Generative Control as Optimization: Time Unconditional Flow Matching for Adaptive and Robust Robotic Control

Zunzhe Zhang[*,1,3] **Runhan Huang**[*,1] **Yicheng Liu**[1,3] **Shaoting Zhu**[1]
Linzhan Mou[2] **Hang Zhao**[†,1,3]

[1]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
[2]Department of Computer Science, Princeton University, New Jersey, US
[3]Galaxea Inc., Beijing, China

[*]Equal contribution. Author order was determined randomly.
[†]Corresponding author: hangzhao@mail.tsinghua.edu.cn

Abstract

Diffusion and flow-based generators are increasingly used in robot policies to produce action chunks from visual observations, proprioception, and language instructions. However, closed-loop robot control requires solving action-generation problems of varying difficulty, whereas standard diffusion and flow-matching inference typically follows a fixed integration schedule. We introduce **Generative Control as Optimization (GeCO)**, a time-unconditional framework that turns action synthesis from fixed-time integration into iterative optimization. GeCO learns a stationary velocity field over action sequences, and refines actions until the field norm becomes small. This enables adaptive computation at each planning call: simple states can terminate early, while difficult states can use additional refinement. The same field norm also provides a lightweight signal for non-convergent action generation under distribution shift. GeCO can be instantiated in both diffusion-transformer policies and flow-matching Vision-Language-Action (VLA) systems without changing the surrounding policy interface. Across simulation benchmarks and real-world tasks, GeCO matches or improves task performance while enabling convergence-based inference as a plug-and-play replacement for standard time-conditioned action generators. Code and videos can be found at https://hrh6666.github.io/GeCO/.

1 Introduction

Generative modeling has rapidly established itself as a cornerstone of modern robotic imitation learning [4, 9, 21, 19, 18, 46, 23, 6]. By representing policies as conditional distributions rather than deterministic mappings, diffusion models [16, 40, 47] and flow matching [29, 32] can capture the multi-modal nature of human behavior and iteratively refine noisy action chunks into feasible robot behaviors. These methods have demonstrated strong performance across manipulation and locomotion tasks [9, 20, 19, 46, 1] and diverse robotic benchmarks [8, 30, 48]. In compact visuomotor policies, such generators may be implemented as diffusion-transformer policies [9]; in Vision-Language-Action (VLA) systems, they often appear as flow-matching action heads attached to vision-language backbones [4, 20]. However, once deployed inside a closed-loop robot controller, the difficulty of action generation is highly state-dependent. Free-space transit may require only coarse refinement, while pre-grasp alignment, contact-rich insertion, and visually grounded object selection often demand substantially more computation.

This deployment setting exposes a mismatch in standard time-conditioned generative policies. Diffusion and flow-matching controllers typically learn dynamic vector fields that evolve according to a

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

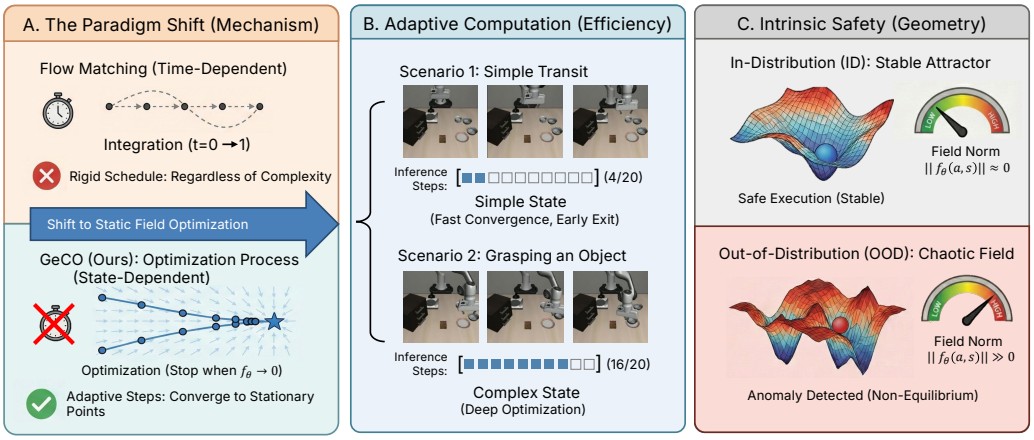


Figure 1: Generative Control as Optimization (GeCO). (A) From fixed integration to action-space optimization: Standard flow-matching controllers rely on time-dependent integration schedules, whereas GeCO learns a stationary observation-conditioned field over action sequences and generates actions through iterative optimization. **(B) Adaptive Computation:** The stationary action field provides a convergence criterion, allowing the policy to exit early during simple transit phases and refine longer for contact-rich or semantically ambiguous states. **(C) Deployment-Time Anomaly Detection:** 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.

prescribed generative time schedule [16, 29]. While this schedule is effective for bridging Gaussian noise and the expert data distribution, it assigns computation through a fixed or externally chosen number of function evaluations at each planning call. Such a schedule does not directly reflect whether the current robot action sequence has converged to a valid solution. As a result, the policy may waste computation on simple states while lacking a mechanism to allocate additional refinement to difficult ones. More importantly, fixed-schedule integration produces no intrinsic per-call diagnostic of whether the generated action lies near an in-distribution behavior; obtaining such a signal typically requires extra forward passes, ensembles, or auxiliary networks [17].

We propose **Generative Control as Optimization (GeCO)**, a time-unconditional framework for adaptive continuous action generation in robot policies. Building on stationary, time-unconditional generative dynamics [43, 41], GeCO learns a stationary observation-conditioned field over action sequences and generates actions by optimizing this field until convergence. This reframes action synthesis as convergence-driven optimization in the action space, decoupled from the fixed integration horizon of standard generative-time schedules. The resulting field norm serves two deployment-facing roles in closed-loop control: its convergence governs adaptive inference latency at each planning call, while its residual magnitude indicates non-convergent action generation under distribution shift.

GeCO can be instantiated across different generative robot-policy architectures. For diffusion-transformer [35] policies, it replaces the standard time-conditioned denoising or flow-based action generator with a stationary action-sequence field [9–11]. For flow-matching-based VLA systems, it serves as a plug-and-play continuous action decoder while preserving the vision-language backbone, multimodal fusion pipeline, action representation, and robot-control interface [4, 20]. Building on this architectural flexibility, we evaluate GeCO on standard simulation benchmarks [8, 30, 48] and real-world manipulation tasks. Empirically, GeCO improves the success–latency trade-off by allocating computation according to robot-state difficulty, while its field norm provides a practical signal for detecting distribution-shifted execution states.

Our contributions are summarized as follows:

- **Time-unconditional action generation.** We formulate continuous action generation for receding-horizon robot control as optimization over an observation-conditioned stationary field, rather than fixed integration along a generative-time schedule.
- **Convergence-aware adaptive optimization.** We develop an adaptive inference procedure that allocates computation at each robot planning call, refining longer for contact-rich,

alignment-sensitive, or semantically ambiguous states while exiting early during simple free-space motion.

- **Policy instantiations and deployment validation.** We instantiate GeCO in both diffusion-transformer policies and flow-matching VLA systems, and evaluate it on simulation and real-world manipulation, showing improved success–latency trade-offs and intrinsic monitoring under distribution shift. Code and videos can be found at `https://hrh6666.github.io/GeCO/`.

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) [3, 2, 31] for high-level planning [13, 28, 42] and action generation [23, 24, 6]. Parallel efforts utilize video generation models [33, 45, 44] to serve as predictive priors for robotic decision-making [14, 7, 38]. In the realm of precise robotic control, diffusion [16, 40] and flow matching [29] have demonstrated exceptional effectiveness, functioning as robust planners and action-chunk generators [9, 18, 50, 19, 21, 1, 11, 46]. Notably, the integration of continuous flow-matching heads into VLA architectures has emerged as a powerful paradigm [4, 20, 27, 39, 22]; 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 purely autoregressive, discretized action decoding [23, 24], enabling higher fidelity control and improved real-time performance [5, 20]. 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 [16, 40, 29] 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 [41] 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 [29, 16]. Equilibrium Matching [43] addresses this by learning a stationary vector field compatible with an implicit energy landscape [25], enabling optimization-driven sampling [15]. GeCO brings this time-unconditional perspective to visuomotor action generation [9], where the model must produce temporally coherent action chunks that can be executed and replanned in a closed-loop robot controller. While equilibrium-style dynamics have shown promise in image generation [43], GeCO repurposes the stationary field as a closed-loop control primitive: its norm both regulates state-dependent inference latency and provides a deployment-time signal for non-convergent action generation.

3 Method

Our goal is to turn generative control for robotic manipulation from a fixed-time integration process into an optimization problem over a stationary velocity field. At each control step, the robot observes the current visual scene, proprioception, and language instruction, synthesizes a short-horizon action sequence, executes part of it, and replans from the next observation. We first revisit time-conditioned diffusion and flow matching for robot action generation, then introduce our time-unconditional field and optimization-based inference, and finally show how the same field norm provides a deployment-time anomaly signal.

3.1 Revisiting Time-Conditioned Generative Control

At control step t, the robot receives an observation $o_t = (\mathbf{v}_t, \mathbf{p}_t, l)$, where $\mathbf{v}_t$ denotes visual observations, $\mathbf{p}_t$ denotes proprioception, and l denotes the language instruction. A policy encoder maps this observation to a conditioning representation $z_t = \mathrm{Enc}_\psi(o_t)$. The encoder Enc_ψ may be a compact visuomotor encoder or a VLA/VLM backbone. The policy then predicts a sequence of future low-level actions $a_t = [u_t, \ldots, u_{t+T_a-1}] \in \mathcal{A}^{T_a}$, where T_a is the action horizon.

Diffusion- and flow-based controllers typically learn a *time-conditioned* velocity field $v_\theta(x_\gamma, \gamma, z_t)$, where x_γ is a noisy or partially denoised action sequence and γ is a generative time variable. 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 or 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 $v_\theta(\cdot, \gamma, z_t)$ is tied to a specific notion of progress along the generative-time axis.

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 field whose norm directly indicates whether a final action chunk has settled for the current observation and instruction. GeCO decouples action refinement from this explicit time variable: instead of conditioning on γ, we learn a stationary field over action sequences and use it for test-time optimization.

3.2 Time-Unconditional Velocity Fields for Control

GeCO builds on the time-unconditional training principle of Equilibrium Matching [43], but applies it to visuomotor action generation, where the model produce temporally coherent action chunks that can be executed and replanned in a closed-loop robot controller. Rather than a time-indexed field $v_\theta(\cdot, \gamma, z_t)$, GeCO learns

$$f_\theta : \mathcal{A}^{T_a} \times \mathcal{Z} \to \mathcal{A}^{T_a}, \qquad (a, z_t) \mapsto f_\theta(a, z_t).$$

For a candidate action sequence a, the vector $f_\theta(a, z_t)$ gives the update direction for refining the current action proposal under the current visual observation, proprioception, and language instruction.

To train such a field, we introduce γ only as a hidden training variable. Given a demonstration action sequence a and Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$, we sample $\gamma \sim \mathcal{U}(0, 1)$ and form the interpolated action sequence $x_\gamma = \gamma a + (1 - \gamma)\varepsilon$, without providing γ to the model. At each x_γ, we define a velocity-rescaled target

$$g^\star(a, \varepsilon, \gamma) = (\varepsilon - a)c(\gamma),$$

where $c(\gamma)$ decays to zero as $\gamma \to 1$. Because $c(1) = 0$, the target field vanishes near demonstrated action sequences. This is the key property that makes the field norm usable as a stopping signal: action sequences close to demonstrations are trained to have small field norm, so convergence of the field corresponds to reaching a demonstration-consistent solution. The training objective is

$$\mathcal{L}(\theta, \psi) = \mathbb{E}_{(o_t, a)} \mathbb{E}_{\varepsilon, \gamma} \left[\left\| f_\theta\big(x_\gamma, \mathrm{Enc}_\psi(o_t)\big) - g^\star(a, \varepsilon, \gamma) \right\|^2 \right].$$

Here γ controls only the interpolation and the target scale through $c(\gamma)$; since it never enters the model, $f_\theta(\cdot, z_t)$ must organize all interpolation levels into a single stationary field for each observation and instruction.

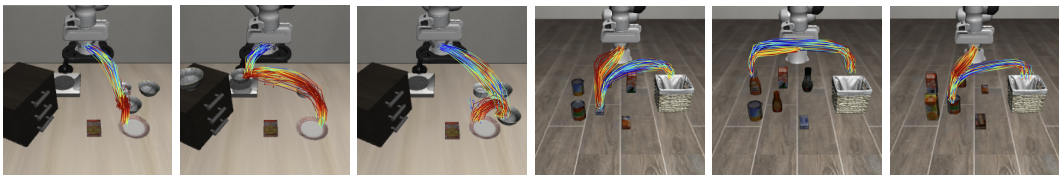
3.3 Optimization-Based Inference

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 control step, it computes $z_t = \mathrm{Enc}_\psi(o_t)$ and optimizes an action sequence using the stationary field $f_\theta(\cdot, z_t)$. We initialize $a_t^{(0)}$ from a prior $p_0(\cdot \mid z_t)$ and perform iterative updates

$$a_t^{(k+1)} = a_t^{(k)} - \eta_k f_\theta(a_t^{(k)}, z_t).$$

Because there is no time index, there is no requirement that the effective total step length match a prescribed integration interval. Taking more or fewer steps, or even using a total step length larger than one, remains optimization in the same stationary field rather than extrapolation beyond a trained time domain.

Convergence is monitored through the field norm $\rho_t^{(k)} = \|f_\theta(a_t^{(k)}, z_t)\|_2$, and optimization stops when $\rho_t^{(k)} \le \tau_{\mathrm{opt}}$ or $k \ge K_{\max}$. The final number of function evaluations K_t becomes the



(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.

computation used by the current planning call. After optimization, the robot executes the first H actions from the optimized chunk and replans from the next observation. Thus GeCO behaves as an optimization-based, receding-horizon controller grounded in a learned generative field. The complete optimization procedure is provided in Algorithm 1 in Appendix B. Simple scenes (e.g., unobstructed reaches) reach a low-norm region in few iterations; cluttered or contact-rich scenes require more.

GeCO can be instantiated without changing the surrounding robot-policy interface. For diffusion-transformer policies, Enc_ψ is the visuomotor encoder and GeCO replaces the standard time-conditioned denoising or flow-based action generator [9–11]. For flow-matching-based VLA systems, Enc_ψ is the VLM/VLA backbone and GeCO replaces the continuous action decoder while preserving the multimodal fusion pipeline, action representation, and control API [4, 20, 22, 39].

3.4 Field Geometry for Deployment-Time Anomaly Detection

GeCO also provides a simple anomaly signal during deployment. Robot failures often appear as planning calls that do not settle under the current observation, for example due to visual ambiguity, semantic grounding errors, object-layout shifts, contact mismatch, or unseen physical configurations. Because GeCO already optimizes until the field norm becomes small, the final field norm directly measures whether the current action-generation process has converged.

Given a test observation o_t, we run Algorithm 1, obtain the optimized action chunk $\hat{a}_t$ and final field norm ρ_t, and define $\mathrm{Score}(o_t) = \rho_t = \|f_\theta(\hat{a}_t, \mathrm{Enc}_\psi(o_t))\|_2$. For in-distribution robot states, optimization tends to reach regions that behave like high-γ samples, where $c(\gamma)$ is small and the learned field nearly vanishes. For anomalous observations, the induced field is not guaranteed to lead to a low-norm action sequence, leaving a larger final score. We obtain a binary decision via $\mathrm{isOOD}(o_t) = \mathbf{1}\{\mathrm{Score}(o_t) > \tau_{\mathrm{OOD}}\}$. This incurs essentially no overhead, since the same field norm is already computed for convergence checking during action optimization.

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) **Anomaly Detection**, 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 [30] 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

Table 1: Performance vs. Efficiency on LIBERO. GeCO improves success by adaptive refinement while keeping the average NFE low.

Method	Max Steps	Success Rate (%) Goal	Spat.	Obj.	Long	Avg.	Efficiency **NFE**
Diff. Policy	100	82.9	91.4	88.9	82.7	86.5	100.0
Rectified Flow	20	92.4	94.6	97.0	76.0	90.0	20.0
GeCO	5	91.6	95.4	98.2	82.4	91.9	**5.0**
GeCO	10	93.0	95.8	99.0	81.8	92.4	8.7
GeCO	20	**95.2**	**95.8**	**99.0**	83.8	**93.5**	11.6
GeCO	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
ACT [49]	**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 [4]	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**

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.

Training-seed variation and additional baselines. Across five independent training runs, GeCO obtains $92.30\% \pm 1.02\%$ LIBERO success, compared with $89.12\% \pm 0.80\%$ for three 20-step fixed-RF runs (mean $\pm$ sample standard deviation). Each run is evaluated on 40 tasks and 2,000 episodes. Additional inference-budget comparisons show that GeCO obtains 91.90% success at 5.00 NFE, versus 90.25% for fixed RF at the same NFE and 89.60% for AdaFlow at 4.46 NFE. At a larger budget, GeCO obtains 93.50% at 11.60 NFE, compared with AdaFlow's 88.70% at 12.60 NFE. Full results and the AdaFlow training setup are provided in Appendix O.1–O.2.

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 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. A critical advantage of GeCO is its architectural compatibility with existing time-conditional backbones. Since π_0 is originally trained as

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**	**36.4**

a time-dependent flow matching model expecting a time embedding $t \in [0, 1]$, we adapt it to our time-unconditional framework by simply fixing the input time $t = 0$ during fine-tuning. This effectively freezes the time channel, forcing the network to learn a single, stationary velocity field $f_\theta(x, s)$ using the pre-trained weights, rather than a sequence of time-varying fields. This plug-and-play adaptation integrates the rich visual-semantic representations of foundation models with the native adaptive optimization inference of GeCO, with no need for structural modifications or training from scratch. 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 C.

VLABench Result We benchmark our method on VLABench [48], 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 [8] 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 variations. 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.

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 Diff-DAgger [26], we use the diffusion training objective as a baseline uncertainty signal. The original formulation estimates the expected denoising loss by averaging over many sampled timesteps (e.g., 512 samples), which introduces hundreds of additional forward passes per planning call and severely slows deployment. For a practical comparison, we use a single-step proxy that evaluates the flow-matching loss at a fixed timestep $t = 1$. In contrast, GeCO

Table 4: OOD Detection on LIBERO Suite Shift. We report planning-level AUROC, operating-point detection rates, and time saved under Goal → Spatial shift.

Method	AUROC ↑	TPR(%)	TNR(%)	Time Saved(%)
Diff-DAgger (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**	**82.6**	**90.5**	**43.2**

derives its anomaly score from the final field norm computed during action optimization, requiring no timestep sampling and at most one additional field evaluation. More details of OOD baseline experiments can be found in Appendix. M.

Anomaly Score and AUROC Computation. An episode consists of multiple *planning calls* (each call runs optimization to produce an action chunk that is then executed in the environment). 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 4 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 **TNR** $\approx 90\%$ for our detector when possible, and report the corresponding TPR. **Time Saved** measures the fraction of OOD episode execution avoided after the first online OOD trigger, computed as $1 - t_{\text{report}}/T_{\text{total}}$; the results show that GeCO detects OOD states earlier and avoids over 40% of unnecessary OOD execution.

Additional visual shifts. We additionally evaluate 50% pixel replacement and a wrist-camera extrinsic shift, using 250 held-out ID episodes for calibration and paired sets of 250 ID and 250 OOD episodes for evaluation. GeCO achieves AUPRC of 0.755 and 0.648, compared with 0.749 and 0.607 for Diff-DAgger at $B = 1$. The measured FPRs are 8.0% for GeCO and 7.6% for this baseline on both shifts. Appendix O.4 provides perturbation settings and full results.

4.4 Ablation Study

Isolating adaptive stopping. In a paired 2,000-episode evaluation, we fix the checkpoint, update schedule, maximum budget of 20 evaluations, task and initial-state IDs, and random seeds, and switch only the residual-based stopper. Enabling stopping changes success from 91.8% to 92.5%, while reducing average NFE from 20.00 to 10.85 (45.8%); see Appendix O.3.

We provide a detailed ablation study in Appendix H, covering the truncation parameter α, the early-stop threshold τ, step-size schedules, and OOD-detection hyperparameters. Overall, the results show that GeCO is robust to a broad range of hyperparameter choices: truncation with $\alpha < 1$ consistently preserves the equilibrium behavior of the stationary field, while removing truncation substantially degrades performance; the early-stop threshold provides a smooth trade-off between success rate and inference cost; and the OOD detector remains effective under local variations of its filtering parameters. These findings indicate that GeCO's gains primarily come from the stationary-field formulation and convergence-aware inference, rather than fragile hyperparameter tuning.

4.5 Real-World Experiment

To validate the practical effectiveness and efficiency of GeCO in unstructured physical environments, we conduct real-world benchmarks using the Galaxea R1 Lite[22] mobile manipulator.

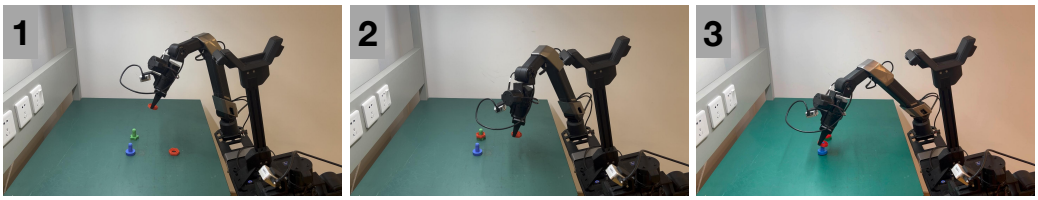


Figure 3: GeCO policy execution for the Nut Assembly task.

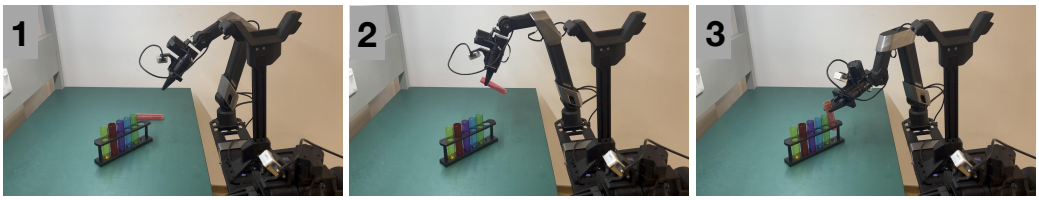


Figure 4: GeCO policy execution for the Chemistry Tube Arrangement task.

Experimental Setup. As shown in Fig.3 and Fig.4, we use **Galaxea G0 Plus** [22] as our baseline, which employs standard flow matching (FM) as VLA action head, and compare it against **G0 Plus with GeCO**. For both policies, the flow-matching heads are fine-tuned on a demonstration dataset collected via teleoperation. We select two complex and high-precision manipulation tasks for the benchmark: *Nut Assembly* and *Chemistry Tube Arrangement*. These tasks require millimeter-level precision, providing a rigorous testbed for GeCO's ability to perform deep refinement during contact-rich bottleneck phases while exiting early during simple transit motions. Each task is evaluated over 20 independent trials on the physical hardware. We control the same total gradient steps during training. Detailed experimental setup can be found in Appendix F.

Quantitative Results. As shown in Table 5, GeCO consistently outperforms the standard flow-matching baseline in success rate across both challenging real-world tasks, while also substantially reducing the average NFE and inference time. On *Nut Assembly*, GeCO improves the success rate from 30% to 70%, reduces the average NFE from 10.0 to 5.06, and lowers the average inference time from 252.3 ms to 172.4 ms (a 31.7% reduction). On *Tube Arrangement*, GeCO improves the success rate from 25% to 80%, reduces the average NFE from 10.0 to 3.82, and lowers the average inference time from 254.6 ms to 150.3 ms (a 41.0% reduction). While the baseline is constrained to a fixed schedule of 10 steps, GeCO's adaptive inference mechanism allows it to settle at equilibrium faster, thereby improving efficiency without compromising control fidelity (see Appendix G for detailed NFE analysis). We attribute this substantial performance margin to two key advantages: enhanced sample efficiency and superior generalization. First, given that both policies were fine-tuned using identical datasets and training budgets, GeCO's higher success rate clearly underscores the improved sample efficiency inherent to its time-unconditional formulation. Second, our method demonstrates remarkable robustness during physical deployment. Under domain-randomized evaluations, GeCO's optimization-based inference maintains robust convergence amidst real-world visual and physical variations, whereas the fixed-schedule baseline proves brittle to such unstructured domain shifts.

Table 5: Real-world performance on Galaxea R1 Lite. We report the success rate, average Number of Function Evaluations (NFE), and average inference time for each task.

Method	Nut Assembly			Tube Arrangement		
	Success (↑)	Avg. NFE (↓)	Avg. Infer. Time (↓)	Success (↑)	Avg. NFE (↓)	Avg. Infer. Time (↓)
G0 Plus	30%	10.0	252.3	25%	10.0	254.6
G0 Plus + GeCO (Ours)	**70%**	**5.06**	**172.4**	**80%**	**3.82**	**150.3**

5 Discussion and Limitations

GeCO reformulates generative action synthesis as a robust optimization task, and the proposed framework delivers promising performance alongside strong adaptivity and OOD awareness, which validates the effectiveness of our core design. While the framework performs well in practical settings, we acknowledge that its theoretical foundation can be further strengthened. Future work will focus on deriving more rigorous theoretical bounds for performance gains, the convergence speed of the optimization process, and the OOD detection mechanism in the context of control and VLA to further solidify GeCO's theoretical foundation.

6 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 matching based VLA models, and yields an optimization-native safety signal: the final field norm serves as a training-free OOD score.

Acknowledgements and Disclosures

We thank the Institute for Interdisciplinary Information Sciences (IIIS) at Tsinghua University and GalaXea Inc. for providing computing resources that supported this work. The authors declare no competing interests.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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*, 2025.

[8] 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*, 2025.

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

[10] 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.

[11] 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.

[12] J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980. ISSN 0377-0427. doi: https://doi.org/10.1016/0771-050X(80)90013-3. URL `https://www.sciencedirect.com/science/article/pii/0771050X80900133`.

[13] 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.

[14] 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.

[15] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022.

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

[17] 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.

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

[19] 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, 2025.

[20] 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.

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

[22] 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.

[23] 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.

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

[25] Yann LeCun, Sumit Chopra, Raia Hadsell, et al. A tutorial on energy-based learning. 2006.

[26] 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)*, pages 4845–4852. IEEE, 2025.

[27] 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.

[28] 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.

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

[30] 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, 2023.

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

[32] 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.

[33] 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*, 2024.

[34] 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.

[35] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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

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

[42] 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.

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

[44] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025.

[45] 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.

[46] 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.

[47] 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*, pages 3836–3847, 2023.

[48] 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*, pages 11142–11152, 2025.

[49] 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.

[50] 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 [10] codebase. We use the same network backbone as a Continuous Rectified Flow policy [32], 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 [34] vision encoder and a T5 [37] text encoder. The T5 hidden dimension is set by the pretrained configuration:

$$\texttt{t5_hidden_dim} \leftarrow \texttt{T5Config.from_pretrained(t5_model).d_model}.$$

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

$$c(\gamma) = \begin{cases} \lambda, & \gamma \le \alpha_0, \\ \lambda \dfrac{1-\gamma}{1-\alpha_0}, & \gamma > \alpha_0, \end{cases} \tag{1}$$

with scale $\lambda = 4$ and onset $\alpha_0 = 0.8$.

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 GeCO Inference Pseudocode

Algorithm 1 summarizes the deployment-time optimization procedure used by GeCO. At each planning call, the policy optimizes an action chunk under the stationary field and exits once the field norm falls below the convergence threshold.

C Benchmarks

LIBERO. We evaluate on the LIBERO simulation benchmark [30], 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.

Algorithm 1 GeCO Receding-Horizon Action Optimization

Input: Observation o_t, encoder Enc_ψ, action field f_θ
  max steps $K_{\max}$, tolerance τ_{opt}, step sizes $\{\eta_k\}$
Output: Optimized action chunk $\hat{a}_t$, final field norm ρ_t, NFE K_t
$z_t \leftarrow \mathrm{Enc}_\psi(o_t)$
$a_t^{(0)} \sim p_0(\cdot \mid z_t)$
for $k = 0, \ldots, K_{\max} - 1$ **do**
  $v_k \leftarrow f_\theta(a_t^{(k)}, z_t)$
  $\rho_k \leftarrow \|v_k\|_2$
  if $\rho_k \le \tau_{\mathrm{opt}}$ **then**
    break ▷ convergence-based early exit
  end if
  $a_t^{(k+1)} \leftarrow a_t^{(k)} - \eta_k v_k$
end for
$\hat{a}_t \leftarrow a_t^{(k)}, \quad \rho_t \leftarrow \rho_k, \quad K_t \leftarrow k + 1$
return $\hat{a}_t$, ρ_t, K_t

Table 6: 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

RoboTwin 2.0. We additionally benchmark on RoboTwin 2.0 [8], 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 [48], 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 6, which complements the track-level summary in Table 3.

D 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`). 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.

E 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 $(\tau_{\text{ma}}, \tau, \lambda, B) =$ `(0.5, 0.2, 0.7)` and keep them fixed across all experiments.

Baseline score (training-loss proxy). As a baseline OOD signal, we follow the Diff-DAgger-style idea [26] 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

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

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 **TNR** $\approx 90\%$ on ID instances for our detector when possible. Specifically, we choose the threshold θ such that $\mathrm{TNR}(\theta)$ is closest to 0.9 and report the corresponding TPR. All reported TNR and TPR are episode level metrics, instead of per-frame metric.

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.

F Real World Experiment Settings

Hardware Configuration. We conduct our real-world experiments using the **Galaxea R1 Lite** mobile manipulator. The robot is equipped with two dexterous arms and 3 RGB cameras. For our policy input, we utilize three camera views: a head camera (`cam_head`), and two wrist-mounted cameras (`cam_left_wrist`, `cam_right_wrist`). We also include the robot's proprioceptive state (`observation.state`), which contains joint positions and gripper states. The control frequency of the physical system is set to 15 Hz, and the execution action chunk size is set to $T_a = 16$.

Data Collection and Task Description. We select two contact-rich, high-precision manipulation tasks: *Nut Assembly* and *Chemistry Tube Arrangement*. The task setup is shown in Fig. 5.

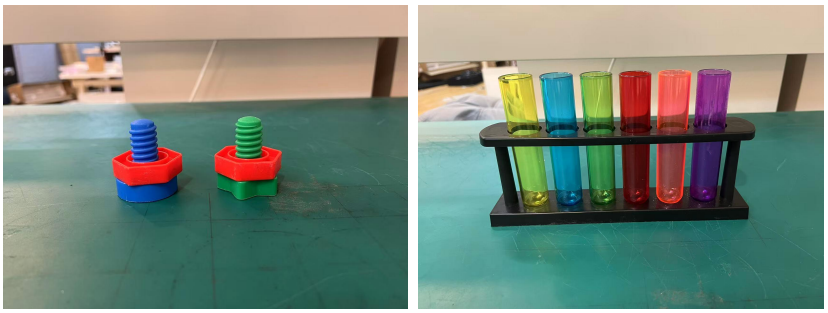


Figure 5: Task setups for the real-world robotic deployment, showing the configurations for both the Nut Assembly and the Chemistry Tube Arrangement tasks.

- **Nut Assembly:** The robot shall grasp a plastic nut, accurately align it with a threaded bolt, and perform the insertion. A trial is considered successful only when both two nuts are fully threaded onto their respective bolts.
- **Chemistry Tube Arrangement:** The robot shall grasp chemistry tubes from a randomized initial pose and insert them into designated tight-tolerance slots on a rack with random position and rotation. A successful trial requires the tube to be fully seated in the correct slot without dropping or tipping over.

Training Details. Our baseline policy is **Galaxea G0 Plus**, which employs a standard flow-matching-based continuous action head. To train our **G0 Plus with GeCO** policy, we initialize the model with the pre-trained G0 Plus weights. Following the GeCO formulation, we fix the time conditioning to $t = 0$ during fine-tuning to learn a stationary velocity field. We fine-tune the model on the expert demonstrations for 4 full epochs with a batch size of 32. We use the AdamW optimizer with a peak learning rate of `2.5e-5` and a cosine learning rate schedule with `warmup_steps=200`.

GeCO Inference Setup. During real-world deployment, the GeCO policy synthesizes action chunks via iterative optimization. We set the maximum number of optimization steps to $K_{\max} = 10$. We use a step-size schedule $\{\eta_k\}$ initialized at `0.25` and decaying to `0.02` for deeper refinement. The adaptive early stopping mechanism is triggered when the gradient norm falls below a threshold τ_{opt}:

$$\|f_\theta(a_k, s_t)\|_2 < \tau_{\text{opt}},$$

where τ_{opt} is empirically set to `1.5`. If the threshold is not met, the optimization continues until $K_{\max}$ is reached, ensuring complex states (e.g., insertion alignment) receive maximum computational budget while simple transits exit early.

Evaluation Protocol. We evaluate both the baseline (G0 Plus) and our method (G0 Plus + GeCO) over 20 independent physical trials for each task. The initial positions of the objects (nuts, bolts, tubes, racks) are randomized within the workspace to test robustness to spatial variations. We report the success rate and record the average Number of Function Evaluations (NFE) per planning call. For the baseline, inference is bound to a fixed schedule of NFE $= 10$. For GeCO, NFE varies dynamically based on the early stopping criteria.

G Detailed Analysis of Real-World Adaptive Computation

To provide a deeper understanding of how GeCO dynamically modulates its computation during physical deployment, we present a detailed analysis of both the Number of Function Evaluations (NFE) and the total inference time over representative real-world rollouts in Fig. 6.

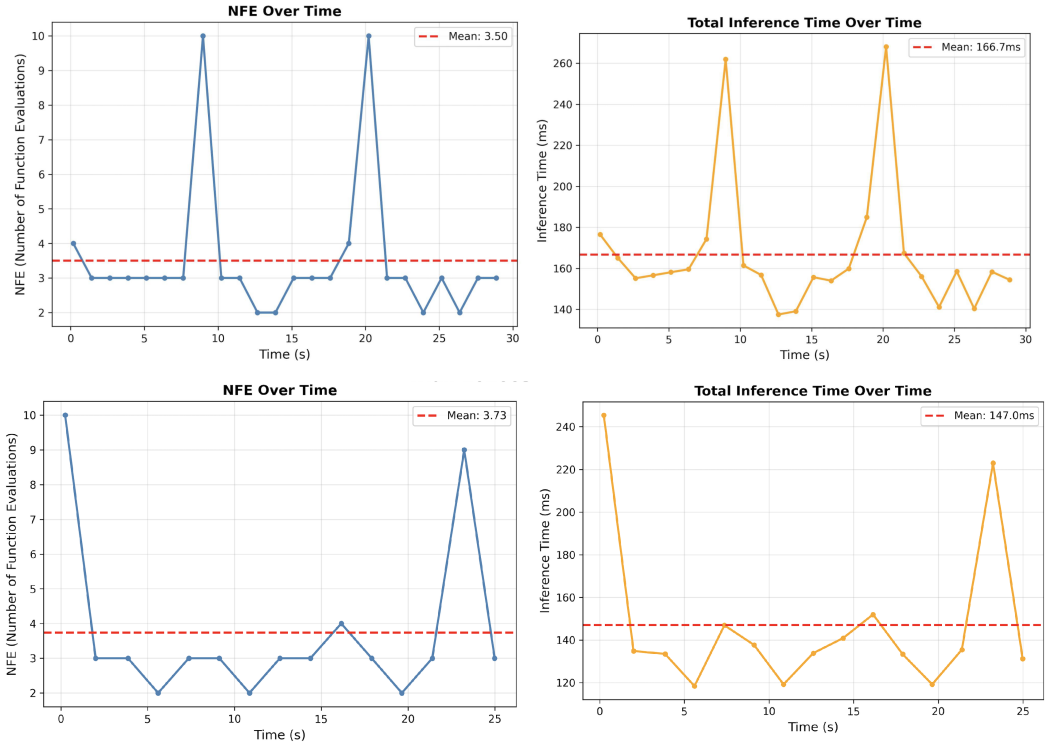


Figure 6: Adaptive computation during real-world execution. We visualize the per-step Number of Function Evaluations (NFE) and total inference time over time for representative physical rollouts of *Nut Assembly* and *Tube Arrangement*. For *Nut Assembly*, GeCO uses an average of 3.50 NFEs and 166.7 ms total inference time per decision. For *Tube Arrangement*, GeCO uses an average of 3.73 NFEs and 147.0 ms per decision. Rather than relying on a fixed computation budget, GeCO automatically increases computation during bottleneck phases, where the NFE rises to 9–10, and exits early during easier free-space motions, where the NFE drops to as low as 2–3. The corresponding inference-time traces mirror this behavior, showing that GeCO adaptively allocates computation according to task difficulty and thereby improves real-world inference efficiency.

As shown in Fig. 6, peaks in computation coincide with the most challenging stages of execution, such as fine alignment, grasping, and contact-rich insertion. In contrast, during simpler semantic stages and

free-space transit, the policy quickly converges and requires substantially fewer function evaluations. The total inference time follows the same overall trend as the NFE profile, confirming that GeCO's adaptive stopping behavior directly translates into lower online compute cost on physical hardware. These results further demonstrate that our time-unconditional flow matching formulation enables state-dependent computation in real-world manipulation while maintaining robust performance.

H Additional Ablation Studies

In this section, we provide additional ablation studies to further validate the design choices and hyperparameters of our proposed method.

H.1 Ablation on Stationary Fields

To evaluate the sensitivity of our adaptive optimization to the truncation schedule $c(\gamma)$, we ablate the parameter α. Crucially, removing truncation entirely ($\alpha = 1.0$) degrades the formulation into standard flow matching that merely discards the time conditioning. Without the velocity rescaling decay, the vector field loses its equilibrium property: vector magnitudes no longer decay towards zero near the optimal state, leaving the optimization without a natural termination mechanism. This instability directly leads to the worst empirical performance across all tasks.

Conversely, the results highlight the strong hyperparameter robustness of our approach. While smaller values ($\alpha \leq 0.6$) yield overly smooth fields that can marginally slow convergence due to softer attractors, any choice of $\alpha < 1$ successfully preserves the essential equilibrium characteristics. As a result, all $\alpha < 1$ configurations maintain highly competitive success rates, averaging above 0.90 on LIBERO. Setting $\alpha = 0.8$ strikes the best balance between rapid convergence and precise policy control under a limited step budget.

Table 7: Ablation on truncation parameter α. LIBERO success rates (%) under different truncation schedules.

α	LIBERO Success Rate (%)				Avg.
	Object	Goal	Spatial	Long	
0.0	**98.4**	94.6	93.4	77.0	90.9
0.2	96.8	**95.4**	95.8	80.2	92.1
0.4	98.2	93.4	95.6	78.2	91.4
0.6	98.2	94.2	95.6	77.8	91.5
0.8	97.8	94.0	**96.0**	**81.0**	**92.2**
1.0	93.2	84.0	89.8	55.8	80.7

H.2 Ablation on Truncation Parameter α for VLA Backbones

While Appendix H.1 validates the stationary-field formulation using the DiT-based policy, it is important to verify that the same design choice generalizes across different network backbones. Therefore, we further conduct an ablation study on the truncation parameter α using the VLA architecture. As discussed in Section 3.2, α controls the onset of the velocity rescaling decay and directly affects equilibrium convergence. Table 8 presents the success rates of the VLA-based policy 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 convergence and the optimizer often fails to reach the tight equilibrium manifold within a limited step budget. Conversely, an excessively large α, e.g., $\alpha = 0.9$, creates a sharper and stiffer optimization landscape, making it harder for the iterative solver to converge stably. Setting $\alpha = 0.8$ strikes the best balance, yielding attractors that are sharp enough for precise control while remaining smooth enough for rapid and stable inference.

H.3 Ablation Study: Effect of the Early-Stop Threshold τ

To investigate the trade-off between policy performance and inference speed, we ablate the early-stop threshold τ. Table 9 summarizes the success rates and the average number of inference steps

Table 8: Ablation on truncation parameter α for VLA backbones. We report LIBERO success rates (%) across different values of α.

α	LIBERO Success Rate				Avg.
	Goal	Spat.	Obj.	Long	
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

required. A smaller threshold (e.g., $\tau = 0.4$) enforces stricter convergence criteria, allowing the solver to approach the exact equilibrium. This yields the highest average success rate (93.5%) but requires more inference steps (11.6 steps on average). Conversely, relaxing the threshold to $\tau = 0.8$ significantly accelerates inference, reducing the average steps to 6.6, with only a marginal drop in overall performance (91.9% average success). This demonstrates that our early-stopping mechanism provides a flexible and robust knob to balance task accuracy and computational cost during real-world deployment.

Table 9: Ablation on the Early-Stop Threshold τ. We report the success rates (%) and the average number of inference steps across the LIBERO benchmark.

τ	LIBERO Object	LIBERO Goal	LIBERO Spatial	LIBERO Long	Avg	Avg. Steps
0.4	99.0	95.2	95.8	83.8	93.5	11.6
0.6	98.8	94.0	95.6	81.0	92.35	7.7
0.8	96.8	93.2	96.0	81.6	91.9	6.6

H.4 Ablation Study: Robustness to Fixed Step-Size Schedules

Finally, we ablate the step-size schedule to evaluate its impact on optimization stability. Table 10 compares our current schedule against various fixed step-size baselines (e.g., applying a constant step size for 20 steps). The empirical results show that even simple fixed step-size schedules remain highly competitive. For instance, the 0.02×20 schedule achieves an impressive 93.05% average success rate, which is only marginally lower than our current schedule (93.5%). The fact that our current schedule provides only a modest improvement over several reasonable alternatives is a crucial finding. It strongly suggests that GeCO's performance gains stem primarily from the fundamental stationary-field formulation itself, rather than relying on a highly specialized or fragile learning-rate schedule.

Table 10: Ablation on Fixed Step-Size Schedules. We compare our current adaptive schedule with several fixed step-size schedules (constant step size $\times$ total steps) on the LIBERO benchmark.

Step-size schedule	LIBERO Object	LIBERO Goal	LIBERO Spatial	LIBERO Long	Avg
0.015×20	97.0	92.0	94.8	77.2	90.3
0.02×20	98.0	94.8	95.4	84.0	93.1
0.025×20	98.4	95.0	94.2	80.4	92.0
0.035×20	98.0	95.0	95.0	77.4	91.4
Current schedule	99.0	95.2	95.8	83.8	93.5

H.5 Ablation Study: OOD Detection Hyperparameters

We further study the sensitivity of the proposed OOD detector to its decision-rule hyperparameters. In particular, we evaluate two lightweight temporal filtering strategies: Moving Average, parameterized by the window size w and threshold τ_{ma}, and Leaky Bucket, parameterized by the threshold τ, leakage rate λ, and bucket capacity B. Table 11 reports the true positive rate (TPR), true negative rate (TNR), and saved inference time under different configurations.

The results show that the OOD detector is robust under local perturbations of its hyperparameters. As expected, more aggressive settings, such as smaller thresholds or bucket capacities, increase TPR and save more inference time, but reduce TNR. Conversely, larger thresholds make the detector more conservative, improving TNR at the cost of detecting fewer OOD cases and saving less time. The

Table 11: Ablation on OOD Detection Hyperparameters. We compare different settings for Moving Average and Leaky Bucket detectors. The detector remains effective across a broad range of configurations, indicating that the OOD signal is primarily induced by the stationary-field geometry rather than fragile hyperparameter tuning.

Method	Hyperparameters	TPR	TNR	Saved Time
Moving Average	$w = 3,\ \tau_{ma} = 0.6$	89.4	78.6	50.7
Moving Average	$w = 5,\ \tau_{ma} = 0.6$	82.4	89.7	42.4
Moving Average	$w = 7,\ \tau_{ma} = 0.6$	70.6	95.6	33.1
Moving Average	$w = 5,\ \tau_{ma} = 0.5$	91.8	67.2	53.7
Moving Average	$w = 5,\ \tau_{ma} = 0.7$	63.4	96.2	29.2
Leaky Bucket	$\tau = 0.4,\ \lambda = 0.0,\ B = 0.7$	96.4	61.7	56.3
Leaky Bucket	$\tau = 0.5,\ \lambda = 0.0,\ B = 0.7$	90.4	85.2	48.9
Leaky Bucket	$\tau = 0.6,\ \lambda = 0.0,\ B = 0.7$	85.4	91.1	43.3
Leaky Bucket	$\tau = 0.5,\ \lambda = 0.1,\ B = 0.7$	85.8	89.2	45.3
Leaky Bucket	$\tau = 0.5,\ \lambda = 0.2,\ B = 0.7$	82.6	90.5	43.2
Leaky Bucket	$\tau = 0.5,\ \lambda = 0.0,\ B = 0.5$	93.4	73.9	53.4
Leaky Bucket	$\tau = 0.5,\ \lambda = 0.0,\ B = 0.9$	87.4	89.6	44.8

leakage parameter λ has a comparatively smaller effect and mainly serves as a secondary smoothing term. Overall, these trends suggest that the OOD behavior is driven by the residual-norm signal induced by the learned stationary field, while the detector hyperparameters primarily control the desired TPR–TNR–latency trade-off.

H.6 Comparison to Established OOD Detection Methods

We compare GeCO with Diff-DAgger [26] under different batch sizes to evaluate whether the stationary-field residual provides a competitive intrinsic OOD signal without relying on repeated forward passes. Table 14 reports the AUROC of Diff-DAgger with batch sizes ranging from 8 to 64, together with our method.

Table 12: Comparison with Diff-DAgger for OOD Detection. We report AUROC under different Diff-DAgger batch sizes. GeCO achieves the strongest OOD discrimination while avoiding the additional inference cost introduced by larger Diff-DAgger batches.

Method **Batch Size**	**Diff-DAgger** 8	**Diff-DAgger** 16	**Diff-DAgger** 32	**Diff-DAgger** 64	**GeCO** –
AUROC	0.753	0.762	0.782	0.812	**0.930**

The results show that increasing the Diff-DAgger batch size improves OOD detection, but the gain comes from additional computation due to multiple forward evaluations. In contrast, GeCO obtains a substantially higher AUROC of 0.930, outperforming the strongest Diff-DAgger variant by 0.118 absolute AUROC. This indicates that the residual norm induced by the learned stationary field is a more direct and efficient OOD signal, enabling lightweight safety monitoring without requiring expensive ensemble-style inference.

I Comparison with Adaptive ODE Solvers

A natural question is whether GeCO's efficiency advantage can be matched by replacing the fixed-step Euler integrator of standard rectified flow with an adaptive ODE solver such as Dormand–Prince (Dopri5) [12]. A structural difference makes this comparison unfavorable for adaptive ODE solvers on robotic action generation.

Multiple function evaluations per accepted step. Dopri5 is a multi-stage explicit Runge–Kutta method with an embedded error estimator. Each GeCO refinement step is a single function evaluation, so reported NFE equals the iteration count.

Empirical comparison. We evaluate three inference strategies on the same Continuous Rectified Flow backbone on LIBERO-Goal. Dopri5 uses default tolerances `rtol=atol=1e-3`.

Method	SR (%)	Avg. NFE
Dopri5 + FM	91.2	56.17
FM (Euler, fixed)	92.4	20.0
GeCO (ours)	**95.2**	**8.67**

Adaptive ODE integration is substantially *less* efficient than fixed-step Euler here, because the per-step RK overhead dominates any savings from coarser stepping. GeCO achieves both the highest success rate and the lowest NFE by removing both constraints: no integration horizon to traverse, and every function evaluation directly refines the action.

J Comparison with Image-Domain Equilibrium Matching

GeCO is inspired by time-unconditional generative dynamics in image generation, such as Equilibrium Matching [43], but the role of a stationary field is fundamentally different in closed-loop robotic control. Image generation is an offline sampling problem, where reducing refinement steps mainly improves rendering efficiency. Robot policy inference, however, is embedded inside a physical feedback loop: each planning call must produce temporally smooth, executable actions under real-time latency constraints, and the difficulty of this generation problem varies sharply across states, from free-space transit to precise grasping, contact-rich insertion, or language-grounded object selection. GeCO leverages the stationary field to address this control-specific challenge by turning fixed-schedule generation into convergence-driven action optimization, allowing computation to be allocated according to the physical and perceptual difficulty of the current state.

Moreover, anomaly detection is a first-order safety requirement in robotic deployment rather than a post-hoc sample-quality metric. A low-quality image sample can simply be discarded offline, whereas an erroneous action chunk may immediately affect the physical scene, causing unsafe contacts, task failure, or irreversible state changes before the next replan. This makes it important for a robot policy to detect abnormal action-generation behavior *before* execution. GeCO provides such a signal directly from its optimization dynamics: persistent high residual norms indicate that the policy has not settled to a low-norm action solution under the current observation, which may arise from distribution shift, visual ambiguity, language-grounding errors, or unseen contact configurations. Thus, compared with image-domain stationary generative modeling, GeCO's contribution is to make stationary geometry serve not only as an adaptive inference mechanism, but also as a training-free safety monitor for embodied control.

It is also important to distinguish our anomaly detector from the OOD mechanism in image-domain Equilibrium Matching. EqM frames its stationary dynamics through an energy-based perspective and performs OOD detection by evaluating an energy value, where ID samples are expected to have lower energy than OOD samples [43]. GeCO does not introduce or query an explicit scalar energy function for detection, as it can harm the action generation quality. Instead, we use the optimization process itself: the detector operates directly on the residual norm of the learned velocity field, which is already computed during action refinement for convergence checking. This distinction is critical for robotic deployment. Rather than treating OOD detection as a static density- or energy-estimation problem over completed samples, GeCO detects whether the current action-generation process is failing to converge before the action chunk is executed. In this sense, the anomaly score is not an additional energy-modeling head or post-hoc likelihood proxy, but an online safety signal induced by the velocity-field optimization dynamics.

K Additional LIBERO Results for GeCO + VLA

LIBERO [30] comprises four suites that isolate different distribution shifts in language-conditioned manipulation: goal changes (Goal), spatial relation shifts (Spat.), object category shifts (Obj.), and long-horizon compositional execution (Long). Table 13 reports additional LIBERO results for integrating GeCO with VLA backbones. GeCO remains comparable overall relative to standard fine-tuning: for π_0, it achieves a comparable average success rate, and for $\pi_{0.5}$, it remains close to the corresponding baseline. Beyond maintaining competitive task performance, GeCO enables more efficient inference through its adaptive optimization process, which allocates fewer refinement steps to easier states and more computation to difficult ones. These results suggest that GeCO preserves the

strong task performance of flow-matching VLA policies while providing adaptive inference efficiency studied in the main paper.

Table 13: Additional LIBERO results for GeCO + VLA. We report VLA success rates (%) on the LIBERO benchmark. Higher is better.

Method	Goal	Spat.	Obj.	Long	Avg.
π_0 + FAST [36]	88.6	96.4	96.8	60.2	85.5
π_0 [4]	95.8	96.8	98.8	85.2	94.2
$\pi_{0.5}$ [20]	**98.0**	**98.8**	98.2	**92.4**	**96.9**
GeCO (π_0)	96.4	96.8	97.0	85.4	93.9
GeCO ($\pi_{0.5}$)	96.4	97.4	**98.6**	**92.4**	96.2

L Compute Resources

For the DiT-based policy experiments, each training run used a single NVIDIA RTX5090 GPU with 32GB GPU memory and required approximately 4 hours. For the VLA policy experiments, each training run used 8 NVIDIA A100 GPUs with 80GB GPU memory per GPU and required approximately 18 hours, corresponding to about 144 H100 GPU-hours per run. The storage requirement was dominated by benchmark datasets, checkpoints, and rollout logs; in our experiments, a standard shared filesystem with sufficient capacity for dataset storage and model checkpoints was used. The full research project required additional compute beyond the final reported runs for preliminary experiments, debugging, and hyperparameter validation, but the compute numbers above summarize the resources needed to reproduce the main reported DiT-policy and VLA-policy training runs.

M Comparison to Established OOD Detection Methods

We compare GeCO with Diff-DAgger [26] under different batch sizes to evaluate whether the stationary-field residual provides a competitive intrinsic OOD signal without relying on repeated forward passes. Diff-DAgger estimates uncertainty by averaging the flow-matching loss over B sampled timesteps per planning call, so its inference cost scales linearly with B. In contrast, GeCO reuses the residual norm that is already computed during action refinement and therefore introduces *no additional forward passes* beyond the optimization itself. Table 14 reports both AUROC and the average per-call wall-time of the OOD signal under the same hardware (single NVIDIA RTX 5090).

Table 14: Comparison with Diff-DAgger for OOD Detection. We report AUROC and the average wall-time (ms) per planning call for computing the OOD signal, measured over 1,900 planning calls. Diff-DAgger's cost scales linearly with batch size B (empirically $\approx 2.34B + 12.1$ ms), while GeCO's signal is amortized into the optimization step that produces the action chunk and adds no extra forward passes.

Method	**Diff-DAgger**	**Diff-DAgger**	**Diff-DAgger**	**Diff-DAgger**	**GeCO**
Batch size B	8	16	32	64	–
AUROC ↑	0.753	0.762	0.782	0.812	**0.930**
Wall-time per call (ms) ↓	29.5	50.3	88.1	161.2	**0.0**[†]

[†]GeCO's OOD signal is the residual norm $\|f_\theta(a_K, s)\|_2$ already computed by the optimization step that produces the action chunk; no additional forward pass is required.

The results expose a fundamental trade-off in Diff-DAgger: improving AUROC requires increasing the batch size, which directly inflates inference latency. Even at $B = 64$, the strongest Diff-DAgger setting in our comparison still trails GeCO by 0.118 absolute AUROC while incurring an additional $\sim$161 ms per planning call. In our implementation, a single DiT function call takes approximately 9.5 ms, so the Diff-DAgger OOD probe at $B = 64$ is equivalent to roughly 17 additional function calls before every action chunk is executed. This overhead would substantially increase closed-loop latency on the real robot, especially in contact-rich manipulation settings where delayed corrective actions can directly degrade execution robustness. GeCO sidesteps this trade-off entirely: because the residual norm is a by-product of the action-generation optimization, the OOD signal is obtained *for free* during deployment.

Table 15: Licenses and terms of use for existing assets used in this work.

Asset	How it was used	License	Relevant terms of use
CleanDiffuser	Diffusion-policy implementation.	Apache License 2.0	Permits commercial use, modification, and distribution. Requires preserving copyright and license notices.
LIBERO	Manipulation benchmark, task suites, and demonstration data.	Code: MIT License Data: CC BY 4.0	**MIT:** Requires copyright and permission notices. **CC BY 4.0:** Requires appropriate attribution and indication of changes.
RoboTwin 2.0	Bimanual benchmark, simulator, and released datasets.	MIT License	Permits commercial use, modification, and distribution, provided copyright and permission notices are retained.
VLABench	Language-conditioned benchmark and fine-tuning data.	MIT License	Permits commercial use, modification, and distribution, provided copyright and permission notices are retained.
DINOv2	Pretrained visual backbone.	Apache License 2.0	Permits commercial use, modification, and distribution. Requires preserving copyright and license notices.
T5	Pretrained language model.	Apache License 2.0	Permits commercial use, modification, and distribution. Requires preserving copyright and license notices.

Practical implication. For real-time robot deployment with planning-call budgets in the tens of milliseconds, the linear scaling of Diff-DAgger makes large-B settings impractical: at $B = 64$, the OOD probe alone consumes more than 160 ms per call, exceeding typical real-time control envelopes for high-frequency manipulation. GeCO's residual-norm signal therefore offers a strictly dominant operating point—higher AUROC at zero marginal cost—and serves as a lightweight, deployment-ready safety monitor for flow-matching robot policies.

N Licenses and Terms of Use for Existing Assets

We used several existing open-source software packages, pretrained models, and benchmark assets in this work. We reviewed the official repositories or model/dataset cards of these assets and complied with their corresponding license terms. Table 15 summarizes the license names and the relevant terms of use.

For all assets, we used the public releases according to their documented licenses and terms of use. We did not remove copyright notices, license files, or attribution notices from the original assets. When adapting or integrating the assets into our experimental pipeline, we retained the corresponding license information and cited the original papers, repositories, or model/dataset cards in the main paper. We do not claim ownership of these third-party assets.

O Additional Experimental Validation

O.1 Variation across Training Seeds

We evaluate five independently trained GeCO checkpoints and three independently trained 20-step fixed-RF checkpoints. Each checkpoint is evaluated on all 40 LIBERO tasks over 2,000 episodes, totaling 16,000 evaluation episodes. Table 16 reports aggregate success for each training run and the mean and sample standard deviation across runs.

Table 16: LIBERO success across independent training runs (%). Dispersion denotes sample standard deviation across training runs.

Method	Success by training run	Mean $\pm$ s.d.
Fixed RF, 20 steps	90.00 / 88.90 / 88.45	89.12 ± 0.80
GeCO	93.50 / 91.85 / 93.05 / 92.20 / 90.90	92.30 ± 1.02

O.2 Success–Compute Comparison with Fixed RF and AdaFlow

We compare fixed-step RF, AdaFlow, and GeCO using the same DiT architecture, data, conditioning, normalization, and evaluation protocol. Each operating point covers all 40 LIBERO tasks and 2,000 episodes. NFE counts sequential backbone evaluations. For AdaFlow, we follow its released two-stage training design: the shared 258.3M-parameter RF backbone is frozen while a 722,945-parameter variance head (0.28% of the backbone size) is trained for an additional 30k steps. The parameter η_{AF} controls AdaFlow's adaptivity. At exactly 5 NFE, GeCO exceeds fixed RF by 1.65 percentage points.

Table 17: Success–compute comparison on LIBERO. Cap is the maximum inference budget; average NFE is the measured backbone evaluation count.

Cap	Method	Setting	Success (%)	Avg. NFE
5	Fixed RF	Exactly 5 steps	90.25	5.00
5	AdaFlow	$\eta_{\mathrm{AF}} = 0.2$	89.60	4.46
5	GeCO	Adaptive	91.90	5.00
10	Fixed RF	Exactly 10 steps	90.15	10.00
10	AdaFlow	$\eta_{\mathrm{AF}} = 0.2$	89.60	6.23
10	GeCO	Adaptive	92.40	8.70
20	Fixed RF	Exactly 20 steps	90.00	20.00
20	AdaFlow	$\eta_{\mathrm{AF}} = 0.2$	89.55	7.03
20	AdaFlow	$\eta_{\mathrm{AF}} = 0.1$	88.70	12.60
20	GeCO	Adaptive	93.50	11.60

Compared with AdaFlow at 4.46 NFE, GeCO at 5.00 NFE achieves 2.30 percentage points higher success. At the larger operating point, GeCO achieves 93.50% success with 11.60 average NFE, compared with 88.70% at 12.60 NFE for AdaFlow and 90.00% at 20.00 NFE for fixed RF. Relative to 20-step fixed RF, this corresponds to 3.50 percentage points higher success and 42% fewer backbone evaluations.

O.3 Paired Ablation of Adaptive Stopping

To isolate adaptive stopping, we run a paired 2,000-episode evaluation holding the checkpoint, update schedule, $K_{\max} = 20$, task and initial-state IDs, and random seeds fixed. Only the residual-based stopper is switched off or on. As shown in Table 18, enabling the stopper changes aggregate success from 91.8% to 92.5% and reduces average NFE from 20.00 to 10.85, a reduction of 45.8%. This paired experiment is separate from the fixed step-size schedule comparison in Table 10.

O.4 Additional Visual Distribution Shifts

We extend the OOD evaluation to two visual shifts: replacement of 50% of image pixels, and a wrist-camera extrinsic perturbation. For the latter, translation is sampled independently within ± 2 cm

Table 18: Paired evaluation of the residual-based stopper with the remaining inference and evaluation settings fixed.

Stopper	Success (%)	Avg. NFE
OFF	91.8	20.00
ON	92.5	10.85

along each axis and rotation is sampled about a random axis by up to 12°; the camera offset remains fixed within each episode. We calibrate the operating point to approximately 10% FPR using 250 held-out ID episodes, and evaluate on paired sets of 250 ID and 250 OOD episodes. Table 19 reports the results for GeCO and the low- and high-probe-budget Diff-DAgger settings. Across both shifts,

Table 19: Additional OOD evaluations. Precision, TPR, TNR, FPR, and Time Saved are percentages. B is the Diff-DAgger probe batch size.

Shift	Method	AUPRC	Prec.	TPR	TNR	FPR	Saved
Pixel replacement	GeCO	0.755	82.9	38.8	92.0	8.0	32.3
	Diff-DAgger, $B = 1$	0.749	79.1	28.8	92.4	7.6	24.0
	Diff-DAgger, $B = 64$	0.944	86.7	83.6	87.2	12.8	70.1
Wrist-camera shift	GeCO	0.648	74.4	23.2	92.0	8.0	19.4
	Diff-DAgger, $B = 1$	0.607	65.5	14.4	92.4	7.6	12.0
	Diff-DAgger, $B = 64$	0.757	78.4	46.4	87.2	12.8	38.7

GeCO obtains higher AUPRC, precision, TPR, and Time Saved than the $B = 1$ baseline at similar measured FPR. Increasing the baseline probe budget to $B = 64$ improves detection performance while requiring additional detector computation. These results complement the suite-shift evaluation with visual corruption and camera-pose changes.

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 state the main contributions of GeCO as time-unconditional action optimization, adaptive inference, VLA compatibility, and intrinsic OOD detection. These claims are supported by the simulation, VLA, OOD, and real-world experiments reported in the main paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a Discussion and Limitations section that acknowledges the need for stronger theoretical foundations, including convergence guarantees, performance bounds, and a deeper understanding of the OOD detection mechanism. The empirical scope and practical deployment setting are also reflected through benchmark and real-world evaluations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: [N/A]

   Justification: The paper does not present formal theorems or proof-based theoretical claims. The mathematical content is used to define the GeCO objective, stationary field construction, and inference procedure rather than to establish formal theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

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: The paper and appendix provide the model architecture, training recipe, inference procedure, benchmark protocols, evaluation rollouts, and hyperparameters needed to reproduce the main experimental results. Implementation details are provided for adaptive inference, VLA fine-tuning, OOD detection, and real-world deployment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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: We provide anonymized code and reproduction instructions in the supplementary material, together with detailed benchmark, training, and evaluation settings in the appendix.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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 paper specifies the benchmark protocols, rollout counts, model architecture, training schedules, inference step budgets, stopping thresholds, optimizer choices, and real-world evaluation details. Additional implementation details are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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: [No]

Justification: We provide statistical validation through standardized benchmark evaluation protocols and large-scale repeated rollouts, rather than formal hypothesis tests. Specifically, we report average success over 50 rollouts per task for LIBERO and VLABench and 100 rollouts per setting for RoboTwin 2.0, covering multiple task suites, tracks, difficulty settings, and environment randomizations; this follows common practice in recent VLA and robotic manipulation benchmarks and provides appropriate evidence for the reliability of the reported trends.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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 have indicated the details regarding to the computing resources in our appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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: The research uses standard robotics benchmarks, controlled robot experiments, and cited pretrained components, and does not involve private personal data, human-subject studies, or deceptive user-facing deployment. We have reviewed the NeurIPS Code of Ethics and are not aware of any deviations.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: No societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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: The paper does not release high-risk assets such as scraped datasets, image generators, large language models, or broadly deployable autonomous systems.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: The paper cites the existing datasets, benchmarks, pretrained encoders, VLA models, and codebases used in the experiments, including CleanDiffuser, LIBERO, RoboTwin, VLABench, DINOv2, T5, and OpenPI-related models.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

Answer: [N/A]

Justification: The paper does not introduce a new dataset, benchmark, or released pretrained model as a primary contribution. The contribution is a method and inference framework evaluated on existing benchmarks and controlled real-world tasks.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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: The paper does not involve crowdsourcing experiments or human-subject studies.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: The work does not involve crowdsourcing or human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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 does not use LLMs as an original or non-standard component of the research. Pretrained text encoders and VLA backbones used for language-conditioned robotic control are cited as existing model components rather than introduced as new LLM-based methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.