
Implicit Drifting Policy: One-Step Action Generation via Conditional Expert Geometry

Zemin Yang¹, Yaoyu He¹, Yiming Zhong¹, Yuhao Zhang², Xinge Zhu³
Yao Mu², Qingqiu Huang⁴, Yuexin Ma^{1*}

¹ShanghaiTech University ²Shanghai Jiao Tong University

³The Chinese University of Hong Kong ⁴Morphi Robot

{csyangzm, mayuexin}@shanghaitech.edu.cn

Project Page: <https://implicit-drifting-policy.github.io/>

Abstract

Generative action policies based on diffusion or flow matching excel in behavior cloning, yet their iterative sampling is prohibitive for high-frequency robot control. While recent one-step formulations alleviate this latency, they inevitably discard the intermediate trajectory evolution that provides crucial action correction. Directly recovering this mechanism by explicitly estimating a training-time drifting field is challenging under sparse conditional demonstrations. We introduce **Implicit Drifting Policy (IDP)**, a one-step imitation learning framework that brings the training-time correction of Drifting into policy learning without explicit vector field estimation. IDP extracts a **conditional expert geometry** from the local variation of observation-similar expert actions, and compares it against a global **reference geometry** to isolate condition-specific constraints. This local geometric structure adaptively weights a scalar potential objective. Combined with an expert-proximal terminal evaluation, IDP incorporates local geometric correction into the one-step generator during training. Extensive evaluations across 2D, 3D, and real-world manipulation tasks show IDP effectively maintains adherence to valid action manifolds, improving upon explicit drifting methods and achieving competitive performance with strong one-step baselines.

1 Introduction

Robot imitation learning maps observations to continuous actions. To achieve this, behavior cloning commonly trains a policy with a regression loss to predict an expert action or action sequence from an observation. This form is simple and efficient, but it compresses action generation into a single prediction. Diffusion Policy introduced diffusion models into this setting by formulating action prediction as conditional denoising, gradually generating actions from noise toward expert actions [5]. Subsequent flow-based policies further describe action generation with continuous flows or flow matching [34]. Compared with direct prediction, diffusion and flow policies retain intermediate states during generation, allowing an action to be formed through multiple updates. This intermediate computation is one of the key distinctions between generative policies and regression-based policies.

The inference cost of generative policies has motivated faster action generation methods. Consistency Policy and One-Step Diffusion Policy compress diffusion policies through consistency training or distillation [27, 31]. Shortcut Models and flow-map methods learn longer-range generative maps to reduce sampling steps [10, 22]. Beyond sampling speed, a more basic question is what multi-step generation contributes to control. Much Ado About Noising analyzes generative robotic control from this perspective and argues that its benefits are closely tied to stochasticity injection and supervised intermediate computation, rather than multimodality alone [25]. This raises a question for one-step policies: *when inference no longer contains multi-step updates, how can training still provide a useful action-correction signal?*

*Corresponding author.

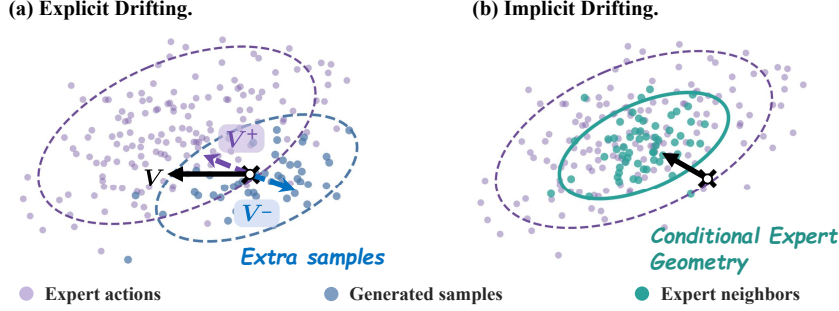


Figure 1: **(a) Explicit Drifting** constructs targets by estimating a drifting field between expert actions (purple) and policy predictions (blue). Relying on the evolving policy state and mini-batch sampling can make the target sensitive to minibatch composition and the current policy state. **(b) Implicit Drifting Policy (Ours)** directly extracts a Conditional Expert Geometry (green) from expert actions under similar observations.

Drifting models study a related mechanism in generative modeling. Drifting trains a single-pass generator and uses a drifting field during training to move the current model distribution toward the data distribution [7]. Recent analyses connect this field to score-based transport directions and terminal data-grounded correction [18, 22]. Here, correction affects the learning dynamics while inference remains a single generator evaluation. This makes Drifting relevant to one-step action policies: the training objective itself can carry part of the action-correction role, reducing reliance on additional sampling steps at deployment. However, behavior cloning provides a different form of supervision. For a given observation, demonstrations usually provide a single expert action. Estimating a drifting field around the policy output thus requires batch neighbors and the prediction location to approximate local conditional structure. Such estimates vary with the mini-batch and evolving policy state, making the target sensitive to implementation details (Fig 1 (a)).

We argue that a reliable correction signal should instead emerge from the local structure already present in demonstrations. A single observation provides one expert action, while expert actions under similar observations provide local references. Similar observations often correspond to nearby task states or control contexts. Within these local references, variation among expert actions indicates which action directions are more consistent near the current condition and which directions allow more variation. We use this demonstration-supported information to build a training signal that emphasizes errors along locally important action directions, without estimating a drifting field as a separate supervision target. This keeps the training-time correction view of Drifting while grounding the correction source in local action structure observable in imitation learning.

Motivated by this, we propose **Implicit Drifting Policy (IDP)**. IDP formulates an implicit correction objective that minimizes a geometry-aware potential. The core of this potential lies in its adaptive directional weights, which we derive by extracting a **conditional expert geometry** from the local variation of expert actions under similar observations (Fig 1 (b)). To isolate condition-specific constraints, we further compare this local geometry with a global **reference geometry**, ensuring that inherently low-variance dimensions are not mistakenly penalized. Finally, to ensure the network effectively captures the local manifold topology defined by this potential, IDP introduces an expert-proximal evaluation during training, exposing the one-step generator to the local correction structure near expert actions. Across 2D, 3D imitation tasks, and real-world robot experiments, IDP improves over explicit drifting variants and remains competitive with strong one-step policy baselines. In summary, our contributions are as follows:

- We introduce **Implicit Drifting Policy (IDP)**, a one-step imitation learning framework. By minimizing a geometry-aware potential and introducing an expert-proximal training evaluation, IDP brings the training-time correction of drifting into policy learning.
- We propose a method to extract local action correction structures by comparing the **conditional expert geometry** under similar observations against a global **reference geometry**. This adaptively isolates condition-specific constraints to weight the correction potential.
- We evaluate IDP across 2D, 3D, and real-world manipulation tasks. Results demonstrate that IDP outperforms explicit drifting variants and exhibits stronger adherence to local action constraints over strong one-step baselines.

2 Related Work

Generative action policies and one-step acceleration. Continuous-action imitation learning commonly uses behavior cloning, where a policy predicts expert actions from observations. Diffusion Policy formulates action prediction as conditional denoising, so that actions are formed through multiple denoising steps [5]. Flow matching provides a continuous-time transport formulation for generative modeling [23], and has been used for policies with point-cloud inputs, spatially structured action representations, 3D manipulation, and action-to-action generation [6, 11, 34, 16]. Recent work reduces the inference cost of generative policies from several directions. Consistency Policy [27] and ManiCM [24] use consistency models to accelerate diffusion policies. One-Step Diffusion Policy [31], One-Step Flow Policy [20], and MP1 [30] obtain single-step policies through distillation, self-distillation, or average-velocity learning. Streaming Diffusion Policy [13], Falcon [4], and Real-Time Iteration Scheme [8] reduce, reorganize, or reuse the denoising process to lower latency. In broader generative modeling, Shortcut Models [10], Mean Flows [12], flow-map distillation [2], and Align Your Flow [29] study long-interval or one-step generative mappings. Much Ado About Noising [25] moves the discussion from sampling steps to mechanism, arguing that the benefit of generative control policies is closely tied to stochasticity injection and supervised intermediate computation. Our work follows this concern for one-step policy training and asks how training can retain an action-correction signal once deployment uses a single policy evaluation.

Drifting models and training-time correction. Drifting models study one-step generative modeling from the perspective of training-time distribution evolution, where a drifting field describes the direction that moves the current model distribution toward the data distribution. Generative Modeling via Drifting [7] trains a single-pass generator and uses this drifting field during training. Subsequent analysis relates the drifting field to score-based transport directions [18]. Long-Short Flow-Map [22] further interprets data-grounded terminal correction as supervision for a long map. These works provide a view of training-time correction for one-step generation, primarily in distribution-level generative modeling. Behavior cloning provides a more specific form of supervision. Since demonstrations usually provide only a single deterministic expert action for a given observation, estimating a continuous drifting field around the current policy output is forced to depend on local mini-batch neighborhoods and the evolving prediction location. We build on the training-time correction view of Drifting while grounding the correction signal in local information available in imitation learning.

Geometric structure in imitation learning. Imitation learning has long used demonstrations to represent action distributions, trajectory structure, and variability. Gaussian mixture regression, dynamic movement primitives, probabilistic movement primitives, kernelized movement primitives, and stable dynamical systems learn local means, covariances, temporal evolution, or stable control fields from demonstration trajectories [3, 15, 26, 14, 17]. More recently, Geometry-aware Policy Imitation treats demonstrations as geometric curves and derives distance fields that induce attraction and progression flows for policy synthesis [21]. Another line of deep imitation learning focuses on multimodal continuous actions. Implicit Behavioral Cloning uses energy-based objectives, and VQ-BeT uses latent action tokens for behavior generation [9, 19]. These works show that imitation data contains structure beyond a single regression target. Our use of geometry focuses on the local variation of expert actions under similar observations as a training-time correction signal for one-step policies. Compared with distance-field policy synthesis or action-distribution modeling, our objective uses local geometry during training while preserving single-evaluation deployment.

3 Preliminaries

From Multi-Step to One-Step Action Generation. In conditional imitation learning, given a dataset of expert demonstrations $\mathcal{D} = \{(o_i, a_i^*)\}_{i=1}^N$, the objective is to learn a policy mapping an observation $o \in \mathcal{O}$ to an expert action chunk a^* from the target data distribution $p_{\text{data}}(a^*|o)$. Traditional behavior cloning (BC) frames this as a single-step direct mapping $a = f_{\theta}(o)$, optimizing the isotropic mean squared error $\mathbb{E}_{o, a^* \sim p_{\text{data}}} \|f_{\theta}(o) - a^*\|_2^2$. While deployment is highly efficient, this objective reduces generation to an isolated endpoint prediction, failing to exploit the geometric structure of the expert action manifold. Generative action policies address this by constructing a continuous trajectory over a virtual time $t \in [0, 1]$, where a_t denotes the intermediate state. Starting from an initial source sample $a_0 \sim p_0$, diffusion models progressively reach the final action via a

denoising path, typically optimized using the denoising loss $\mathcal{L}_{\text{DP}} = \mathbb{E}[\|\epsilon_\theta(a_t, o, t) - \epsilon\|_2^2]$. Evolving from this, flow matching directly formulates the trajectory via an ordinary differential equation (ODE) $da_t = v_\theta(a_t, o, t)dt$ driven by a velocity field, optimizing $\mathcal{L}_{\text{FM}} = \mathbb{E}[\|v_\theta(a_t, o, t) - \frac{d}{dt}a_t\|_2^2]$. This multi-step evolution naturally retains the capacity for progressive error correction.

To reduce inference latency, recent acceleration algorithms compress the intermediate path to directly learn a one-step endpoint mapping or a large-stride velocity. For instance, Consistency Policy [27] learns a mapping $f_\theta(o, a_t, t)$ by imposing a cross-time self-consistency constraint: $\mathcal{L}_{\text{CM}} = \mathbb{E}[\|f_\theta(o, a_t, t) - f_\phi(o, a_{t'}, t')\|_2^2]$, enforcing any intermediate point on the trajectory to map to the same endpoint (where ϕ is the target network parameter). Similarly, MeanFlow [12] learns the average velocity $u_\theta(a_t, o, r, t)$ over a time interval $[r, t]$ by satisfying an intrinsic identity between average and instantaneous velocities: $\mathcal{L}_{\text{MF}} = \mathbb{E}[\|u_\theta(a_t, o, r, t) - \text{sg}[v_t - (t - r)\frac{d}{dt}u_\theta(a_t, o, r, t)]\|_2^2]$, where $\text{sg}[\cdot]$ denotes the stop-gradient operator. However, whether relying on target-network distillation or higher-order derivative identities, the optimization processes encounter specific challenges. When the virtual intermediate evolution is bypassed, reconstructing the corrective signal for the one-step mapping without relying on expensive cross-step constraints remains a crucial problem.

Explicit Drifting in Behavior Cloning. Drifting models offer a mathematical framework to isolate the correction mechanism entirely within the training phase. Given the target data distribution p_{data} and generated distribution q_θ , Drifting constructs an explicit continuous vector field $V_{p,q}(a) = V_p^+(a) - V_q^-(a)$ comprising attraction and repulsion. This field dictates the optimal displacement for a generated sample a toward the data manifold. Based on a kernel function $k(\cdot, \cdot)$, the attraction and repulsion fields are respectively defined as:

$$V_p^+(a) = \frac{\mathbb{E}_{y \sim p_{\text{data}}}[k(a, y)(y - a)]}{\mathbb{E}_{y \sim p_{\text{data}}}[k(a, y)]}, \quad V_q^-(a) = \frac{\mathbb{E}_{y \sim q_\theta}[k(a, y)(y - a)]}{\mathbb{E}_{y \sim q_\theta}[k(a, y)]}. \quad (1)$$

The one-step generator f_θ is trained to regress this field-displaced frozen target:

$$\mathcal{L}_{\text{Drift}}(\theta) = \mathbb{E}_{a_0 \sim p_0} \|f_\theta(o, a_0, 0) - \text{sg}[f_\theta(o, a_0, 0) + V_{p,q}(f_\theta(o, a_0, 0))]\|_2^2. \quad (2)$$

In unconditional generation, the rich sample sets supporting the distributions enable the vector field to provide smooth manifold guidance. However, directly porting this explicit drifting mechanism to conditional behavior cloning encounters a fundamental structural degeneration. Consider the following proposition:

Proposition 3.1 (Degeneration of Explicit Drifting in Behavior Cloning). *Assume that in behavior cloning, the empirical expert conditional contains only one recorded action a_i^* for a specific observation o_i , and the generator produces a single prediction $a = f_\theta(o_i, a_0, 0)$ per training iteration. The empirical conditional distributions degenerate into Dirac measures $\hat{p}_{\text{data}}(y|o_i) = \delta(y - a_i^*)$ and $\hat{q}_\theta(y|o_i) = \delta(y - a)$. For any translation-invariant positive kernel $k(\cdot, \cdot)$, the empirical conditional drifting field mathematically evaluates exactly to $V_{\hat{p}, \hat{q}}(a|o_i) = a_i^* - a$. Consequently, the field-displaced regression objective becomes exactly equivalent to the isotropic Mean Squared Error, losing all local geometric guidance. (Proof. See Appendix A.1.)*

While the underlying control task may tolerate variations around a_i^* (e.g., alternative valid trajectories or recoverable states), the preceding analysis highlights that the empirical target distribution in standard BC training is too sparse to directly support the estimation of a continuous drifting field. To circumvent this degeneration in practice, models must resort to empirical approximations, such as synthesizing a localized empirical distribution by drawing multiple independent noise samples per observation during training and applying self-masking heuristics. While successful in avoiding the collapse to MSE, these approximations significantly increase computational overhead and entangle the correction field with transient sampling densities. This practical challenge motivates us to explore an alternative path: rather than regressing an explicit dynamic vector field, we extract the static local geometric structure directly from demonstrations.

4 Implicit Drifting Policy

Given the challenges of estimating explicit continuous vector fields under sparse empirical distributions, we explore an alternative path. Our core idea is to shift away from regressing dynamic

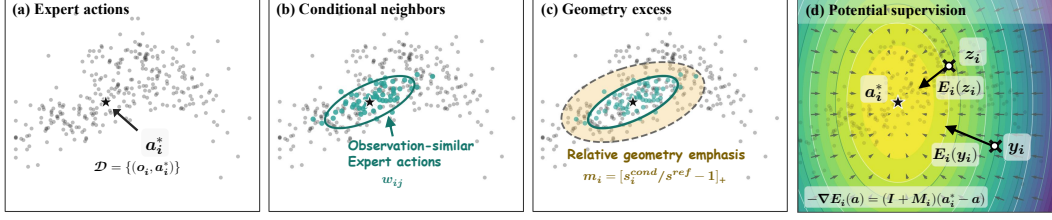


Figure 2: **Overview of Implicit Drifting Policy.** (a) Expert action a_i^* anchors the local action space. (b) Weights w_{ij} select observation-similar expert actions to form a Conditional Expert Geometry G_i around a_i^* . (c) Comparing G_i with reference geometry Σ_{ref} yields the Local Geometry Excess m_i . (d) The induced metric M_i defines a potential E_i , where $-\nabla E_i(a)$ supervises both deployment proposal y_i and expert-proximal evaluation z_i .

drifting fields, and instead consider extracting the local geometric structure directly from static expert demonstrations, converting it into a potential objective that induces correction (see Fig 2).

4.1 Implicit Correction Objective and Local-Aware Evaluation

To avoid regressing a high-dimensional and unstable continuous drifting field, we consider constructing an implicit correction field directed toward the expert target a_i^* . In the most basic case, a simple attraction field can be written as $V_{\text{base}}(a) = a_i^* - a$. However, this simple attraction is isotropic. In multi-step generative models, isotropic targets, like MSE loss, are sufficient because the model can implicitly adjust and align with the action manifold through hundreds of inference iterations along the intermediate trajectory. However, for a one-step generator, it loses the ability to progressively correct errors along the intermediate path. If the training objective does not explicitly reinforce the tightness of constraints across different action dimensions, the prediction is highly prone to diverging from the valid manifold structure in a single stride. Therefore, for a given predicted action a , we consider applying a geometry-aware correction vector:

$$\Delta_i(a) = M_i(a_i^* - a), \quad (3)$$

where $M_i \succeq 0$ is a positive semi-definite symmetric matrix acting as geometry-adaptive weights, determining the strength of the correction force along different directions.

For fixed symmetric $M_i \succeq 0$, the geometric correction is the negative gradient of the following potential (Appendix A.2):

$$E_i^{\text{geo}}(a) = \frac{1}{2}(a - a_i^*)^\top M_i(a - a_i^*). \quad (4)$$

The complete geometric potential objective incorporates the isotropic error of behavior cloning:

$$E_i(a) = \frac{1}{2}\|a - a_i^*\|_2^2 + E_i^{\text{geo}}(a) = \frac{1}{2} \underbrace{\|a - a_i^*\|_2^2}_{\text{isotropic error}} + \underbrace{(a - a_i^*)^\top M_i(a - a_i^*)}_{\text{geometry-adaptive penalty}} \quad (5)$$

whose full negative gradient naturally yields a combined correction force: $-\nabla_a E_i(a) = (I + M_i)(a_i^* - a)$ (Fig 2 (d)). The first term of this potential is equivalent to the isotropic error of behavior cloning, while the second term adaptively increases the penalty for constrained directions based on M_i . We transform the regression of an explicit vector field into the minimization of a potential.

However, the constraint characterized by matrix M_i is statistically grounded primarily within the local vicinity of the expert action a_i^* . If the optimization process applies this potential constraint exclusively at the standard initial prediction point $y_i = f_\theta(o_i, a_0, 0)$, the network might struggle to adequately learn the true topology of the local manifold. To ensure that this local potential effectively guides network learning, we consider introducing an expert-proximal evaluation anchor during training. By selecting a time scalar $t_* \in (0, 1)$ close to 1, we construct a probing input $\tilde{a}_i = a_i^* + (1 - t_*)\epsilon_i$ (where $\epsilon_i \sim \mathcal{N}(0, I)$) by injecting exploratory noise proportional to the distance around the expert action. We evaluate the mapping at this proximal location, yielding $z_i = f_\theta(o_i, \tilde{a}_i, t_*)$. Here, t indexes the virtual generation stage of the action input. We use $a_0 = 0$ for the direct prediction. Therefore, the single-step prediction and the local probing are controlled by the same geometric potential objective, formulating our total training objective as:

$$\mathcal{L} = \mathbb{E}_{i \sim \mathcal{D}} [E_i(y_i) + \lambda_{\text{prox}} E_i(z_i)], \quad (6)$$

where λ_{prox} is a hyperparameter balancing the standard endpoint regression and the expert-proximal evaluation. Implementation settings are given in Appendix D.2. The expectation also covers the input perturbation noise. This construction establishes the skeleton of the entire implicit correction framework. The core challenge then becomes how to extract the adaptive constraint matrix M_i from static demonstration data to reflect the current control state.

4.2 Conditional Expert Geometry as the Correction Source

To ensure the adaptive penalty effectively limits invalid actions, we consider grounding the construction of matrix M_i in the authentic structure of the demonstration data. In control manifolds, similar observation states usually correspond to analogous robot poses or environmental layouts, which physically often impose similar operational bottlenecks and obstacle avoidance constraints. Under similar observations, expert-action variation provides a local reference for which action coordinates are more consistent. Comparing this variation with the minibatch reference lets us strengthen the potential along relatively more consistent coordinates while retaining the baseline penalty elsewhere.

To extract this structure, we need to identify and aggregate similar control contexts in the feature space. We normalize the embeddings to remove feature-magnitude effects and apply stop-gradient to keep the geometry statistics fixed during backpropagation:

$$h_i = \frac{\text{sg}[\phi_\theta(o_i)]}{\|\text{sg}[\phi_\theta(o_i)]\|_2 + \varepsilon}. \quad (7)$$

To evaluate the correlation between different states, we first compute the pairwise feature inner products $s_{ij} = h_i^\top h_j$. The scale of similarity differences can vary across rows, affecting how concentrated the softmax weights become. To mitigate this issue, we adopt row-normalization $\bar{s}_{ij} = \frac{s_{ij} - \mu_i}{\sigma_i + \varepsilon}$, and compute the neighborhood weights $w_{ij} \propto \exp(\bar{s}_{ij})$ by applying softmax over all B samples in the minibatch. Based on these weights, to quantify the degree of action divergence under the current observation, we define the Conditional Expert Geometry matrix within the local space of the expert action a_i^* (Fig 2 (b)):

$$G_i = \sum_{j=1}^B w_{ij} (a_j^* - a_i^*)(a_j^* - a_i^*)^\top. \quad (8)$$

G_i describes the spread of related expert actions around a_i^* . Smaller spread along a direction indicates more consistent expert actions around the target. This local geometry provides the basis for constructing the correction weights M_i (Appendix A.3).

4.3 Local Geometry Excess

Directly utilizing the inverse of G_i as M_i could introduce biases. This is because certain action coordinates may naturally maintain extremely low variance globally across the entire task. We consider it necessary to distinguish the condition-specific constraints triggered purely by the current observation from this inherent global action scale.

In a general multivariate form, distinguishing the condition-specific constraints from the global marginal prior typically requires comparing their inverse covariance matrices. Let $P_i^{\text{cond}} = (G_i + \varepsilon I)^{-1}$ and $P^{\text{ref}} = (\Sigma_{\text{ref}} + \varepsilon I)^{-1}$. The geometry excess can be conceptualized via a non-negative truncated comparison, such as $M_i \propto [(P^{\text{ref}})^{-1/2} P_i^{\text{cond}} (P^{\text{ref}})^{-1/2} - I]_+$, which usually involves expensive generalized eigenvalue decomposition operations. Considering computational efficiency and to ensure relatively independent penalties across dimensions, we adopt a diagonal instantiation in practice. Specifically, we compute the diagonal conditional variance $v_{i,d}^{\text{cond}} = \sum_{j=1}^B w_{ij} (a_{j,d}^* - a_{i,d}^*)^2$ and the global reference variance $v_d^{\text{ref}} = \frac{1}{B} \sum_{i=1}^B (a_{i,d}^* - \bar{a}_d^*)^2$. Both statistics are computed within the current minibatch: the local second moment is centered at a_i^* , and the reference variance is centered at the batch mean. To eliminate dimensional impacts and compare the tightness of different dimensions on the same relative baseline, we convert them into normalized inverse-standard-deviation scales:

$$s_{i,d}^{\text{cond}} = \frac{(v_{i,d}^{\text{cond}} + \varepsilon)^{-1/2}}{\mathbb{E}_k[(v_{i,k}^{\text{cond}} + \varepsilon)^{-1/2}]}, \quad s_d^{\text{ref}} = \frac{(v_d^{\text{ref}} + \varepsilon)^{-1/2}}{\mathbb{E}_k[(v_k^{\text{ref}} + \varepsilon)^{-1/2}]}. \quad (9)$$

Table 1: **Performance on 3D pointcloud manipulation.** We assess performance on 56 challenging tasks with 3 random seeds. The aggregate standard deviation averages the per-task standard deviations across seeds.

Methods	NFE	Adroit			DexArt				MetaWorld				Avg.
		Hammer	Door	Pen	Laptop	Faucet	Bucket	Toilet	Easy (28)	Medium (11)	Hard (5)	Very Hard (5)	
Multi-Step:													
DP3†	10	100±0	62±4	43±6	83±1	63±2	46±2	82±4	90.9±1.4	61.6±6.5	38.0±4.2	49.0±6.8	73.9±3.3
Flow	10	100±0	72±3	69±3	92±3	37±5	34±5	80±7	66.7±7.7	41.3±7.4	40.3±7.2	46.7±7.6	57.9±7.0
DP3*	10	100±0	58±5	53±3	83±2	40±2	25±1	74±2	91.5±0.9	74.5±5.7	54.2±1.8	71.8±4.0	79.4±2.4
Flow*	10	99±1	69±3	45±3	85±0	42±4	26±3	74±3	92.2±0.7	74.6±5.0	55.2±3.0	70.8±3.0	79.9±2.2
Flow-Map / Shortcut / Few-Step:													
CP	1	98±3	67±5	65±5	87±3	44±4	29±4	82±4	68.3±1.8	32.1±4.3	28.7±5.2	36.0±2.3	54.7±2.9
MP1	1	100±0	74±3	72±3	88±5	42±1	33±0	82±2	70.5±6.1	37.9±5.9	34.3±9.5	32.3±5.5	57.4±5.8
OneDP	1	99±1	67±2	65±2	89±4	45±3	33±6	83±4	77.7±0.5	39.6±9.2	38.7±4.4	41.7±1.6	62.4±3.0
OFP	1	100±0	79±7	76±7	92±3	46±4	39±0	81±3	87.9±4.6	52.4±5.7	43.3±2.4	49.2±0.7	71.6±4.1
CP*	1	97±2	66±0	54±2	76±11	38±0	25±5	73±4	79.8±14	59.4±12	41.4±14	62.0±8.2	68.4±11.9
MP1*	1	99±1	48±2	58±4	82±3	46±2	26±3	69±3	91.3±1.2	75.4±5.6	57.6±4.6	72.8±4.2	79.7±2.8
FlowPolicy*	1	98±2	66±1	55±2	86±2	40±1	28±0	76±3	90.4±0.8	67.0±6.2	51.2±4.8	67.0±3.0	76.9±2.5
Drifting Methods:													
Naive Drifting*	1	100±0	67±3	43±5	95±4	46±3	29±1	82±6	92.7±1.8	70.7±6.5	54.2±5.8	73.2±5.2	79.8±3.5
Ours*	1	100±0	67±2	52±4	96±3	48±1	30±0	83±6	93.2±0.8	73.6±3.9	56.4±4.2	74.2±1.4	81.3±2.0

* Results are reproduced on a single NVIDIA A40. † Results are cited from DP3. All other baselines cited from OFP.

Since we only wish to penalize dimensions that become more strictly constrained due to the current observation than they usually are, we consider utilizing the filtering property of the one-sided ReLU operator to define the Geometry Excess (Fig 2 (c)):

$$m_{i,d} = \text{ReLU} \left[\frac{s_{i,d}^{\text{cond}}}{s_d^{\text{ref}} + \varepsilon} - 1 \right], \quad M_i = \text{Diag}(m_i). \quad (10)$$

The ReLU retains only the part of the normalized local-to-reference ratio exceeding one, leaving a baseline coefficient of one in the other coordinates. For action chunks, $m_{i,\ell,d}$ is retained separately for each time–action coordinate pair. Consequently, we filter out inherent features that do not surpass the baseline, extract the specific geometry associated with the current state, and use it as the adaptive weight matrix M_i for the potential in Section 4.1.

5 Experiments

We design our experiments to examine four questions that follow the construction of IDP:

- **H1: Performance.** Can IDP improve one-step action generation across simulated and real manipulation tasks, while narrowing the performance gap to multi-step generative policies?
- **H2: Implicit Correction and Local-Aware Evaluation.** Can IDP benefit from local geometric supervision by being evaluated near expert actions during training?
- **H3: Conditional Expert Geometry.** Do expert actions under similar observations provide useful local geometry for guiding action correction?
- **H4: Local Geometry Excess.** Does comparing conditional geometry with reference geometry isolate condition-specific constraints from marginal action variation?

5.1 Experimental Setup

Tasks. We evaluate IDP on three settings: (i) 2D state- and image-based benchmarks following the MIP protocol [25] on Robomimic (Lift, Can, Square, Transport, with both ph and mh), Tool-Hang, and PushT, (ii) 3D pointcloud-conditioned manipulation following OFP [20] and FlowPolicy [34] on 56 tasks across Adroit [28], DexArt [1], and MetaWorld [32], and (iii) a real-world “Pick Peach” task on a single-arm ALOHA platform with a strictly bimodal demonstration distribution.

Baselines. We compare against multi-step generative policies (DP [5], DP3 [33], Flow [23]), few-step / one-step accelerated policies (CP [27], MIP [25], MP1 [30], OneDP [31], OFP [20], FlowPolicy [34]), and the explicit drifting baseline (Naive Drifting [7]).

Table 2: Performance on 2D **state-based** robotics manipulation tasks. We report the best success rate / the mean over the last five evaluations during training. Each result is averaged over two network architectures (Transformer and CNN), three training seeds, and 50 environment initial conditions.

Methods	NFE	Lift		Can		Square		Transport		Tool-Hang	PushT	Avg.
		mh	ph	mh	ph	mh	ph	mh	ph			
Multi-Step:												
DP	100	1.00/0.99	1.00/1.00	0.98/0.92	1.00/0.98	0.81/0.72	0.96/0.92	0.45/0.34	0.73/0.62	0.84/0.71	0.95/0.92	0.87/0.81
Flow	9	1.00/0.99	1.00/1.00	0.96/0.92	1.00/0.98	0.76/0.65	0.95/0.90	0.33/0.23	0.74/0.59	0.70/0.57	0.96/0.94	0.84/0.78
Few-Step / Flow-Map / Shortcut:												
MIP	2	1.00/0.99	1.00/1.00	0.96/0.94	1.00/0.98	0.80/0.69	0.96/0.88	0.44/0.35	0.71/0.58	0.68/0.55	0.96/0.94	0.85/0.79
CP	1	0.99/0.97	1.00/1.00	0.83/0.74	0.99/0.96	0.49/0.45	0.88/0.82	0.01/0.01	0.32/0.24	0.43/0.33	0.94/0.91	0.69/0.64
MP1	1	1.00/0.99	1.00/1.00	0.96/0.87	1.00/0.97	0.77/0.62	0.96/0.88	0.24/0.14	0.52/0.39	0.76/0.63	0.95/0.92	0.82/0.74
Drifting Methods:												
Naive Drifting	1	1.00/0.99	1.00/1.00	0.96/0.91	0.99/0.96	0.70/0.60	0.97/0.90	0.30/0.22	0.68/0.58	0.53/0.42	0.95/0.91	0.81/0.75
Ours	1	1.00/0.99	1.00/1.00	0.98/0.94	1.00/0.97	0.82/0.68	0.95/0.88	0.41/0.29	0.72/0.58	0.81/0.60	0.97/0.92	0.87/0.79

Table 3: Performance on 2D **image-based** robotics manipulation tasks. We report the best success rate / the mean over the last five evaluations during training. Each result is averaged over two network architectures (Transformer and CNN), three training seeds, and 50 environment initial conditions.

Methods	NFE	Lift		Can		Square		Tool-Hang	PushT	Avg.
		mh	ph	mh	ph	mh	ph			
<i>Multi-Step:</i>										
DP	100	1.00/0.98	1.00/0.99	0.90/0.82	0.98/0.94	0.87/0.76	0.97/0.91	0.48/0.33	0.93/0.88	0.89/0.83
Flow	9	1.00/0.98	1.00/0.99	0.93/0.83	0.99/0.94	0.86/0.78	0.98/0.90	0.47/0.35	0.92/0.86	0.89/0.83
<i>Few-Step / Flow-Map / Shortcut:</i>										
MIP	2	1.00/0.96	1.00/0.99	0.89/0.83	0.97/0.93	0.82/0.71	0.95/0.83	0.46/0.28	0.87/0.84	0.87/0.80
CP	1	1.00/0.98	1.00/1.00	0.77/0.63	0.80/0.60	0.70/0.59	0.92/0.86	0.22/0.12	0.88/0.83	0.79/0.70
MP1	1	1.00/0.97	1.00/0.99	0.87/0.77	0.77/0.68	0.85/0.78	0.96/0.90	0.43/0.27	0.86/0.79	0.84/0.77
<i>Drifting Methods:</i>										
Naive Drifting	1	0.99/0.96	1.00/0.98	0.90/0.79	0.95/0.84	0.80/0.69	0.92/0.84	0.46/0.26	0.91/ 0.86	0.87/0.78
Ours	1	1.00/0.97	1.00/0.99	0.95/0.88	0.99/0.94	0.82/0.74	0.94/0.86	0.48/0.33	0.92/0.86	0.89/0.82

Metrics. We report success rates averaged across 2 architectures (Transformer / CNN) and 3 seeds for 2D tasks, and across 3 seeds for 3D tasks. Detailed task descriptions, baseline reproduction settings, hyperparameters, and compute resources are deferred to Appendix C–D.

5.2 One-Step Policy Performance (H1)

Simulated Manipulation. To address **H1**, we evaluate the control performance of IDP across the 2D and 3D simulated benchmarks. As detailed in Tables 2, 3, and 1, IDP achieves highly competitive success rates across state, image, and pointcloud modalities. Compared to the explicit Naive Drifting baseline, IDP generally yields more stable and improved performance, suggesting that the implicit geometric objective can serve as a more reliable correction mechanism than empirical vector field estimation. When assessed against other leading single-step methods (e.g., CP, MP1, and OFP), IDP demonstrates solid performance, matching or slightly exceeding their success rates on the majority of tasks. Furthermore, operating purely at NFE = 1, IDP performs closely to the multi-step generative references (e.g., 100-step DP and 10-step DP3) in several settings. These results indicate that incorporating implicit geometry effectively aids in synthesizing high-quality actions without necessarily relying on iterative inference steps.

Real-World Manipulation. We further validate IDP on a single-arm ALOHA platform on a “Pick Peach” task with a strictly bimodal demonstration distribution (50% left, 50% right). Each method is evaluated over 10 trials (5 per mode) under both state-based (qpos) and vision-based modalities. The full hardware and protocol are deferred to Appendix C.3. In the vision-based setting (Table 4), IDP achieves 50% success while DP, Naive Drifting, and MIP all fail at 0%. Qualitatively, the failing baselines exhibit a consistent *deviation from the action manifold*: they grasp at locations noticeably *behind* the target peach, and multi-step DP additionally outputs invalid intermediate gripper commands (e.g., freezing at 0.769) while breaking the high-frequency control loop with its sluggish inference. IDP adheres to valid grasping poses with real-time reactivity, supporting our geometric hypothesis. In the state-based setting, where joint-angle proximity is a poor proxy for task context, IDP-qpos (30%) underperforms MIP-qpos (40%). We discuss this regime in Appendix F.

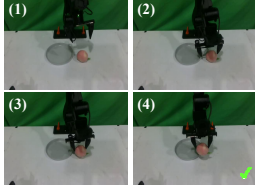


Figure 3: Real-world “Pick Peach” task visualization.

Table 4: Success rates on the real-world “Pick Peach” task.

Modality	Method	SR (%)
State-based (qpos)	DP	10
	MIP	40
	Naive Drifting	0
	Ours	30
Vision-based (image)	DP	0
	MIP	0
	Naive Drifting	0
	Ours	50

Table 5: Ablation results on ToolHang-State (Max / Avg).

Method	ToolHang-State
Naive Drifting	53.2 / 42.4
IDP (Ours)	81.0 / 60.0
w/o terminal eval	40.0 / 25.5
uniform neighbors	65.0 / 59.5
shuffled neighbors	75.0 / 52.0
w/o reference geometry	78.0 / 54.5
absolute excess	75.0 / 57.0

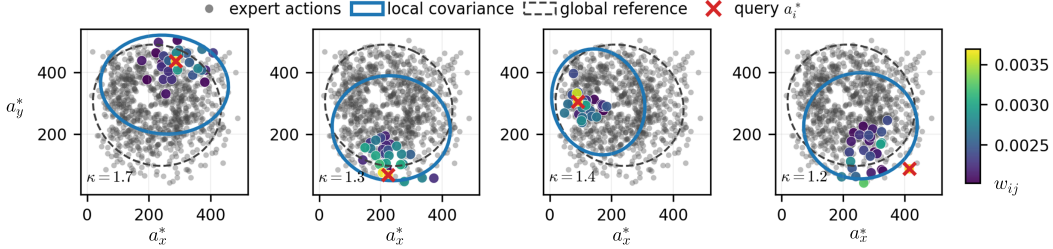


Figure 4: Observation-conditioned expert-action scatter on PushT-State. Colored points denote weighted neighbors. Blue and gray ellipses illustrate local and global action scatter.

All evaluated methods, including IDP, MIP, and 100-step DP, learn a single placement mode under perfectly balanced bimodal demonstrations. Multi-step DP also fails to complete the task in this setting. IDP retains action-validity within the learned mode. Mode coverage remains an open problem we discuss in Appendix F.

5.3 Ablation Study (H2 ~ H4)

We ablate the core designs of IDP to address hypotheses **H2** ~ **H4**. As our geometric objective primarily benefits tasks with narrow successful action regions, we conduct quantitative ablations on ToolHang-State (Table 5).

Expert-proximal evaluation (H2): Removing the expert-proximal probe collapses the success rates on ToolHang-State from 81.0%/60.0% to 40.0%/25.5%, well below Naive Drifting (53.2%/42.4%). The probe is therefore essential for internalizing the local correction structure into the one-step generator. **Conditional expert geometry (H3):** Replacing observation-conditioned neighborhood weights with uniform or shuffled ones degrades the success rates to 65.0%/59.5% and 75.0%/52.0% respectively, confirming that the geometry must be condition-specific. Fig. 4 visualizes this construction on PushT-State: observation-similar demonstrations form local covariances that deviate from the global reference with moderate anisotropy ($\kappa \approx 1.2$ – 1.7). Further component ablations are reported in Appendix E.2. **Reference geometry and local excess (H4):** Removing the reference geometry reduces success from 81.0%/60.0% to 78.0%/54.5%, while absolute excess gives 75.0%/57.0%. The full construction outperforms both variants, supporting the reference comparison and positive relative excess for geometry-aware supervision.

6 Conclusion

This paper addresses the loss of intermediate error correction in one-step action generation by proposing the Implicit Drifting Policy (IDP). IDP extracts local geometric constraints directly from expert actions under similar observations. By combining this conditional expert geometry with an expert-proximal training evaluation, IDP structurally internalizes the local manifold topology into a single-step generator. Extensive evaluations across 2D, 3D, and real-world manipulation benchmarks demonstrate that IDP effectively adheres to valid action manifolds, substantially improving upon explicit drifting methods while achieving superior geometric precision against strong one-step baselines. Appendix F discusses the limitations of IDP. Improving demonstration coverage and extending the policy to multi-modal action generation are directions for future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Project Number 62595774), MoE Key Laboratory of Intelligent Perception and Human-Machine Collaboration (KLIP-HuMaCo), HPC Platform of ShanghaiTech University.

References

- [1] Chen Bao, Helin Xu, Yuzhe Qin, and Xiaolong Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21190–21200, 2023.
- [2] Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025. URL <https://arxiv.org/abs/2505.18825>.
- [3] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.
- [4] Haojun Chen, Minghao Liu, Chengdong Ma, Xiaojian Ma, Zailin Ma, Huimin Wu, Yuanpei Chen, Yifan Zhong, Mingzhi Wang, Qing Li, and Yaodong Yang. Falcon: Fast visuomotor policies via partial denoising, 2025. URL <https://arxiv.org/abs/2503.00339>.
- [5] 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, 2024. URL <https://arxiv.org/abs/2303.04137>.
- [6] Eugenio Chisari, Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Learning robotic manipulation policies from point clouds with conditional flow matching, 2024. URL <https://arxiv.org/abs/2409.07343>.
- [7] Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting, 2026. URL <https://arxiv.org/abs/2602.04770>.
- [8] Yufei Duan, Hang Yin, and Danica Kragic. Real-time iteration scheme for diffusion policy, 2025. URL <https://arxiv.org/abs/2508.05396>.
- [9] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning, 2021. URL <https://arxiv.org/abs/2109.00137>.
- [10] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models, 2025. URL <https://arxiv.org/abs/2410.12557>.
- [11] Niklas Funk, Julen Urain, Joao Carvalho, Vignesh Prasad, Georgia Chalvatzaki, and Jan Peters. Actionflow: Equivariant, accurate, and efficient policies with spatially symmetric flow matching, 2024. URL <https://arxiv.org/abs/2409.04576>.
- [12] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL <https://arxiv.org/abs/2505.13447>.
- [13] Sigmund H. Høeg, Yilun Du, and Olav Egeland. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models, 2024. URL <https://arxiv.org/abs/2406.04806>.
- [14] Yanlong Huang, Leonel Roza, João Silvério, and Darwin G. Caldwell. Kernelized movement primitives, 2018. URL <https://arxiv.org/abs/1708.08638>.
- [15] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

- [16] Jindou Jia, Gen Li, Xiangyu Chen, Tuo An, Yuxuan Hu, Jingliang Li, Xinying Guo, and Jianfei Yang. Action-to-action flow matching, 2026. URL <https://arxiv.org/abs/2602.07322>.
- [17] S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.
- [18] Chieh-Hsin Lai, Bac Nguyen, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, Stefano Ermon, and Molei Tao. A unified view of drifting and score-based models, 2026. URL <https://arxiv.org/abs/2603.07514>.
- [19] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions, 2024. URL <https://arxiv.org/abs/2403.03181>.
- [20] Shaolong Li, Lichao Sun, and Yongchao Chen. One-step flow policy: Self-distillation for fast visuomotor policies, 2026. URL <https://arxiv.org/abs/2603.12480>.
- [21] Yiming Li, Nael Darwiche, Amirreza Razmjoo, Sichao Liu, Yilun Du, Auke Ijspeert, and Sylvain Calinon. Geometry-aware policy imitation, 2025. URL <https://arxiv.org/abs/2510.08787>.
- [22] Zhiqi Li and Bo Zhu. A long-short flow-map perspective for drifting models, 2026. URL <https://arxiv.org/abs/2602.20463>.
- [23] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL <https://arxiv.org/abs/2210.02747>.
- [24] Guanxing Lu, Zifeng Gao, Tianxing Chen, Wenxun Dai, Ziwei Wang, Wenbo Ding, and Yansong Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation, 2025. URL <https://arxiv.org/abs/2406.01586>.
- [25] Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much ado about noising: Dispelling the myths of generative robotic control, 2026. URL <https://arxiv.org/abs/2512.01809>.
- [26] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems*, volume 26, 2013.
- [27] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation, 2024. URL <https://arxiv.org/abs/2405.07503>.
- [28] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems (RSS)*, 2018.
- [29] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation, 2025. URL <https://arxiv.org/abs/2506.14603>.
- [30] Juyi Sheng, Ziyi Wang, Peiming Li, and Mengyuan Liu. Mp1: Meanflow tames policy learning in 1-step for robotic manipulation, 2025. URL <https://arxiv.org/abs/2507.10543>.
- [31] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation, 2024. URL <https://arxiv.org/abs/2410.21257>.
- [32] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2020.
- [33] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Robotics: Science and Systems (RSS)*, 2024. URL <https://arxiv.org/abs/2403.03954>.

- [34] Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation, 2024. URL <https://arxiv.org/abs/2412.04987>.

Appendix

A Proofs and Derivations

A.1 Proof of Proposition 3.1 (Degeneration of Explicit Drifting in Behavior Cloning)

Proof. We compute the empirical conditional drifting field explicitly under the Dirac assumptions and substitute it into the field-displaced regression objective.

We first restate the conditional drifting field. Conditioning on o_i , the attraction and repulsion components are

$$V_{\hat{p}}^+(a \mid o_i) = \frac{\mathbb{E}_{y \sim \hat{p}_{\text{data}}(\cdot \mid o_i)}[k(a, y)(y - a)]}{\mathbb{E}_{y \sim \hat{p}_{\text{data}}(\cdot \mid o_i)}[k(a, y)]}, \quad V_{\hat{q}}^-(a \mid o_i) = \frac{\mathbb{E}_{y \sim \hat{q}_{\theta}(\cdot \mid o_i)}[k(a, y)(y - a)]}{\mathbb{E}_{y \sim \hat{q}_{\theta}(\cdot \mid o_i)}[k(a, y)]}, \quad (11)$$

where the empirical conditional distributions are $\hat{p}_{\text{data}}(y \mid o_i) = \delta(y - a_i^*)$ and $\hat{q}_{\theta}(y \mid o_i) = \delta(y - a)$ with $a = f_{\theta}(o_i, a_0, 0)$.

We next evaluate the attraction $V_{\hat{p}}^+$. By the sifting property of the Dirac measure,

$$\mathbb{E}_{y \sim \hat{p}_{\text{data}}(\cdot \mid o_i)}[k(a, y)(y - a)] = \int k(a, y)(y - a) \delta(y - a_i^*) dy = k(a, a_i^*)(a_i^* - a), \quad (12)$$

$$\mathbb{E}_{y \sim \hat{p}_{\text{data}}(\cdot \mid o_i)}[k(a, y)] = \int k(a, y) \delta(y - a_i^*) dy = k(a, a_i^*). \quad (13)$$

Since k is positive ($k(a, a_i^*) > 0$ for all a , by assumption), the ratio is well-defined and the kernel cancels out exactly:

$$V_{\hat{p}}^+(a \mid o_i) = \frac{k(a, a_i^*)(a_i^* - a)}{k(a, a_i^*)} = a_i^* - a. \quad (14)$$

The kernel thus contributes nothing to the conditional attraction direction once the support of $\hat{p}_{\text{data}}(\cdot \mid o_i)$ collapses onto a single point.

Turning to the repulsion $V_{\hat{q}}^-$, the term reduces to a self-evaluation since $\hat{q}_{\theta}(\cdot \mid o_i)$ is concentrated on the prediction a :

$$\mathbb{E}_{y \sim \hat{q}_{\theta}(\cdot \mid o_i)}[k(a, y)(y - a)] = \int k(a, y)(y - a) \delta(y - a) dy = k(a, a)(a - a) = 0, \quad (15)$$

$$\mathbb{E}_{y \sim \hat{q}_{\theta}(\cdot \mid o_i)}[k(a, y)] = \int k(a, y) \delta(y - a) dy = k(a, a). \quad (16)$$

Positivity gives $k(a, a) > 0$, so the self-evaluated repulsion vanishes:

$$V_{\hat{q}}^-(a \mid o_i) = \frac{0}{k(a, a)} = 0. \quad (17)$$

Composing the two components,

$$V_{\hat{p}, \hat{q}}(a \mid o_i) = V_{\hat{p}}^+(a \mid o_i) - V_{\hat{q}}^-(a \mid o_i) = (a_i^* - a) - 0 = a_i^* - a. \quad (18)$$

At the current prediction, the singleton empirical estimate is the expert–prediction difference, independently of the positive kernel. The empirical support is fixed when evaluating this expression.

It remains to substitute this drifting field into the field-displaced regression objective. With $a = f_{\theta}(o_i, a_0, 0)$,

$$\mathcal{L}_{\text{Drift}}(\theta) = \mathbb{E}_{a_0 \sim p_0} \|f_{\theta}(o_i, a_0, 0) - \text{sg}[f_{\theta}(o_i, a_0, 0) + V_{\hat{p}, \hat{q}}(f_{\theta}(o_i, a_0, 0) \mid o_i)]\|_2^2 \quad (19)$$

$$= \mathbb{E}_{a_0 \sim p_0} \|a - \text{sg}[a + (a_i^* - a)]\|_2^2 \quad (20)$$

$$= \mathbb{E}_{a_0 \sim p_0} \|a - \text{sg}[a_i^*]\|_2^2. \quad (21)$$

Since a_i^* is a fixed dataset target and not a function of θ , the stop-gradient operator acts as the identity on it, yielding

$$\mathcal{L}_{\text{Drift}}(\theta) = \mathbb{E}_{a_0 \sim p_0} \|f_{\theta}(o_i, a_0, 0) - a_i^*\|_2^2. \quad (22)$$

□

A.2 Derivation of the Geometry-Induced Potential

Proof. For fixed symmetric $M_i \succeq 0$, let $e = a - a_i^*$. Differentiating the quadratic potential gives

$$\nabla_a E_i^{\text{geo}}(a) = \frac{1}{2}(M_i + M_i^\top)e = M_i e. \quad (23)$$

Its negative gradient is therefore $M_i(a_i^* - a)$, the geometric correction in Eq. (3). Adding the isotropic term gives $-\nabla_a E_i(a) = (I + M_i)(a_i^* - a)$. Since $I + M_i \succ 0$, the expert action is the unique minimum of the complete potential as a function of the action variable with geometry fixed. $\square$

A.3 Interpretation of Conditional Expert Geometry

For fixed nonnegative weights with $\sum_j w_{ij} = 1$, let $\mu_i^w = \sum_j w_{ij} a_j^*$. Expanding Eq. (8) about this weighted mean cancels the cross terms:

$$G_i = \sum_j w_{ij} (a_j^* - \mu_i^w)(a_j^* - \mu_i^w)^\top + (\mu_i^w - a_i^*)(\mu_i^w - a_i^*)^\top. \quad (24)$$

The first term measures how related expert actions vary around their weighted mean. The second measures the offset of that mean from the current expert target a_i^* . Together, they describe action variation around the target used for supervision.

A.4 General Multivariate Excess and Diagonal Instantiation

In Sec. 4.3 of the main text we instantiated the geometry excess via a coordinate-wise comparison of normalized inverse standard deviations and noted, in passing, the existence of a more general multivariate form. Here we describe this conceptual extension and distinguish it from the implemented diagonal weights.

Full multivariate excess. Let $P_i^{\text{cond}} := (G_i + \varepsilon I)^{-1}$ and $P^{\text{ref}} := (\Sigma_{\text{ref}} + \varepsilon I)^{-1}$ be the regularized conditional and reference precision matrices. To compare them on a common scale, we whiten P_i^{cond} against P^{ref} :

$$R_i := (P^{\text{ref}})^{-1/2} P_i^{\text{cond}} (P^{\text{ref}})^{-1/2}. \quad (25)$$

Eigenvalues $\lambda_d(R_i)$ of R_i measure how much tighter the conditional precision is along each generalized direction relative to the reference. The natural matrix-valued positive excess is then

$$M_i^{\text{full}} = U_i \text{Diag}[(\lambda_d(R_i) - 1)_+] U_i^\top, \quad (26)$$

where U_i is the orthogonal matrix of eigenvectors of R_i and $(\cdot)_+ = \max(\cdot, 0)$.

Diagonal construction. Computing M_i^{full} requires a generalized eigenvalue decomposition (GEVD) of P_i^{cond} and P^{ref} at a cost of $\mathcal{O}(d_a^3)$ per training sample, which is prohibitive in the inner loop of policy training. Under the diagonal restriction $G_i \approx \text{Diag}(v_i^{\text{cond}})$ and $\Sigma_{\text{ref}} \approx \text{Diag}(v^{\text{ref}})$, both precision matrices become diagonal, $U_i = I$, and the eigenvalue ratio reduces to the per-coordinate inverse-variance ratio:

$$\lambda_d(R_i) = \frac{1/(v_{i,d}^{\text{cond}} + \varepsilon)}{1/(v_d^{\text{ref}} + \varepsilon)} = \frac{v_d^{\text{ref}} + \varepsilon}{v_{i,d}^{\text{cond}} + \varepsilon}. \quad (27)$$

For the implemented diagonal weights, we normalize the conditional and reference inverse standard deviations across coordinates and retain their positive relative excess, as defined in Eqs. (9)–(10).

Why the diagonal approximation is justified here:

- *Computational cost.* The cost drops from $\mathcal{O}(d_a^3)$ to $\mathcal{O}(d_a)$ per sample, eliminating GEVD from the training inner loop entirely.
- *Use case for M_i .* The metric M_i is consumed only via the contraction $r^\top M_i r$ for $r = a - a_i^*$. Coordinate-wise penalties along physically meaningful action axes (joint angles, end-effector velocities, gripper command) already constitute a strong inductive bias.

Each action-chunk entry has its own weight $m_{i,\ell,d}$.

B Algorithms

Algorithm 1 summarizes IDP training. Geometry is computed from the current minibatch and held fixed during backpropagation. The policy parameters, including the observation encoder, are optimized using Eq. (6).

Algorithm 1 IDP Training

Require: Demonstrations $\mathcal{D}$, policy f_θ , t_* , λ_{prox}

- 1: **while** training **do**
- 2: Sample a minibatch $\mathcal{B}$
- 3: Compute h_i using Eq. (7)
- 4: Compute row-standardized similarities and weights w_{ij}
- 5: Compute local and reference action statistics
- 6: Compute M_i using Eqs. (9)–(10)
- 7: Sample $\epsilon_i \sim \mathcal{N}(0, I)$ and set $\tilde{a}_i = a_i^* + (1 - t_*)\epsilon_i$
- 8: $y_i \leftarrow f_\theta(o_i, 0, 0)$ and $z_i \leftarrow f_\theta(o_i, \tilde{a}_i, t_*)$
- 9: Evaluate $\mathcal{L}$ using Eq. (6)
- 10: Update θ
- 11: **end while**
- 12: **Deployment:** $f_\theta(o, 0, 0)$

All batch samples participate in softmax, including the current sample i . We evaluate two ways of removing it with fixed policy parameters on ToolHang-State and Box-Close. Post-softmax removal renormalizes the remaining weights. Pre-standardization removal excludes the sample before computing row statistics and softmax. Table 6 reports the relative L_2 changes in geometry.

Table 6: Self-inclusion diagnostics. The last two columns report relative L_2 changes, summarized using the statistic shown for each task.

Task	Statistic	Mean self weight (%)	Post-softmax removal	Pre-standardization removal
ToolHang-State	Median	0.52	9.2×10^{-6}	1.75×10^{-3}
Box-Close	Mean	2.09	2.7×10^{-6}	5.0×10^{-3}

C Detailed Experimental Setup

C.1 2D Benchmarks

We follow the protocol of MIP [25] for all 2D experiments. The benchmark suite consists of six continuous-control manipulation tasks:

- **Robomimic Lift / Can / Square / Transport:** four manipulation primitives ranging from picking-and-lifting (Lift, easy) to two-arm transport (hardest). Each task is evaluated on ph (proficient-human, 200 expert demonstrations) and mh (multi-human, 300 demonstrations from operators of varying skill).
- **Tool-Hang:** a long-horizon precision task where a hook tool must be inserted onto a wall mount. Notable for its narrow valid grasping pose region, which makes it a strong testbed for IDP’s local geometric correction.
- **PushT:** a 2D pushing task with an underactuated dynamic shape. The action manifold is broad, making the task relatively insensitive to anisotropic geometric constraints. Its action scatter is visualized in Fig. 4.

For the task subsets reported in Tables 2 and 3, we evaluate under the corresponding observation modalities: (i) *state-based*, where the observation is the low-dimensional state of the simulator, and (ii) *image-based*, where the observation consists of RGB rendering(s). We use the same dataset splits, action chunking horizons, and replan strides as MIP. Evaluation is run for 50 environment

initial conditions per seed, repeated over 3 random seeds, and averaged across two architectures (Transformer and CNN) for both observation modalities.

C.2 3D Benchmarks

For 3D pointcloud-conditioned manipulation we follow OFP [20] and FlowPolicy [34]. The suite contains 56 tasks organized as:

- **Adroit** [28] (3 tasks): Hammer, Door, Pen. A 26-DoF dexterous hand-arm, requiring high-precision contact-rich manipulation.
- **DexArt** [1] (4 tasks): Laptop, Faucet, Bucket, Toilet. Articulated-object manipulation with diverse joint configurations.
- **MetaWorld** [32] (49 tasks): a broad set of parallel-gripper tasks. Following OFP, we partition the 49 tasks into difficulty levels (28 Easy / 11 Medium / 5 Hard / 5 Very-Hard) for analysis.

Observations are point clouds of the workspace (1024 points per frame, sampled following DP3 [33]). All training and evaluation hyperparameters that are not specific to IDP follow OFP / FlowPolicy unchanged. Each method is averaged over 3 random seeds, and per-task standard deviations are reported in Table 11.

C.3 Real-world “Pick Peach” Setup

Hardware. We use a single-arm ALOHA platform equipped with a parallel-jaw gripper. Two RGB cameras provide visual feedback: one front-view and one head-mounted (egocentric).

Task. The robot must pick up a peach placed at one of two designated positions and transport it onto a target plate. The task simultaneously tests spatial generalization (peach position varies within each side) and multi-modal modeling (left vs. right placement).

Demonstrations. We collect teleoperated demonstrations under a strictly bimodal placement distribution: 50% of trajectories place the peach on the left side of the plate and 50% on the right. The two modes are exactly balanced, providing a clean test of whether single-step policies can preserve multi-modality without iterative stochastic injection.

Evaluation. Each method is evaluated for 10 trials, 5 from each mode (left / right). A trial is counted as successful only if the peach is grasped, lifted, transported, and released onto the plate. We test under two observation modalities: state-based (qpos only, no visual feedback) and vision-based (both RGB streams). Results are summarized in Table 4 of the main text.

C.4 Baseline Reproduction Details

For 2D state- and image-based experiments, the baselines DP, MIP, MP1, and Flow follow MIP’s protocol and code base unchanged. For 3D experiments, the multi-step DP3 / Flow baselines and the few-step CP / MP1 / FlowPolicy baselines are reproduced from the OFP repository on a single NVIDIA A40 to ensure fair compute comparison. Results marked with * in Table 1 (and Tab. 11) indicate this re-running protocol. Numbers without * are cited from the corresponding original papers (OFP / FlowPolicy / DP3) when reproduction was infeasible due to compute budget.

Naive Drifting estimates the empirical field from expert and generated samples and regresses to the resulting frozen target. IDP w/o geometry retains both training inputs and sets $M_i = 0$. The deployment-call counts are reported in the result tables.

D Implementation Details and Hyperparameters

D.1 Architecture

Policy backbone. For 2D experiments we adopt the Transformer and CNN backbones from MIP [25] unchanged. Each task is run with both architectures and the success rates are then averaged.

For 3D experiments we use the DP3 PointNet-based encoder [33] for the point-cloud observation and a Transformer policy network, identical to the setup used by OFP [20] and FlowPolicy [34].

Observation embedding ϕ_θ . The embedding $\phi_\theta(o_i)$ used to compute the neighborhood weights w_{ij} is read from the final hidden representation of the observation encoder, prior to any task-specific head. Stop-gradient (detach) and L_2 -normalization are applied to $\phi_\theta(o_i)$ before computing pairwise inner products, so that the geometry statistics are fixed during the current backward pass (Appendix B).

D.2 Hyperparameters

Table 7 summarizes the training and geometry settings. Other quantities (learning rate, batch size, training steps, action-chunk horizon, and evaluation protocol) retain the original benchmark configurations.

Table 7: Training and geometry settings. Values are fixed across the evaluated settings.

Symbol	Value	Description
t_*	0.9	Schedule scalar for the expert-proximal input.
λ_{prox}	$(t_*/(1-t_*))^2 = 81$	Proximal loss weight (Eq. (6)).
Probe noise ε_i	$\mathcal{N}(0, I)$	Scaled by $(1-t_*)$ in $\tilde{a}_i = a_i^* + (1-t_*)\varepsilon_i$.
Neighborhood scope	full minibatch	w_{ij} over all $j \in \{1, \dots, B\}$ including $j = i$.
Softmax temperature	1	Applied directly on row z-scored similarities (no extra temperature).
Numerical ε	10^{-6}	Used in L_2 -normalization, variance flooring, and ratio denominators.
v_d^{ref} estimation	per-minibatch	$v_d^{\text{ref}} = \frac{1}{B} \sum_i (a_{i,d}^* - \bar{a}_d^*)^2$, no EMA / no precompute.
Stop-gradient on w_{ij}, M_i	yes	Geometry path does not flow gradient into θ .

Implementation note on λ_{prox} . We fix $t_* = 0.9$ and $\lambda_{\text{prox}} = [t_*/(1-t_*)]^2 = 81$ across all 2D, 3D, and real-world experiments, without per-task tuning. The training inputs, relative time weighting, and 2D experimental configuration follow [25].

D.3 Compute Resources

The main benchmark experiments used an NVIDIA A40 GPU. Geometry computation consists of batch feature similarities, softmax weights, and coordinate-wise action statistics. The corresponding costs are $O(B^2 d_h)$ and $O(B^2 H d_a)$. We measure training step time, peak memory, and inference latency on the same machine with the same architecture and batch size. Endpoint regression uses the direct training input. IDP w/o geometry retains both training inputs and sets $M_i = 0$. Inference latency is measured with batch size one.

Table 8: Training and inference costs. NFE is the number of policy evaluations at deployment.

Input	Method	Step time (ms)	Peak memory (GiB)	NFE	Latency (ms)
State	Endpoint regression	78.5 ± 4.6	2.38	1	8.67
	IDP w/o geometry	131.4 ± 6.5	4.39	1	8.79
	IDP	130.7 ± 4.1	4.39	1	8.64
	MIP	119.7 ± 0.7	4.39	2	13.73
Image	Endpoint regression	254.0 ± 0.3	13.32	1	12.19
	IDP w/o geometry	312.9 ± 1.4	15.33	1	12.18
	IDP	313.1 ± 1.3	15.33	1	12.35
	MIP	303.5 ± 0.3	15.33	2	17.21

IDP and IDP w/o geometry have similar training step times and the same peak memory, while IDP retains single-step deployment.

E Detailed Experimental Results

E.1 Per-Task Results

Tables 9 and 10 report the detailed 2D results for each architecture. Table 11 reports the 3D task results, and Table 12 reports expert success rates.

Table 9: Detailed 2D simulation results on **state-based** robotic manipulation tasks. For each task, we report the best success rate / the mean over the last five evaluations during training. Results are averaged over three training seeds.

Architecture	Method	Lift		Can		Square		Transport		Tool-Hang	Push-T
		mh	ph	mh	ph	mh	ph	mh	ph		
Transformer	DP	1.00/0.99	1.00/1.00	0.97/0.89	1.00/0.97	0.74/0.67	0.95/0.90	0.37/0.25	0.64/0.54	0.77/ 0.68	0.92/0.88
	Flow	1.00/1.00	1.00/1.00	0.92/0.87	1.00/0.98	0.69/0.53	0.94/0.88	0.23/0.14	0.71/0.56	0.58/0.46	0.94/ 0.92
	MIP	1.00/0.99	1.00/1.00	0.94/0.91	1.00/0.98	0.77/0.69	0.98/0.91	0.40/0.26	0.69/0.54	0.73/0.61	0.94/0.91
	CP	0.98/0.96	1.00/1.00	0.78/0.73	0.99/0.93	0.43/0.40	0.92/0.85	0.02/0.01	0.34/0.27	0.38/0.33	0.94/0.91
	MP1	1.00/0.99	1.00/1.00	0.94/0.80	1.00/0.97	0.64/0.46	0.94/0.84	0.14/0.05	0.47/0.33	0.70/0.56	0.93/0.88
	Naive Drifting	1.00/0.99	1.00/1.00	0.94/0.86	0.99/0.96	0.56/0.47	0.98/0.89	0.15/0.11	0.61/0.55	0.46/0.37	0.93/0.88
	IDP	1.00/0.99	1.00/1.00	0.97/0.91	1.00/0.97	0.76/0.63	0.96/0.90	0.34/0.23	0.65/ 0.57	0.78/0.68	0.97/0.92
CNN	DP	1.00/1.00	1.00/1.00	1.00/0.95	1.00/0.99	0.88/0.77	0.97/ 0.93	0.53/0.43	0.81/0.71	0.91/0.74	0.98/0.96
	Flow	1.00/0.99	1.00/1.00	1.00/0.96	1.00/0.99	0.83/0.77	0.96/0.91	0.43/0.31	0.78/0.63	0.81/0.69	0.98/0.96
	MIP	1.00/1.00	1.00/1.00	0.99/0.96	1.00/0.98	0.83/0.69	0.94/0.84	0.48/ 0.45	0.73/0.62	0.63/0.49	0.98/0.97
	CP	1.00/0.98	1.00/1.00	0.88/0.76	1.00/0.98	0.56/0.49	0.84/0.80	0.01/0.00	0.29/0.22	0.49/0.33	0.94/0.91
	MP1	1.00/1.00	1.00/1.00	0.99/0.94	1.00/0.98	0.89/0.78	0.98/0.91	0.33/0.22	0.57/0.44	0.81/0.70	0.97/0.96
	Naive Drifting	1.00/0.99	1.00/1.00	0.99/0.95	1.00/0.97	0.83/0.74	0.96/0.90	0.46/0.33	0.74/0.61	0.59/0.48	0.96/0.94
	IDP	1.00/1.00	1.00/1.00	1.00/0.96	1.00/0.98	0.88/0.72	0.94/0.86	0.48/0.36	0.78/0.60	0.84/0.53	0.98/0.91

Table 10: Detailed 2D simulation results on **image-based** robotic manipulation tasks. For each task, we report the best success rate / the mean over the last five evaluations during training. Results are averaged over three training seeds.

Architecture	Method	Lift		Can		Square		Tool-Hang	Push-T
		mh	ph	mh	ph	mh	ph		
Transformer	DP	1.00/0.98	1.00/0.99	0.97/0.92	0.99/0.96	0.85/0.74	0.97/0.89	0.35/0.17	0.93/0.87
	Flow	1.00/0.99	1.00/0.99	0.92/0.85	0.99/0.94	0.81/0.73	0.97/0.88	0.35/ 0.26	0.90/0.84
	MIP	1.00/0.97	1.00/1.00	0.91/0.86	0.99/0.94	0.85/0.73	0.98/0.86	0.33/0.14	0.93/0.89
	CP	1.00/0.98	1.00/1.00	0.78/0.64	0.86/0.61	0.72/0.63	0.92/0.87	0.25/0.16	0.86/0.82
	MP1	1.00/1.00	1.00/0.99	0.91/0.82	0.79/0.69	0.85/0.75	0.95/ 0.91	0.30/0.13	0.83/0.72
	Naive Drifting	1.00/0.99	1.00/0.99	0.95/0.90	0.97/0.86	0.85/0.73	0.94/0.89	0.37/0.19	0.90/0.86
	Ours	1.00/0.98	1.00/1.00	0.97/0.91	0.99/0.94	0.81/0.74	0.96/0.89	0.37/0.21	0.92/0.87
CNN	DP	1.00/0.99	1.00/0.99	0.82/0.73	0.98/0.92	0.89/0.78	0.97/ 0.93	0.61/0.48	0.92/ 0.89
	Flow	1.00/0.98	1.00/0.99	0.93/0.81	0.99/0.93	0.91/0.83	0.99/0.92	0.59/0.45	0.93/0.88
	MIP	1.00/0.96	1.00/0.99	0.87/0.81	0.96/0.92	0.79/0.68	0.91/0.81	0.60/0.42	0.81/0.79
	CP	1.00/0.98	1.00/1.00	0.76/0.62	0.75/0.59	0.67/0.56	0.92/0.86	0.19/0.09	0.89/0.85
	MP1	1.00/0.95	1.00/0.99	0.83/0.72	0.74/0.66	0.86/0.80	0.97/0.88	0.55/0.40	0.90/0.86
	Naive Drifting	0.99/0.94	1.00/0.98	0.85/0.69	0.93/0.83	0.75/0.64	0.90/0.80	0.55/0.33	0.91/0.87
	Ours	1.00/0.96	1.00/0.99	0.92/ 0.85	0.99/0.94	0.83/0.75	0.92/0.84	0.59/0.45	0.92/0.86

Table 11: Detailed 3D simulation results on **Adroit**, **DexArt**, and **Meta-World**. Results are averaged over 3 random seeds and reported as mean $\pm$ standard deviation. The best result in each column is highlighted in **bold**.

Methods	NFE	Adroit			DexArt				Meta-World (Easy)			
		Hammer	Door	Pen	Laptop	Faucet	Bucket	Toilet	Coffee Button	Dial Turn	Door Close	Door Lock
DP3	10	100±0	58±5	53±3	83±2	40±2	25±1	74±2	100±0	91±3	100±0	100±0
Flow	10	99±1	69±3	45±3	85±0	42±4	26±3	74±3	100±0	81±5	100±0	100±0
CP	1	97±2	66±0	54±2	76±11	38±0	25±5	73±4	90±14	82±2	93±9	74±34
MP1	1	99±1	48±2	58±4	82±3	46±2	26±3	69±3	100±0	91±5	100±0	100±0
FlowPolicy	1	98±2	66±1	55±2	86±2	40±1	28±0	76±3	100±0	81±5	100±0	99±1
Naive Drifting	1	100±0	67±3	43±5	95±4	46±3	29±1	82±6	100±0	97±2	100±0	100±0
Ours	1	100±0	67±2	52±4	96±3	48±1	30±0	83±6	100±0	97±4	100±0	100±0

Meta-World (Easy)										
Methods	NFE	Button Press	ButtonPress Topdown	ButtonPress Topdown Wall	ButtonPress Wall	Door Open	Door Unlock	Drawer Close	Drawer Open	Faucet Close
DP3	10	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
Flow	10	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
CP	1	95±8	70±42	71±41	100±0	85±22	100±0	100±0	92±12	100±0
MP1	1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
FlowPolicy	1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
Naive Drifting	1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
Ours	1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0

Meta-World (Easy)											
Methods	NFE	Faucet Open	Handle Press	Handle Pull	Handle Press Side	Handle Pull Side	Lever Pull	Plate Slide	Plate Slide Back	Plate Slide Back Side	Plate Slide Side
DP3	10	100±0	98±1	59±11	100±0	66±5	85±2	100±0	100±0	100±0	100±0
Flow	10	100±0	100±0	72±8	100±0	72±2	85±1	100±0	100±0	100±0	100±0
CP	1	85±21	91±10	66±0	88±16	17±0	65±25	74±37	93±9	92±11	99±1
MP1	1	100±0	98±3	60±9	100±0	49±6	84±4	100±0	100±0	100±0	100±0
FlowPolicy	1	100±0	97±3	64±1	100±0	41±0	79±4	100±0	100±0	100±0	100±0
Naive Drifting	1	100±0	98±2	68±16	100±0	65±7	88±9	100±0	100±0	100±0	100±0
Ours	1	100±0	100±0	80±1	100±0	67±4	93±2	100±0	100±0	100±0	100±0

Meta-World (Easy)											Meta-World (Medium)			
Methods	NFE	Reach	Reach Wall	Window Close	Window Open	Peg Unplug Side	Basketball	Bin Picking	Box Close	Coffee Pull				
DP3	10	25±2	67±1	100±0	100±0	72±1	100±0	48±18	69±5	88±5				
Flow	10	28±1	73±2	100±0	100±0	71±1	100±0	37±13	67±1	88±2				
CP	1	27±1	56±16	96±6	75±33	57±23	96±0	48±0	60±18	72±22				
MP1	1	28±1	76±2	100±0	100±0	71±3	100±0	44±15	69±10	91±2				
FlowPolicy	1	30±1	70±2	100±0	99±0	71±6	86±8	35±11	74±2	88±1				
Naive Drifting	1	48±2	70±4	100±0	100±0	62±9	100±0	35±21	53±10	85±4				
Ours	1	30±3	67±6	100±0	100±0	77±3	100±0	40±6	70±4	85±1				

Meta-World (Medium)											Meta-World (Hard)	
Methods	NFE	Coffee Push	Hammer	Peg Insert Side	Push Wall	Soccer	Sweep	Sweep Into	Hand Insert	Pick Out of Hole		
DP3	10	89±3	87±3	83±4	88±7	26±2	95±5	46±14	15±2	51±0		
Flow	10	89±2	88±4	87±7	90±2	31±5	96±1	48±18	17±2	46±6		
CP	1	86±14	71±25	72±0	47±0	20±8	58±39	23±8	17±4	30±0		
MP1	1	92±1	90±5	86±8	89±3	30±2	95±3	43±13	16±2	38±12		
FlowPolicy	1	93±5	85±4	75±6	51±3	29±2	84±5	37±21	20±1	22±10		
Naive Drifting	1	92±2	84±3	85±4	93±6	33±8	93±5	25±8	12±2	40±11		
Ours	1	92±3	90±4	79±5	95±2	27±1	100±0	32±17	20±6	44±4		

Meta-World (Hard)											Meta-World (Very Hard)	
Methods	NFE	Assembly	Pick Place	Push	Shelf Place	Disassemble	Stick Pull	Stick Push	Pick Place Wall	Avg.		
DP3	10	94±1	54±3	57±3	41±15	78±3	68±1	100±0	72±4	79.4±2.4		
Flow	10	94±1	61±2	58±4	41±9	74±1	67±3	100±0	72±2	79.9±2.2		
CP	1	74±25	51±19	35±24	36±4	60±2	48±23	99±1	67±11	68.4±11.9		
MP1	1	96±1	65±1	73±7	44±12	70±3	72±4	100±0	78±2	79.7±2.8		
FlowPolicy	1	96±3	60±4	58±6	39±6	63±4	64±1	100±0	69±4	76.9±2.5		
Naive Drifting	1	97±2	65±8	57±6	45±4	90±4	58±12	100±0	73±6	79.8±3.5		
Ours	1	96±3	57±4	65±4	43±2	91±1	69±2	100±0	68±2	81.3±2.0		

Table 12: **Success rates of experts on 3D simulation tasks.** We evaluate 200 episodes for each task. Results are averaged over 3 random seeds and reported as mean $\pm$ standard deviation.

Methods	Adroit			DexArt				Meta-World (Easy)			
	Hammer	Door	Pen	Laptop	Faucet	Bucket	Toilet	Coffee Button	Dial Turn	Door Close	Door Lock
Expert	100±0	97±1	96±1	91±1	54±4	67±2	76±1	100±0	100±0	100±0	100±0
Meta-World (Easy)											
Methods	Button Press	ButtonPress Topdown	ButtonPress Topdown Wall	ButtonPress Wall	Door Open	Door Unlock	Drawer Close	Drawer Open	Faucet Close		
Expert	100±0	100±0	100±0	97±2	97±2	100±0	100±0	100±0	100±0	100±0	100±0
Meta-World (Easy)											
Methods	Faucet Open	Handle Press	Handle Pull	Handle Press Side	Handle Pull Side	Lever Pull	Plate Slide	Plate Slide Back	Plate Slide Back Side	Plate Slide Side	
Expert	100±0	100±0	100±0	100±0	100±0	98±0	100±0	100±0	100±0	100±0	100±0
Meta-World (Easy)											
Methods	Meta-World (Easy)					Meta-World (Medium)					
	Reach	Reach Wall	Window Close	Window Open	Peg Unplug Side	Basketball	Bin Picking	Box Close	Coffee Pull		
Expert	100±0	100±0	100±0	100±0	99±1	100±0	96±1	88±2	100±0		
Meta-World (Medium)											
Methods	Meta-World (Medium)							Meta-World (Hard)			
	Coffee Push	Hammer	Peg Insert Side	Push Wall	Soccer	Sweep	Sweep Into	Hand Insert	Pick Out of Hole		
Expert	100±0	100±0	91±2	100±0	91±2	100±0	91±1	100±0	100±0		
Meta-World (Hard)											
Methods	Meta-World (Hard)			Meta-World (Very Hard)						Avg.	
	Assembly	Pick Place	Push	Shelf Place	Disassemble	Stick Pull	Stick Push	Pick Place Wall			
Expert	100±0	100±0	99±0	99±0	95±1	95±1	100±0	100±0	96.7±0.5		

Table 13: Success rates (mean $\pm$ standard deviation across three seeds, %). IDP w/o geometry retains both training inputs and sets $M_i = 0$.

Task	Direct input only	Direct input + geometry	IDP w/o geometry	IDP
Door	96.67 $\pm$ 1.15	96.33 $\pm$ 0.58	95.67 $\pm$ 2.31	97.67 $\pm$ 2.08
Box-Close	63.33 $\pm$ 8.74	58.50 $\pm$ 6.50	62.67 $\pm$ 3.55	67.17 $\pm$ 6.45
Push	44.67 $\pm$ 4.16	49.33 $\pm$ 5.51	50.67 $\pm$ 6.51	52.33 $\pm$ 5.69

E.2 Geometry and Expert-Proximal Training

Conditional expert geometry and expert-proximal supervision act jointly in the training objective. We independently enable these two components on Door, Box-Close, and Push to examine their individual and combined effects. Each configuration uses three training seeds with matched initialization and evaluation initial states. Each evaluation uses 200 episodes on Box-Close and 100 on Door and Push.

Combining both components gives the highest mean success on all three tasks. We also test the correspondence between each observation and its geometric weights by permuting the matrices M_i across samples. This reduces success from 97.67% to 93.00% on Door and from 67.17% to 60.83% on Box-Close, supporting the use of geometry associated with the current observation.

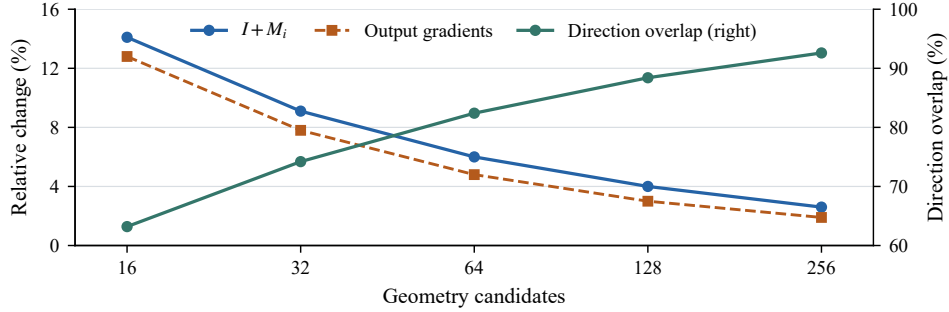
E.3 Analysis of Conditional Expert Geometry

Observation relevance. Observation similarity determines how much each expert action contributes to G_i . We examine whether highly similar observations correspond to related expert actions and object states. Using the ToolHang-State policy, we sample 10,240 observations from the training data. The most similar reference has a normalized expert-action distance of 1.622, compared with 5.879

for a random reference from the same batch. The corresponding object-state distances are 0.643 and 4.768. References from other demonstration trajectories receive 98.9% of the total weight.

A separate encoder comparison keeps the observations and reference candidates fixed. The reference expert-action distance is 2.930 with the policy encoder, 3.263 with a frozen ImageNet encoder, and 4.344 with a random encoder. These results support using the policy’s observation features to identify related expert actions.

Number of reference demonstrations. The geometry estimate uses a finite set of expert actions from the training batch. Reducing this set can change the directions emphasized by M_i and the resulting training signal. We keep the policy parameters and evaluated observations fixed and vary only the number of geometry candidates, using 1,024 candidates as the reference. We measure relative changes in $I + M_i$, overlap between the emphasized coordinates, and changes in the gradients at the two training outputs (Figure 5).



Reference removal	$I + M_i$ change (%)	Direction overlap (%)	Output gradient change (%)
Most similar reference	0.03	99.9	0.02
All similar references	18.0	33.2	17.8

Figure 5: Effect of the number of reference demonstrations on geometric supervision. Results are measured relative to 1,024 candidates with fixed policy parameters and observations. Changes in the weights and output gradients are reported as median relative L_2 differences. Direction overlap is the Jaccard similarity of coordinates with $m_{i,\ell,d} > 0.1$. Gradient change is measured on the concatenated gradients at the two training outputs. The two reference-removal conditions are shown separately below the plot.

With 64 candidates, the median relative changes in $I + M_i$ and the output gradients are 6.0% and 4.8%, respectively. Removing the most similar candidate changes $I + M_i$ by 0.03%, whereas removing all similar candidates reduces direction overlap to 33.2%. We also train policies on Adroit Door while fixing the optimization batch at 128. Using 32, 64, and 128 geometry candidates gives mean success rates of 92.33%, 94.67%, and 97.67%, respectively, across three training seeds.

Visual observation changes. For image observations, visual features determine the expert-action weights used to construct G_i . Appearance changes can therefore alter the reference actions even when the robot and objects remain in the same state. We compare clean and perturbed images from fixed simulator states, measuring reference-set overlap and changes in predicted actions. Actual task-state changes provide a reference for interpreting the magnitude of these changes.

The tested image corruption, background changes, and distractors produce smaller action changes than the task-state reference. Lighting produces larger changes for both methods.

F Limitations

Mode coverage. Under balanced Pick Peach demonstrations, all evaluated methods learn a single placement mode. Extending IDP to represent multiple valid action modes is a direction for future work.

Table 14: Visual perturbations at fixed task states. Policies trained for 150,000 steps are evaluated on 32 paired observations. The distractor condition uses nine valid pairs in which the task objects remain visible. Entries are IDP / IDP w/o geometry. Top-10 overlap is the Jaccard similarity of the reference sets. Normalized action change is $\|a' - a\|_2 / \max(\|a\|_2, 10^{-12})$, where a is the clean-observation prediction.

Change	Top-10 overlap	Normalized action change
Gaussian corruption	0.994 / 0.989	0.0032 / 0.0033
Background recoloring	1.000 / 1.000	0.0033 / 0.0032
Distractor block	1.000 / 1.000	0.0069 / 0.0073
Lighting	0.885 / 0.867	0.0791 / 0.0699
Task-state change	0.894 / 0.955	0.0400 / 0.0404

Related demonstrations. Conditional expert geometry depends on related demonstrations in the minibatch. Removing all similar candidates substantially changes the emphasized directions in the support experiment (Appendix E.3).

Coordinate-wise geometry. The diagonal M_i assigns separate weights to action coordinates and time positions. A structured matrix could additionally model interactions between these coordinates.

Observation representations. On Pick Peach, qpos-based IDP achieves 30% success, compared with 40% for MIP. Similar joint positions can occur with different peach–gripper relationships, so useful geometry depends on observation features that distinguish task contexts.

Batch reference. The reference is estimated from each minibatch, so its composition affects the resulting weights. Running estimates and task-conditioned references are possible extensions for broader data and multi-task training.

Variation across tasks. Tasks close to the success ceiling leave little room for improvement. In the supplementary ToolHang-State evaluations, the differences in success and perturbation recovery between IDP and IDP w/o geometry fall within the observed variation across training seeds.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction accurately describe our proposed framework Implicit Drifting Policy (IDP), its contributions, and experimental scope, which align with the content presented in Section 4 and Section 5.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We dedicate a Limitations section in the Appendix F that discusses, based on our experimental observations, the mode collapse on bimodal demonstrations, the local-density requirement of the conditional expert geometry, the diagonal approximation of M_i , the degradation in proprioceptive (qpos-only) regimes, and the per-batch nature of the reference geometry.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Proposition 3.1 states the empirical-singleton assumptions. The geometry-induced potential is described in Section 4.1. The corresponding derivations are provided in Appendix A.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We elaborate our methodology in detail in Section 4 (Implicit Drifting Policy and its training objective), and disclose implementation details in the Appendix C.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: Our training and evaluation code, together with usage instructions, is publicly available through the project page linked on the first page.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: The implementation details (data splits, hyperparameters, optimizers, etc.) of the simulation experiments follow the protocol of MIP [25] for 2D and OFP [20] / FlowPolicy [34] for 3D, as stated in Appendix C.1 and Appendix C.2. The specific details of the real-world experiments are described in the Appendix C.3.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: For 2D tasks, results are averaged across different architectures, 3 training seeds, and 50 environmental initial conditions. For 3D tasks, results are averaged across 3 random seeds over 56 evaluation environments. Standard deviations across seeds are reported in the per-task tables in the Appendix.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report the computational resources required for all experiments in the Appendix.

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We confirm that the research conducted in this paper conforms, in every respect, with the NeurIPS Code of Ethics.

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Efficient action generation can support more responsive robot assistance and automation. Prediction errors during deployment may cause unintended contact or damage, motivating safety monitoring and task-specific validation.

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper poses no such risks (no pre-trained language models release, no scraped data, no image generator).

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: For all prior works we cite and compare against (Diffusion Policy, DP3, Consistency Policy, MIP, OFP, FlowPolicy, etc.) and all benchmarks we use (PushT, Robomimic, Adroit, DexArt, MetaWorld), we make reference to their original papers.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released code is documented in the repository, including installation, training, and evaluation instructions.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.