQL, modifying only their flow-based action-chunk policies. Specifically, for each demonstrated action chunk, we estimate the action tangent by fitting a quadratic B-spline and analytically differentiating the fitted trajectory. We use five control points, equal to the action-chunk horizon $H = 5$, and estimate the state tangent using the central difference between the neighboring states. The directional derivatives of the conditional flow field with respect to the state and noisy action chunk are computed using Jacobian–vector products, without explicitly materializing either Jacobian.

Specifically, the resulting objective is added to the original flow-matching loss as described in Eq. (16), where we use $\lambda_{\text{1st}} = 0.1$ and the B-spline derivative estimator for every task and both offline RL methods. Chunks that cross episode boundaries are excluded from the first-order loss. All remaining architecture, optimization, sampling, and evaluation hyperparameters follow the corresponding defaults in the official QC implementation. In particular, the critic and value updates, as well as the Q-guided selection among generated action chunks, remain unchanged.

E Additional Results

E.1 Full results on LIBERO plus

Table 3: Evaluation results under different perturbations on LIBERO-Plus (Fei et al., 2025), averaged over training seeds 1000 and 1001. Each cell shows X-VLA → X-VLA + Ours; bold marks improvements, and red ↑ labels give gains in percentage points.

Perturbation	Spatial	Object	Goal	LIBERO-10
Background	91.47 → 87.40	98.79 → 96.58	86.84 → **88.97** ↑2.13	82.87 → **88.24** ↑5.37
Camera	29.79 → 29.26	37.12 → 36.24	35.91 → 33.09	22.91 → 16.83
Language	71.41 → 68.85	93.22 → 86.73	45.73 → **46.95** ↑1.22	75.59 → **81.60** ↑6.01
Lighting	92.30 → 89.38	99.67 → 99.16	80.11 → **91.76** ↑11.65	86.68 → **93.43** ↑6.75
Object	80.26 → **82.60** ↑2.34	59.43 → 55.09	48.83 → **54.36** ↑5.53	57.86 → **66.19** ↑8.33
Robot	86.00 → **87.00** ↑1.00	94.60 → **97.36** ↑2.76	81.05 → **86.80** ↑5.75	73.79 → **83.84** ↑10.05
Sensor	62.40 → 59.97	64.81 → 60.79	50.27 → 47.76	53.79 → **55.79** ↑2.00
Overall	71.82 → 70.69	75.76 → 73.50	59.09 → **61.78** ↑2.69	62.51 → **66.72** ↑4.21

Table 3 shows overall gains on Goal and LIBERO-10, and robot perturbations improve across all four suites. In contrast, the Spatial and Object averages decline slightly, while camera perturbations reduce success in every suite. One possible explanation is that first-order supervision helps preserve coherent action evolution when the task remains recognizable, but does not directly make the policy robust to changes in viewpoint or other observation cues.