GENERATIVE CONTROL AS OPTIMIZATION: TIME UNCONDITIONAL FLOW MATCHING FOR ADAPTIVE AND ROBUST ROBOTIC CONTROL

Zunzhe Zhang[1*], **Runhan Huang**[1*], **Yicheng Liu**[1], **Shaoting Zhu**[1], **Linzhan Mou**[2], **Hang Zhao**[1†]
[1]Tsinghua University, [2]Princeton University

ABSTRACT

Diffusion models and flow matching have become a cornerstone of robotic imitation learning, yet they suffer from a structural inefficiency where inference is often bound to a fixed integration schedule that is agnostic to state complexity. This paradigm forces the policy to expend the same computational budget on trivial motions as it does on complex tasks. We introduce **Generative Control as Optimization (GeCO)**, a time-unconditional framework that transforms action synthesis from trajectory integration into iterative optimization. GeCO learns a stationary velocity field in the action-sequence space where expert behaviors form stable attractors. Consequently, test-time inference becomes an adaptive process that allocates computation based on convergence—exiting early for simple states while refining longer for difficult ones. Furthermore, this stationary geometry yields an intrinsic, training-free safety signal, as the field norm at the optimized action serves as a robust out-of-distribution (OOD) detector, remaining low for in-distribution states while significantly increasing for anomalies. We validate GeCO on standard simulation benchmarks and demonstrate seamless scaling to π_0-series Vision-Language-Action (VLA) models. As a plug-and-play replacement for standard flow-matching heads, GeCO improves success rates and efficiency with an optimization-native mechanism for safe deployment.

1 INTRODUCTION

Generative modeling has rapidly established itself as a cornerstone of modern robotic imitation learning (Black et al., 2024; Chi et al., 2025; Janner et al., 2022; Huang et al., 2025b;a; Ze et al., 2024; Kim et al., 2024; Bu et al., 2025). By representing policies as conditional distributions rather than deterministic mappings, approaches such as diffusion models (Ho et al., 2020; Song et al., 2020; Zhang et al., 2023) and flow matching (Lipman et al., 2022; Liu et al., 2022) have demonstrated exceptional capabilities in capturing the multi-modal nature of human behavior. These methods excel at synthesizing complex trajectories for manipulation and locomotion tasks (Chi et al., 2025; Intelligence et al., 2025; Huang et al., 2025b; Ze et al., 2024; Ajay et al., 2022), driving state-of-the-art performance across diverse benchmarks (Chen et al., 2025b; Liu et al., 2023a; Zhang et al., 2025). Their success stems largely from iteratively refining noise into feasible actions via time-conditioned dynamics and effectively decomposing the challenging generation problem into a sequence of manageable denoising or flow integration steps (Ho et al., 2020; Lipman et al., 2022).

Yet, this reliance on time-conditioning introduces a fundamental structural inefficiency for control applications, a limitation that persists across standard formulations. These methods learn dynamic vector fields that evolve according to a pre-defined fictitious time schedule (Wang & Du, 2025): while this schedule bridges Gaussian noise and the expert data distribution, it creates a rigid inference paradigm that is agnostic to the actual complexity of the current robot state. The policy is compelled to execute a fixed number of integration steps regardless of convergence, thereby wasting computational resources on trivial motions while potentially failing to sufficiently refine actions for complex tasks. This "blind integration" problem not only hinders efficiency but also obscures the

[*]Equal contribution. Listing order is random.
[†]Corresponding at hangzhao@mail.tsinghua.edu.cn

geometric structure of the policy. Time-varying fields lack a stationary energy landscape (Wang & Du, 2025; Sun et al., 2025): with the field direction shifting across timesteps, there is no intrinsic mechanism to verify action validity or detect out-of-distribution (OOD) scenarios, a critical gap for safe real-world deployment (Hodge et al., 2025).

To address these limitations, we reimagine generative control through a **time-unconditional** lens, grounding action synthesis in iterative optimization rather than trajectory integration. Our framework, **Generative Control as Optimization (GeCO)**, embodies a paradigm shift: instead of learning time-dependent dynamics, we learn a single, stationary velocity field in the action-sequence space where expert behaviors form stable attractors. To ensure robust convergence in continuous action spaces, we incorporate a velocity rescaling mechanism that modulates flow magnitude based on distance to the expert manifold, creating a geometric sink around target modes. This transforms inference from a fixed-schedule process into an adaptive optimization loop that naturally settles at equilibrium.

The stationary nature of GeCO's velocity field unlocks two transformative capabilities absent in time-dependent baselines, directly addressing our core motivations of adaptive computation and intrinsic OOD awareness. First, it enables **adaptive inference** rooted in convergence: rather than adhering to an arbitrary step count, the policy dynamically adjusts computation by exiting early for simple states (e.g., basic reaching) and spending more time refining actions for high-precision tasks (e.g., delicate manipulation). This decouples planning horizon from training schedules, allocating computational resources efficiently. Second, and crucially, the static geometric structure provides **intrinsic OOD awareness** as a training-free safety signal. The residual norm of the velocity field acts as a robust epistemic uncertainty metric: in-distribution states converge to zero-velocity attractors (low norm), while OOD states, lacking a learned manifold to settle on, exhibit persistent, non-vanishing gradients. This allows the robot to detect anomalies directly from the field's energy, without auxiliary networks or ensembles.

Notably, GeCO exhibits strong architectural versatility, seamlessly supporting both standard Diffusion Transformers (DiT) (Chi et al., 2025; Dong et al., 2024; 2025) and large-scale Vision-Language-Action (VLA) foundation models (Black et al., 2024; Intelligence et al., 2025). A defining advantage lies in its **plug-and-play** design: it serves as a direct drop-in replacement for standard flow-matching heads (Black et al., 2024), requiring minimal architectural adjustments or modifications to the loss function to integrate into existing systems. Building on this versatility, we conduct extensive validation of GeCO on standard simulation benchmarks (Chen et al., 2025b; Liu et al., 2023a; Zhang et al., 2025). Empirically, GeCO outperforms state-of-the-art time-conditioned baselines while fundamentally redefining inference efficiency through the modulation of computational resources based on task complexity, rather than adherence to rigid fixed schedules. Beyond pure performance gains, we further establish that the intrinsic geometric signals of GeCO's stationary field enable robust detection of task-level distribution shifts, delivering an optimization-native safety mechanism critical for reliable real-world robotic deployment.

2 Related Work

Generative Models for Robotic Control. Generative policy learning has become a dominant paradigm for robotic control in recent years. One prominent direction leverages the reasoning and perceptual capabilities of Vision-Language Models (VLMs) (Beyer et al., 2024; Bai et al., 2023; Liu et al., 2023b) for high-level planning (Driess et al., 2023; Liang et al., 2022; Tian et al., 2024) and action generation (Kim et al., 2024; 2025; Bu et al., 2025). Parallel efforts utilize video generation models (Liu et al., 2024b; Yang et al., 2024) to serve as predictive priors for robotic decision-making (Du et al., 2023; Chen et al., 2025a; Shen et al., 2025). In the realm of precise robotic control, diffusion (Ho et al., 2020; Song et al., 2020) and flow matching (Lipman et al., 2022) have demonstrated exceptional effectiveness, functioning as robust planners and action-chunk generators (Chi et al., 2025; Huang et al., 2025a; Zhou et al., 2024; Huang et al., 2025b; Janner et al., 2022; Ajay et al., 2022; Dong et al., 2025; Ze et al., 2024). Notably, the integration of continuous flow-matching heads into VLA architectures has emerged as a powerful paradigm (Black et al., 2024; Intelligence et al., 2025; Li et al., 2025; Shukor et al., 2025; Jiang et al., 2025); this approach bridges the semantic understanding of VLM backbones with high-frequency, continuous action execution. Such continuous generation mechanisms are widely adopted in modern Vision-Language-Action systems to replace

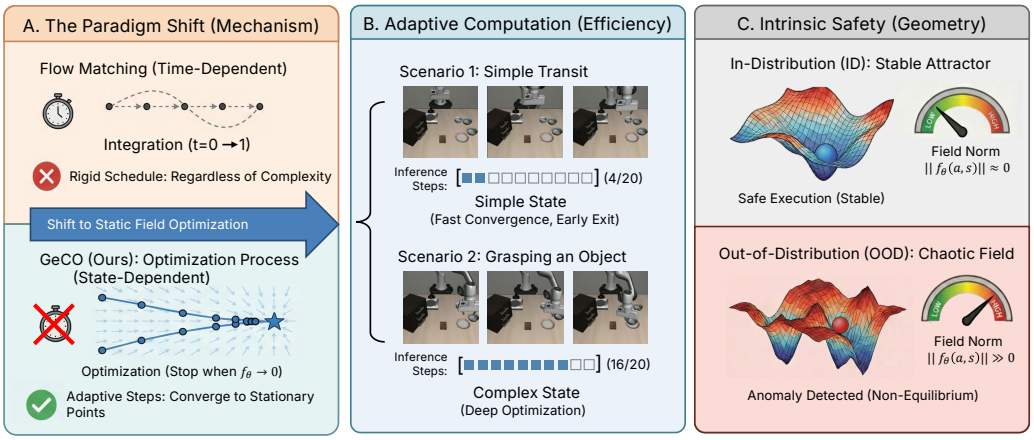


Figure 1: **Generative Control as Optimization (GeCO). (A) The Paradigm Shift:** Unlike standard flow matching which relies on rigid, time-dependent integration schedules (top), GeCO learns a stationary velocity field where inference becomes an iterative optimization process toward stable attractors (bottom). **(B) Adaptive Computation:** This formulation enables the policy to dynamically allocate computational budget based on state complexity—exiting early for simple transit phases (Scenario 1) while performing deep refinement for precise manipulation (Scenario 2). **(C) Intrinsic Safety:** The stationary geometry provides a zero-shot safety mechanism. In-distribution (ID) states converge to low-energy equilibria ($||f_\theta|| \approx 0$), whereas out-of-distribution (OOD) anomalies exhibit persistently high field norms ($||f_\theta|| \gg 0$), enabling robust detection.

purely autoregressive, discretized action decoding (Kim et al., 2024; 2025), enabling higher fidelity control and improved real-time performance (Black et al., 2025; Intelligence et al., 2025). Our work focuses on this continuous generative control family of flow matching, which continues to drive advancements across domains ranging from robotic manipulation to navigation.

Time-Unconditional Generative Dynamics. Most diffusion and flow-based models (Ho et al., 2020; Song et al., 2020; Lipman et al., 2022) formulate generation as a *time-conditioned* dynamical system, where the vector field varies with an explicit noise or time variable, typically necessitating a fixed integration horizon. Recent studies (Sun et al., 2025) indicate that removing this explicit conditioning can lead to graceful degradation, or even performance gains, suggesting that dynamics can often be inferred implicitly from the state. However, naively discarding time conditioning does not guarantee stable equilibrium behavior or well-posed optimization dynamics (Wang & Du, 2025). (Wang & Du, 2025) address this by learning a stationary vector field compatible with an implicit energy landscape, enabling optimization-driven sampling. While this paradigm shows promise in image generation, its effectiveness in robotic control, which demands rigorous accuracy and sequential stability, still remains an open question.

3 METHOD

Our goal is to turn generative control for robotic manipulation from a fixed-time integration process into an optimization problem over a time-invariant velocity field. At each control step, the policy observes images, proprioception, and a language instruction, and synthesizes a short-horizon action sequence. We first revisit time-conditioned diffusion and flow matching for such control, then introduce our time-unconditional field and optimization-based inference, and finally show how the same geometry yields intrinsic OOD awareness.

3.1 REVISITING TIME-CONDITIONED GENERATIVE CONTROL

Diffusion- and flow-based controllers for manipulation typically learn a *time-conditioned* velocity field

$$v_\theta(x_\gamma, \gamma, s_t),$$

where x_γ is a noisy or partially denoised action sequence, γ is a time variable, and s_t is the current observation and instruction. Inference follows an *integration path in the time domain*: starting from noise, the policy traverses γ from 0 to 1 using steps $\{\Delta\gamma_k\}_k$ with $\sum_k \Delta\gamma_k \approx 1$.

Although one can change the number of steps and step sizes at deployment, these choices are constrained by the requirement of representing a valid integration of the time-conditioned field over $\gamma \in [0, 1]$. Stopping too early under-integrates the ODE/SDE; letting $\sum_k \Delta\gamma_k > 1$ pushes the system beyond the interval on which the field is defined and trained. Thus the geometry of $v_\theta(\cdot, \gamma, s_t)$ is always tied to a specific notion of time progress.

This design couples three roles into a single scalar γ: tracking progress along the generative trajectory, controlling the strength of the field, and parameterizing the integration domain. As a result, the field geometry changes with γ, and there is no single, time-invariant notion of "good actions" or a creteria whose magnitude directly measures how close a final action is to the in-distribution manifold.

GeCO decouples this dependence on time: instead of conditioning on γ, we learn a *stationary* field over action sequences and use it as the objective of an unconstrained optimization procedure.

3.2 Time-Unconditional Velocity Fields for Control

At timestep t, the robot observes s_t (multi-view RGB of the workspace, proprioception, and a language instruction) and predicts a sequence of future low-level actions.

$$a = a_{t:t+T_a-1} \in \mathbb{R}^{T_a \times d_a},$$

Rather than a time-indexed $v_\theta(\cdot, \gamma, s_t)$, GeCO learns a *time-unconditional* velocity field $f_\theta(x, s_t)$ over action sequences. We want $f_\theta(\cdot, s_t)$ to point from noisy or suboptimal sequences toward *equilibrium actions* that solve the current task, and to vanish near ground truth in-distribution actions, in order .

To train such a field, we introduce γ only as a implicit training variable. Given a demonstration a and Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$, we form

$$x_\gamma = \gamma a + (1 - \gamma)\varepsilon, \qquad \gamma \sim \mathcal{U}(0, 1), \tag{1}$$

and crucially *do not* provide γ to the model. At each x_γ we define a restoring direction

$$g^\star(a, \varepsilon, \gamma) = (\varepsilon - a)\, c(\gamma), \tag{2}$$

where $c(\gamma)$ is a scalar schedule that decays to zero as $\gamma \to 1$. Because $c(1) = 0$, the target field vanishes at $x_\gamma = a$, transforming ground-truth action sequences into natural stationary equilibrium points of the learned field. Given such rescaling mechanism, the denoising process is able to automatically stop in the ground truth stationary points, effectively solving the non-equilibrium nature of the vanilla time-unconditional flow matching.

Conditioned on s_t, the loss is

$$\mathcal{L}(\theta) = \mathbb{E}_{(s_t,a)}\mathbb{E}_{\varepsilon,\gamma}\Big\| f_\theta(x_\gamma, s_t) - g^\star(a, \varepsilon, \gamma)\Big\|^2. \tag{3}$$

Here γ only controls the interpolation and the scale via $c(\gamma)$; since it never enters the model, $f_\theta(\cdot, s_t)$ must organize all training pairs into a *single* consistent field for each scene. This encourages f_θ to approximate the gradient of an implicit potential field over action sequences whose minima lie near successful demonstrations.

3.3 Optimization-Based Inference

Once f_θ has been learned, inference is no longer framed as integrating over a time axis. At deployment, the controller runs in closed loop: at each timestep t, it observes s_t and solves a small optimization problem in action space using $f_\theta(\cdot, s_t)$.

We initialize $a^{(0)}$ from a prior $p_{\text{prior}}(\cdot \mid s_t)$, e.g., isotropic Gaussian noise, and then perform K gradient-based updates:

$$a^{(k+1)} = a^{(k)} - \eta_k\, f_\theta(a^{(k)}, s_t), \tag{4}$$

Algorithm 1 Adaptive Inference (GeCO)

Input: Observation s, Max steps K, Tolerance τ, Rate η
Output: Action $\hat{a}$, OOD Score S
$a^{(0)} \sim \mathcal{N}(0, I)$ ▷ *Initialize prior*
for $k = 0, \ldots, K-1$ **do**
  $v_k \leftarrow f_\theta(a^{(k)}, s)$
  if $\|v_k\|_2 < \tau$ **then**
    break ▷ *Adaptive early exit*
  end if
  $a^{(k+1)} \leftarrow a^{(k)} - \eta \cdot v_k$
end for
$\hat{a} \leftarrow a^{(k)}$
return $\hat{a}$, $\|f_\theta(\hat{a}, s)\|_2$

where η_k may depend on the sample or iteration. Because there is no time index, and there is no requirement that the effective "total step length" $\sum_k \eta_k$ equal a particular value. Taking more or fewer steps, or even $\sum_k \eta_k > 1$, remains a natural optimization process in the same stationary field, rather than violating a time-integration constraint and leading to non-equilibrium divergence.

Convergence is monitored directly through the field:

$$\|f_\theta(a^{(k)}, s_t)\| \leq \tau_{\text{opt}} \quad \text{or} \quad k \geq K_{\max}. \tag{5}$$

Simple scenes (e.g., unobstructed reaches) reach a low-norm region in few iterations; cluttered or contact-rich scenes require more. GeCO behaves as an optimization-based, receding-horizon manipulation controller grounded in a learned generative field. Detailed inference pseudocode can be found in Algo.1.

3.4 Intrinsic OOD Detection from Field Geometry

GeCO also provides *optimization-native* OOD awareness. Here OOD is defined at the level of the conditional field: training exposes the model to (s_t, a) pairs from $p_{\text{data}}(s_t, a)$, and thus to conditional fields $f_\theta(\cdot, s_t)$ induced by $s_t \sim p_{\text{data}}(s_t)$. At test time, an observation s_t is OOD if it induces a field whose geometry is inconsistent with this family; optimization on such a field need not reach a low-norm equilibrium, and this breakdown in convergence becomes an OOD signal.

Given a test observation s_t, we run optimization as in equation 4–equation 5, obtaining an approximate equilibrium $\hat{a}(s_t)$, and define

$$\text{Score}(s_t) = \big\|f_\theta(\hat{a}(s_t),\, s_t)\big\|. \tag{6}$$

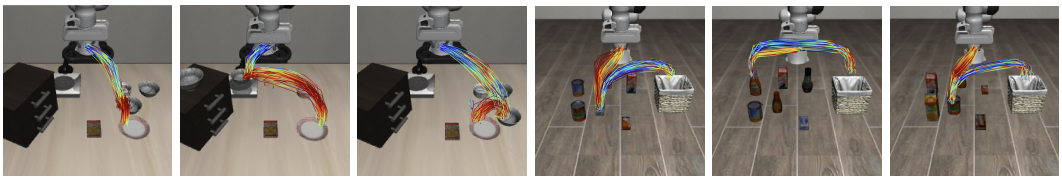
For in-distribution robot control states $s_t \sim p_{\text{data}}$, the induced field resembles those seen during training: optimization can move x_γ into regions that behave like high-γ samples, where $c(\gamma)$ is small and the field is trained to nearly vanish, so $\text{Score}(s_t)$ is low. When s_t is OOD, the induced field is itself out-of-distribution: optimization operates in regions never trained to lead to small norm, the gradient norm at the end of the optimization process at $\hat{a}(s_t)$ remains large. We obtain a binary OOD decision via

$$\text{isOOD}(s_t) = \mathbf{1}\{\text{Score}(s_t) > \tau_{\text{OOD}}\}. \tag{7}$$

This incurs essentially no overhead, as the same gradient norms are already computed for convergence checking.

3.5 Plug-and-Play Integration with VLA Systems

Modern flow-based Vision-Language-Action (VLA) systems consist of two core components: (1) a Vision-Language Model (VLM) that fuses multi-modal inputs (multi-view RGB $\mathbf{v} \in \mathbb{R}^{H \times W \times 3 \times N_v}$, proprioception $\mathbf{p} \in \mathbb{R}^{d_p}$, language instruction $l \in \mathcal{L}$) into a task-aware conditional signal $s_t = \text{VLM}(\mathbf{v}, \mathbf{p}, l)$; (2) a time-conditioned flow-matching head that takes s_t and action sequence x_γ as inputs to predict $v_\theta(x_\gamma, \gamma, s_t)$.



(a) (b) (c) (d) (e) (f)

Figure 2: Computation Follows Task Complexity. We visualize the spatial distribution of inference effort along a single rollout. The first three panels (a–c) are sampled from LIBERO-Spatial, and the last three panels (d–f) are from LIBERO-Object. The color of each line encodes the number of function evaluations (NFE) required for convergence at that state, ranging from blue (NFE = 1) to red (NFE = 20). This visualization illustrates how GeCO allocates more computation to challenging states while using fewer steps in easier regions.

GeCO enables plug-and-play integration with such VLA architectures via minimal modifications: we directly replace the time-conditioned flow-matching head $v_\theta(x_\gamma, \gamma, s_t)$ with our time-unconditional velocity field $f_\theta(x, s_t)$ (Section 3.2), while retaining the VLM's multi-modal fusion pipeline and the VLA system's original input-output interface. No architectural changes to the VLM or control pipeline are required—at inference, the VLA system generates action sequences $\mathbf{a} \in \mathbb{R}^{T_a \times d_a}$ via GeCO's adaptive optimization loop (Algorithm 1) using the same s_t signal, leveraging the stationary geometry of f_θ to eliminate the rigid time-schedule constraints of baseline flow-matching heads.

4 Experiments

We evaluate **GeCO** along three complementary axes: (1) **Adaptive Efficiency**, analyzing how the optimization-based inference dynamically allocates computation based on task complexity; (2) **Scalability**, assessing the method's effectiveness as a plug-and-play head for large Vision-Language-Action (VLA) models (π_0 series); and (3) **Intrinsic Safety**, verifying the reliability of the stationary field norm as a zero-shot Out-of-Distribution (OOD) detector. Implementation details can be found in Appendix A

4.1 Policy Performance and Adaptive Computation

The Efficiency-Performance Trade-off. Table 1 demonstrates that GeCO achieves a superior trade-off between inference speed and task success compared to fixed-schedule baselines on the LIBERO (Liu et al., 2023a) benchmark. Remarkably, GeCO with a budget of only 5 steps already outperforms Rectified Flow (20 steps), achieving a higher success rate (91.9% vs. 90.0%) while requiring **75% less compute**. When the budget is increased to 20 steps, GeCO reaches a peak success rate of 93.5%. Crucially, the average NFE required to achieve this peak is only 11.6, significantly lower than the maximum budget. This indicates that GeCO's optimizer effectively allocates computational resources, autonomously exiting early for simpler states while reserving the budget for complex ones.

Temporal Analysis: Computation Follows Complexity. To verify that the observed variation in NFE is structurally driven by task complexity rather than randomness, we analyze the inference profile shown in Figure 2. We observe distinct optimization behaviors across different task types:

- **Motion Planning Complexity (LIBERO-Spatial):** For tasks requiring spatial reorientation (Fig. 2 a-c), higher NFE concentrates heavily around bottleneck phases, such as pre-grasp alignment and object placement. In contrast, free-space transit phases converge rapidly, demonstrating that the learned vector field provides a smooth gradient in unobstructed regions.
- **Semantic Grounding (LIBERO-Object):** In tasks where the robot must identify a specific target among multiple distractors based on language instructions (e.g. pick the ketchup

Table 1: **Performance vs. Efficiency on LIBERO.** GeCO achieves higher success rates by allowing adaptive refinement. By increasing the max step budget, GeCO refines actions for complex states, yet the average compute remains low compared to fixed-schedule baselines.

Method	Max Steps	Success Rate (%)					Efficiency
		Goal	Spat.	Obj.	Long	Avg.	**NFE**
Diff. Policy	100 (Fixed)	82.9	91.4	88.9	82.7	86.5	100.0
Rectified flow	20 (Fixed)	92.4	94.6	97.0	76.0	90.0	20.0
GeCO (Ours)	5	91.6	95.4	98.2	82.4	91.9	**5.0**
GeCO (Ours)	10	93.0	95.8	99.0	81.8	92.4	8.7
GeCO (Ours)	20	**95.2**	**95.8**	**99.0**	83.8	**93.5**	11.6
GeCO (Ours)	30	93.6	95.2	98.8	**84.8**	93.1	12.8

Table 2: Success rate (%) of different VLA methods on the **RoboTwin 2.0** benchmark. Each task is evaluated under two difficulty levels: **Easy** and **Hard**. For every method we report the success rate for each task and difficulty level, together with the average performance across all five tasks. Higher values indicate better task completion ability. The best result in each column is highlighted in **bold**.

Method	Adjust Bottle		Beat Block Hammer		Blocks Ranking (Size)		Click Alarmclock		Click Bell		Avg.	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
RDT (Liu et al., 2024a)	81.0	75.0	77.0	37.0	0.0	0.0	61.0	12.0	80.0	9.0	**60.0**	27.0
ACT (Zhao et al., 2023)	**97.0**	23.0	**56.0**	3.0	0.0	0.0	32.0	4.0	**58.0**	3.0	49.0	7.0
π_0 (Black et al., 2024)	90.0	56.0	43.0	21.0	**7.0**	**1.0**	63.0	11.0	44.0	3.0	49.0	18.0
π_0 **with GeCO**	86.0	**66.0**	52.0	**36.0**	0.0	0.0	**72.0**	**16.0**	42.0	**21.0**	50.0	**28.0**

and place it in the basket), we observe a distinct NFE spike at the episode onset (Fig. 2 d-f). This suggests that the optimizer requires more iterations to resolve the visual-semantic correspondence, effectively deliberating to lock onto the specific object defined by the instruction before committing to a trajectory.

These results confirm that GeCO acts as an adaptive controller, dynamically modulating its inference horizon based on both kinematic constraints and perceptual uncertainty.

4.2 Scalability to Vision-Language-Action Models

We evaluate the scalability of GeCO by integrating it into flow-matching-based VLA models, with the π_0-series models selected as a representative example. For a fair comparison, both GeCO and the baselines are fine-tuned from the same pre-trained π_0 base for 30,000 steps. Details can be found in Appendix B.

VLABench Result We benchmark our method on VLABench (Zhang et al., 2025), a challenging suite designed to evaluate long-horizon, language-conditioned robotic manipulation with strong requirements on semantic grounding and multi-step reasoning. As shown in Table 3, π_0 with GeCO achieves the strongest overall performance with an average success rate of **0.36**, improving upon the baseline π_0 (0.294). The most substantial gains are observed on Track 1 and Track 6, which respectively evaluate in-distribution manipulation skills and robustness to unseen textures and visual variations. In addition, GeCO consistently improves results on Track 3 (common-sense and world knowledge) and Track 4 (semantic instruction following), both of which emphasize multi-step reasoning and long-horizon task execution. By contrast, performance on Track 2 (cross-category generalization) remains comparable to standard π_0, suggesting that improvements are more pronounced on execution- and reasoning-intensive tasks than on pure category-level transfer.

RoboTwin 2.0 Benchmark Result RoboTwin 2.0 (Chen et al., 2025b) evaluates bimanual manipulation under two difficulty settings: *Easy* uses clean environments, while *Hard* tests the same policies under domain-randomized evaluation with clutter, texture, lighting, and tabletop-height vari-

Table 3: VLA success rates (%) on VLABench tracks. We report track-wise success and the average over the listed tracks; higher is better.

Method	Track 1	Track 2	Track 3	Track 4	Track 6	Avg.
π_0	47.0	21.2	29.1	17.3	32.2	29.4
$\pi_{0.5}$	40.6	**22.6**	18.0	16.1	25.6	24.5
π_0 **with GeCO**	**61.0**	22.0	**34.0**	**20.0**	**45.0**	**36.0**

Table 4: VLA success rates (%) on LIBERO benchmark. We report track-wise success and the average over the listed tracks; higher is better.

Method	Goal	Spat.	Obj.	Long	Avg.
π_0 + FAST (Pertsch et al., 2025)	88.6	96.4	96.8	60.2	85.5
π_0 (Black et al., 2024)	95.8	96.8	**98.8**	85.2	94.2
$\pi_{0.5}$ (Intelligence et al., 2025)	**98.0**	**98.8**	98.2	92.4	**96.9**
Ours (π_0)	96.4	96.8	97.0	**85.4**	93.9
Ours ($\pi_{0.5}$)	96.4	97.4	98.6	**92.4**	96.2

ations. As shown in Table 2, adding GeCO to π_0 substantially improves robustness in the Hard setting, raising the average success rate from 0.18 to **0.28**, while slightly improving the Easy average (0.49→0.50). The gains are consistent across several contact-rich tasks, including Adjust Bottle (0.56→0.66), Beat Block Hammer (0.21→0.36), Click Alarmclock (0.11→0.16), and Click Bell (0.03→0.21), indicating improved tolerance to visual and scene perturbations. By contrast, Blocks Ranking (Size) remains challenging for all methods (near-zero success in both settings), suggesting that fine-grained ordering under bimanual coordination is still a major bottleneck on RoboTwin 2.0.

LIBERO Benchmark Result LIBERO (Liu et al., 2023a) comprises four suites that isolate different distribution shifts in language-conditioned manipulation: goal changes, spatial relation shifts, object category shifts, and long-horizon compositional execution. In Table 4, GeCO remains comparable overall relative to standard fine-tuning: for π_0 it matches a similar average, and for $\pi_{0.5}$ it stays close. These results suggest that GeCO's remain comparable performance across different base VLM models.

Collectively, these results validate that GeCO acts as a better drop-in replacement for VLA action heads, delivering higher performance and robustness under identical training conditions.

4.3 Intrinsic OOD Detection from Optimization

GeCO performs action synthesis via iterative optimization at each planning call. We hypothesize that under distribution shift, this optimization fails to converge cleanly, leading to larger update magnitudes. We therefore use the **final update norm** produced by a single planning call as an intrinsic anomaly score.

Experiment Setup. We train a single policy on *LIBERO-Goal* (10 tasks) and evaluate OOD detection on *LIBERO-Spatial* (10 tasks) as **Task OOD**. We choose LIBERO-Goal for training because tasks share the same scene layout and objects but differ in goal predicates, making performance sensitive to correctly grounding language instructions into behaviors. At test time, LIBERO-Spatial introduces unseen spatial arrangements of otherwise similar objects, creating a controlled workspace-layout shift while keeping the manipulation vocabulary comparable.

Baseline OOD Signal. Following the Diff-DAgger-style idea of using the diffusion training objective as an uncertainty signal (Lee et al., 2025), we define a lightweight baseline based on the *flow-matching training loss*. The original formulation estimates an expectation over many sampled timesteps (e.g., 512 samples); to avoid extra sampling cost, we use a single-step proxy and evaluate the flow-matching loss at a fixed timestep $t = 1$ for each planning call. This produces one scalar baseline score per planning call.

Table 5: **OOD Detection on LIBERO Suite Shift (Goal → Spatial).** We report planning-level AUROC for separating ID (Goal) and OOD (Spatial) using optimization-dynamics signals computed per planning call. At a fixed operating point, **TNR** is specificity on ID instances and **TPR** is recall on OOD instances. **Time Saved** is the average fraction of interaction time avoided on OOD episodes by early reporting, computed as $1 - t_{\text{report}}/T_{\text{total}}$.

Method	AUROC (↑)	TNR	TPR	Time Saved (%)
Baseline (Moving Avg)	0.53	14.4 %	27.8 %	10.3%
Ours (Moving Avg)	**0.93**	82.4 %	89.7 %	42.4%
Ours (Leaky Bucket)	**0.93**	**84.0%**	**90.0%**	**44.3%**

Anomaly Score and AUROC Computation. An episode consists of multiple *planning calls*. We compute an anomaly score *per planning call*. For the j-th planning call at state $s^{(j)}$, we run GeCO optimization for at most $K_{\max} = 10$ steps. With the update $a_{k+1} \leftarrow a_k - \eta\, f_\theta(a_k, s^{(j)})$, we define

$$\text{score}\big(\text{plan } j\big) \;=\; \big\| f_\theta(a_K^{(j)}, s^{(j)}) \big\|_2,$$

where $K \le K_{\max}$ denotes the last executed refinement step (if early termination is triggered, K is the stopping step; otherwise $K = K_{\max}$). We sample 500 in-distribution (ID) episodes from LIBERO-Goal and 500 OOD episodes from LIBERO-Spatial. To avoid overweighting longer episodes, we compute AUROC on a fixed number of planning calls per episode, treating OOD plans as positives and ID plans as negatives.

Temporal Filtering and Early Reporting. Within each planning call, raw per-step norms can be noisy, so we evaluate two lightweight filters over the refinement-step residual norms $r_k = \|f_\theta(a_k, s^{(j)})\|_2$. Both filters output an *online* OOD flag for the current planning call; once the flag triggers (at any planning call), we **immediately terminate the episode** to avoid executing the remaining (eventually failing) interactions.

- **Moving Average.** We smooth r_k by averaging over the last $w = 5$ refinement steps, $\tilde{r}_k = \frac{1}{w}\sum_{i=k-w+1}^{k} r_i$, and trigger an OOD flag when $\tilde{r}_k$ exceeds a threshold.
- **Leaky Bucket.** We maintain an accumulator that integrates only sustained large residuals: $b_k \leftarrow \max\{0,\ (1-\lambda)b_{k-1} + (r_k - \tau)_+\}$, where $(x)_+ = \max(x, 0)$. We raise an OOD flag if $b_k \ge B$ at any step.

Quantitative Results. Table 5 reports separability and operating-point performance for distinguishing ID (LIBERO-Goal) from OOD (LIBERO-Spatial). We treat OOD plans as positives and ID plans as negatives; thus **TPR** measures the fraction of OOD instances correctly flagged, while **TNR** measures the fraction of ID instances correctly retained (i.e., not falsely flagged). The baseline loss signal is less aligned with the suite-level shift, whereas GeCO yields strong separation. We choose the detection threshold to target **TPR** $\approx 90\%$ for our detector when possible, and report the corresponding TNR.

Time Saved measures the fraction of interaction time avoided on OOD episodes by early reporting: for each OOD episode, we record the first reporting time t_{report} (the environment time step when the first planning call triggers the OOD flag) and normalize by the full episode duration T_{total} under the standard evaluation protocol, Time Saved $= 1 - t_{\text{report}}/T_{\text{total}}$; we then average this quantity over OOD episodes.

4.4 Ablation Study: Effect of Truncation Parameter α

To evaluate the sensitivity of our adaptive optimization dynamics to the truncation schedule, we conduct an ablation study on the truncation parameter α. As discussed in Section 3.2, α controls the onset of the velocity rescaling decay, directly impacting the equilibrium convergence. Table 6 presents the success rates on the LIBERO benchmark across different values of $\alpha \in \{0.6, 0.8, 0.9\}$.

The results highlight a clear trade-off governed by the smoothness of the learned vector field. When α is too small (e.g., $\alpha = 0.6$), the field becomes overly smooth. While this provides stable gradient

directions, it drastically slows down convergence. The optimization often fails to reach the tight equilibrium manifold within a limited step budget. Conversely, an excessively large α (e.g., $\alpha = 0.9$) leads to a large Lipschitz constant (L-smoothness) for the velocity field. This creates a sharp and stiff optimization landscape, making it difficult for the iterative solver to converge stably, thus degrading overall performance. Setting $\alpha = 0.8$ strikes the optimal balance, ensuring the attractors are sharp enough for precise control while maintaining a smooth enough field for rapid and stable inference.

Table 6: **Ablation Study on Truncation Parameter α.** We report the success rates (%) on the LIBERO benchmark across different values of α to demonstrate its impact on policy performance.

α	Success Rate (%)				
	Goal	Spat.	Obj.	Long	Avg.
0.6	85.8	**98.0**	91.6	48.6	81.0
0.8	**96.4**	97.4	**98.6**	**92.4**	**96.2**
0.9	95.8	97.8	94.0	81.2	92.2

5 Conclusion

We introduced **GeCO**, a time-unconditional formulation of flow-matching policies that casts action generation as iterative optimization over a stationary velocity field, where expert behaviors form stable attractors via velocity rescaling. This removes the rigid dependence on fixed integration schedules and provides a plug-and-play replacement for continuous action heads in modern VLA systems. Experimentally, GeCO delivers a better efficiency–performance trade-off through state-dependent adaptive optimization, scales to flow matchig based VLA models, and yields an optimization-native safety signal: the final field norm serves as a training-free OOD score.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

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

Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, et al. Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*, 2025a.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b.

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.

Zibin Dong, Yifu Yuan, Jianye Hao, Fei Ni, Yi Ma, Pengyi Li, and Yan Zheng. Cleandiffuser: An easy-to-use modularized library for diffusion models in decision making. *Advances in Neural Information Processing Systems*, 37:86899–86926, 2024.

Zibin Dong, Yicheng Liu, Yinchuan Li, Hang Zhao, and Jianye Hao. Conditioning matters: Training diffusion policies is faster than you think. *arXiv preprint arXiv:2505.11123*, 2025.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

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

Victoria J Hodge, Colin Paterson, and Ibrahim Habli. Out-of-distribution detection for safety assurance of ai and autonomous systems. *arXiv preprint arXiv:2510.21254*, 2025.

Runhan Huang, Haldun Balim, Heng Yang, and Yilun Du. Flexible locomotion learning with diffusion model predictive control. *arXiv preprint arXiv:2510.04234*, 2025a.

Xiaoyu Huang, Takara Truong, Yunbo Zhang, Fangzhou Yu, Jean Pierre Sleiman, Jessica Hodgins, Koushil Sreenath, and Farbod Farshidian. Diffuse-cloc: Guided diffusion for physics-based character look-ahead control. *ACM Transactions on Graphics (TOG)*, 44(4):1–12, 2025b.

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. *arXiv preprint arXiv:2504.16054*, 2025.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Sung-Wook Lee, Xuhui Kang, and Yen-Ling Kuo. Diff-dagger: Uncertainty estimation with diffusion policy for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4845–4852. IEEE, 2025.

Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, et al. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023b.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024b.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *arXiv preprint arXiv:2512.06963*, 2025.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Qiao Sun, Zhicheng Jiang, Hanhong Zhao, and Kaiming He. Is noise conditioning necessary for denoising generative models? *arXiv preprint arXiv:2502.13129*, 2025.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

Runqian Wang and Yilun Du. Equilibrium matching: Generative modeling with implicit energy-based models. *arXiv preprint arXiv:2510.02300*, 2025.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3836–3847, 2023.

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

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Guangyao Zhou, Sivaramakrishnan Swaminathan, Rajkumar Vasudeva Raju, J Swaroop Guntupalli, Wolfgang Lehrach, Joseph Ortiz, Antoine Dedieu, Miguel Lázaro-Gredilla, and Kevin Murphy. Diffusion model predictive control. *arXiv preprint arXiv:2410.05364*, 2024.

A Implementation Details for Adaptive Inference and Training

A.1 Codebase and Baselines

All experiments are implemented on top of the CleanDiffuser (Dong et al., 2024) codebase. We use the same network backbone as a Continuous Rectified Flow policy (Liu et al., 2022), but differ in the learning objective and the test-time inference algorithm.

A.2 Model Architecture

We follow the Rectified Flow architecture with a 1D diffusion-style transformer and a frozen vision-language conditioner.

Vision-language condition. We use a frozen DINOv2 (Oquab et al., 2023) vision encoder and a T5 (Raffel et al., 2020) text encoder. The T5 hidden dimension is set by the pretrained configuration:

`t5_hidden_dim` ← `T5Config.from_pretrained(t5_model).d_model`.

The conditioner is instantiated as:

- `ViTAndT5VisionLanguageCondition` with `emb_dim=768`, `freeze=True`, `To=1`, and `n_views=2`.

Policy backbone. The action generator uses a DiT-style 1D transformer with cross-attention conditioning:

- `DiT1dWithACICrossAttention` with `x_dim=act_dim`, `x_seq_len=Ta(=16)`, `emb_dim=768`, `d_model=384`, `n_heads=6`, `depth=12`.

GeCO uses the same backbone modules for fair comparison, while changing the training objective and inference procedure.

A.3 Adaptive Inference Setup

Sampler. We use a standard gradient-descent (GD) sampler with a maximum of $K_{\max} = 30$ refinement steps. Denoting the refinement direction by $f_\theta(a_k, s)$, the update is:

$$a_{k+1} \leftarrow a_k - \eta_k \, f_\theta(a_k, s), \quad k = 0, \ldots, K_{\max} - 1.$$

Step-size schedule. We use a fixed per-step learning-rate schedule $\{\eta_k\}_{k=1}^{30}$:

$$\eta_1 = 0.1, \quad \eta_{2..4} = 0.05, \quad \eta_{5..16} = 0.02, \quad \eta_{17..30} = 0.01.$$

We enable adaptive early stopping using a norm-based stopper: we terminate refinement once

$$\|f_\theta(a_k, s)\|_2 < 0.4.$$

A.4 Training Recipe and Hyperparameters

Pretrained encoders.

- T5: `google-t5/t5-base`
- ViT: `facebook/dinov2-base`

Training details. We use a truncated linear decay for equilibrium field learning

$$c(\gamma) = \begin{cases} \lambda, & \gamma \le a_0, \\ \lambda \dfrac{1-\gamma}{1-a_0}, & \gamma > a_0, \end{cases} \tag{8}$$

with scale $\lambda = 4$ and onset $a_0 = 0.1$.

Constants.

- Observation horizon: $T_o = 1$
- Action horizon: $T_a = 16$
- Executed actions per step: `num_act_exec=8`
- Normalization parameters: `norm_params=[0.5, 0.5, 0.5]`

B BENCHMARKS

LIBERO. We evaluate on the LIBERO simulation benchmark (Liu et al., 2023a), which contains four task suites—*Goal*, *Spatial*, *Object*, and *Long*—designed to isolate different distribution shifts in language-conditioned manipulation. *Goal* varies task goals while keeping objects and layouts fixed; *Spatial* introduces unseen spatial configurations; *Object* shifts object categories; and *Long* consists of long-horizon, compositional tasks. For policy learning and evaluation, our model predicts action chunks of length $T_a = 16$; after each prediction, we execute the first 8 actions (`num_act_exec`=8) in closed loop before replanning the next chunk. The observation space consists of two-view RGB images at the current time step with $T_o = 1$ (no history), and the language instruction is encoded by a frozen T5 encoder (`google-t5/t5-base`). Unless otherwise specified, we train each suite-specific policy for 30k optimization steps and report success rates averaged over 50 rollouts, following the standard evaluation protocol of LIBERO.

RoboTwin 2.0. We additionally benchmark on RoboTwin 2.0 (Chen et al., 2025b), a bimanual manipulation benchmark that evaluates robustness under controlled domain shifts. RoboTwin 2.0 reports performance under two difficulty settings: *Easy* uses clean environments, while *Hard* introduces domain-randomized perturbations such as clutter distractors, texture changes, lighting variations, and tabletop-height shifts. We evaluate on five representative tasks (*Adjust Bottle*, *Beat Block Hammer*, *Blocks Ranking (Size)*, *Click Alarmclock*, and *Click Bell*) and report Easy/Hard success and their averages (Table 2). All methods are evaluated with 100 rollouts per setting, using the official success criteria of RoboTwin 2.0. We highlight that the Hard setting creates a realistic robustness test by perturbing both visual appearance and scene dynamics while keeping task semantics unchanged.

VLABench. We further evaluate on VLABench (Zhang et al., 2025), a large-scale benchmark for language-conditioned manipulation that emphasizes long-horizon reasoning and semantic grounding. VLABench organizes tasks into multiple tracks covering in-distribution execution, cross-category generalization, common-sense/world-knowledge transfer, semantic instruction understanding, and unseen texture generalization. We report track-wise success rates on Tracks and their average (Table 3), following the benchmark's standard evaluation protocol with 50 rollouts. We include per-task breakdown for the 10-task primitive subset in Table 7, which complements the track-level summary in Table 3.

C VLA TRAINING DETAILS.

We fine-tune π_0 / $\pi_{0.5}$ VLAs using the OpenPI training stack with LeRobot-format datasets. Across all benchmarks, we train for **30k** gradient steps . Unless otherwise specified, weights are initialized from the corresponding OpenPI base checkpoint.

RoboTwin 2.0. For RoboTwin 2.0, we fine-tune a separate π_0 model per task. Observations include three RGB views (`cam_high`, `cam_left_wrist`, `cam_right_wrist`) and robot proprioception (`observation.state`). We train for 30k steps with batch size 32 for each task.

LIBERO. For π_0 on LIBERO, we fine-tune on `physical-intelligence/libero` datasets. We initialize from `gs://openpi-assets/checkpoints/pi0_base/params` and train for 30k steps with batch size 32.

For $\pi_{0.5}$ on LIBERO, we fine-tune for 30k steps with batch size 256. We use AdamW with gradient clipping (`clip_gradient_norm=1.0`) and an EMA of model weights (`ema_decay=0.999`).

Table 7: Per-task success rates on VLABench primitive 10-task evaluation. We report success rates for the baseline π_0 and π_0 with GeCO (ours) across five tracks; higher is better. Track averages correspond to the averages over the 10 tasks in each track.

Task	Track 1		Track 2		Track 3		Track 4		Track 6	
	π_0	Ours	π_0	Ours	π_0	Ours	π_0	Ours	π_0	Ours
add_condiment	0.66	0.80	0.14	0.08	0.34	0.16	0.26	0.10	0.56	0.76
insert_flower	0.18	0.30	0.04	0.00	0.22	0.28	0.02	0.00	0.10	0.02
select_book	0.694	0.84	0.064	0.08	0.417	0.51	0.311	0.23	0.714	0.79
select_chemistry_tube	0.52	0.76	0.12	0.14	0.70	0.94	0.06	0.02	0.28	0.71
select_drink	0.52	0.78	0.224	0.30	0.08	0.10	0.10	0.04	0.44	0.66
select_fruit	0.38	0.56	0.46	0.52	0.083	0.14	0.06	0.18	0.30	0.32
select_mahjong	0.25	0.58	0.02	0.04	0.125	0.12	0.12	0.06	0.02	0.25
select_painting	0.46	0.22	0.26	0.12	0.50	0.60	0.56	0.76	0.30	0.18
select_poker	0.54	0.70	0.26	0.54	0.06	0.22	0.12	0.52	0.28	0.52
select_toy	0.50	0.52	0.36	0.34	0.38	0.32	0.12	0.10	0.18	0.28
Avg_SR	0.47	0.61	0.212	0.22	0.291	0.34	0.173	0.20	0.322	0.45

The learning rate follows a cosine schedule with warmup (`warmup_steps=10,000`, `peak_lr=5e-5`).

VLABench. For the VLABench fine-tuning experiment, we choose to fine-tune π_0 on huggingface datasets `vlabench/vlabench_primitive_ft_lerobot`. We trained for 30k steps with batch size 256.

D OOD Detection Settings

ID/OOD split. We study a controlled suite-level distribution shift on LIBERO by treating *LIBERO-Goal* as in-distribution (ID) and *LIBERO-Spatial* as out-of-distribution (OOD). We train a single policy on the 10 tasks of LIBERO-Goal and evaluate OOD detection on the 10 tasks of LIBERO-Spatial.

Episode and planning-call sampling. We sample 500 ID episodes from LIBERO-Goal and 500 OOD episodes from LIBERO-Spatial. Each episode consists of multiple planning calls, where the policy generates an action chunk and executes a fixed number of actions before replanning. To avoid overweighting longer episodes, we compute planning-level statistics using a fixed number of planning calls per episode: we extract $J = 20$ planning calls from each episode using random J calls and treat each extracted planning call as one evaluation instance.

GeCO score (optimization-dynamics signal). For each planning call at state $s^{(j)}$, GeCO performs iterative optimization for at most $K_{\max} = 10$ steps and produces per-step norms $r_k = \|f_\theta(a_k, s^{(j)})\|_2$. We use the final-step norm as the raw anomaly score for that planning call:

$$\text{score}(\text{plan } j) = \|f_\theta(a_K^{(j)}, s^{(j)})\|_2,$$

where $K \le K_{\max}$ is the last executed refinement step (early termination uses the stopping step; otherwise $K = K_{\max}$).

Temporal filters and triggering rule. We evaluate two online filters over $\{r_k\}_{k=1}^{K}$ within each planning call: (i) **Moving Average** with window $w = 5$, triggering when $\tilde{r}_k > \tau_{\text{ma}}$; (ii) **Leaky Bucket** defined by $b_k \leftarrow \max\{0, (1-\lambda)b_{k-1} + (r_k - \tau)_+\}$, triggering when $b_k \ge B$. We set $(\tau_{\text{ma}}, \tau, \lambda, B) =$ `(0.6, 0.5, 0, 0.7)` and keep them fixed across all experiments.

Baseline score. As a baseline OOD signal, we follow the Diff-DAgger-style idea (Lee et al., 2025) of using the policy's training objective as an uncertainty indicator. For each planning call, let x_0 denote the generated action chunk and let c denote the conditioning input. We draw a single Gaussian

noise sample $\epsilon \sim \mathcal{N}(0, I)$ and construct a noised input at timestep t via linear interpolation

$$x_t \;=\; (1-t)\, x_0 + t\, \epsilon,$$

where T is the total number of diffusion steps used by the training objective. We then evaluate the flow-matching regression loss

$$\mathcal{L}_{\mathrm{FM}}(x_0, c) = \mathbb{E}\, \| f_\theta(x_t, t, c) - (\epsilon - x_0) \|_2^2$$

AUROC computation. We compute planning-level AUROC by sweeping a threshold over the scalar score to separate OOD planning calls (positive) from ID planning calls (negative), using the $500 \times J$ ID instances and $500 \times J$ OOD instances.

Operating point (TNR/TPR). We report TNR/TPR at a single operating point selected to achieve **TPR** $\approx 90\%$ on OOD instances for our detector when possible. Specifically, we choose the threshold θ such that $\mathrm{TPR}(\theta)$ is closest to 0.9 and report the corresponding TNR.

Early reporting and Time Saved. The detector runs online during an episode. Once any planning call triggers an OOD flag, we immediately terminate the episode. We define the reporting time t_{report} as the environment time step when the first trigger occurs, and compute

$$\text{Time Saved} = 1 - \frac{t_{\mathrm{report}}}{T_{\mathrm{total}}},$$

where $T_{\mathrm{total}} = 300$ is the episode horizon under the standard evaluation protocol. We report Time Saved averaged over OOD episodes only.