ADAPTIVE REPLANNING FOR VISION-LANGUAGE-ACTION POLICIES VIA OVERLAP DISAGREEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern vision-language-action (VLA) robot policies commonly use action chunking, where the VLA predicts and executes a sequence of future actions before performing inference again. While action chunking improves computational efficiency and temporal coherence, it introduces a fundamental tradeoff: frequent replanning reduces open-loop execution drift but increases inconsistency between consecutive plans, whereas infrequent replanning improves temporal consistency at the cost of accumulating stale execution error. In this work, we model this tradeoff using a nonparametric class of surrogate costs and show that it induces an optimal replanning horizon. We then show empirically that *weighted overlap disagreement*, an observable quantity that measures the disagreement between overlapping segments of consecutive action chunks, has a minimizer that serves as a reliable predictor of this optimal replanning horizon. Building on this insight, we propose disagreement-guided replanning (DGR), an online adaptive replanning strategy with negligible computational cost that dynamically adjusts the replanning horizon during inference without additional training. DGR is model-agnostic, easy to deploy, requires minimal tuning, and uses a single set of hyperparameters across all models and benchmarks. Across multiple VLA architectures and benchmark suites, DGR improves robustness and long-horizon task performance over baselines. Code is available here.

1 INTRODUCTION

Vision-language-action (VLA) policies have recently emerged as a promising paradigm for general-purpose robot control, enabling a single model to solve diverse manipulation tasks directly from multimodal observations and language instructions Brohan et al. (2022); Zitkovich et al. (2023); Kim et al. (2024; 2025); Black et al. (2024; 2025a); Bjorck et al. (2025). A common design choice in modern VLA systems is *action chunking*: instead of predicting a single action at every timestep, the policy generates a sequence of future actions which are then executed before replanning again Zhao et al. (2023); Chi et al. (2025). This strategy improves computational efficiency, reduces latency, and promotes temporally coherent behavior. However, chunked action generation introduces an important question: how frequently should the policy replan? Replanning too frequently can lead to unstable behavior, since consecutive policy evaluations may produce inconsistent future predictions due to stochastic inference, observation noise, or partial observability. In contrast, replanning too infrequently forces the policy to execute increasingly stale predictions open-loop, causing execution errors to accumulate over time. Figure 1 illustrates this phenomenon empirically: task success rate exhibits a non-monotonic dependence on the replanning horizon, revealing the existence of a nontrivial optimal replanning frequency.

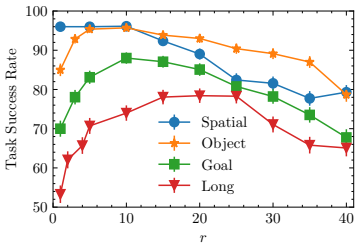

Figure 1: Task success rate of π_0 as a function of the replanning horizon (r) on the LIBERO benchmark.

This tradeoff is studied in Liu et al. (2025), but despite its importance, the optimal replanning horizon due to this tradeoff remains poorly understood. Existing approaches primarily rely on fixed replanning schedules, extremely expensive to tune for each task, or heuristic uncertainty estimates Liang

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

et al. (2026); Wang et al. (2026), without an understanding of why an optimal replanning horizon should emerge or how it can be identified online. Moreover, current adaptive replanning methods often require additional policy evaluations during inference, substantially increasing computational cost for large VLA models.

In this paper, we present an analytical framework to show the existence of an optimal replanning horizon in chunked VLA policies. We model the fundamental tradeoff inherent to replanning between two competing effects: *replanning inconsistency* and *open-loop execution drift*. We show that a measurable metric, called *overlap disagreement*, predicts the optimal replanning horizon and strongly correlates with task failure across multiple VLA architectures and benchmark suites. Noting that the replanning tradeoff, and the related optimal replanning horizon may change during inference, we build on the overlap disagreement metric to develop a lightweight online adaptive replanning strategy that dynamically adjusts the replanning horizon during inference without requiring additional training. Our proposed disagreement-guided replanning (DGR) method is model-agnostic, introduces negligible computational overhead, and requires very little hyperparameter tuning.

The main contributions of this paper are:

- Using a surrogate, we create a simple framework to understand the tradeoff between replanning inconsistency and open-loop drift in chunked VLA policies (Section 3).
- We show empirically that the overlap disagreement metric (Subsection 3.3) predicts the optimal replanning horizon and strongly correlates with task failure rate.
- We propose an online adaptive replanning algorithm, called DGR, based on this metric (Section 4). Our evaluation on multiple VLA architectures and benchmark suites shows that DGR, using one set of hyperparameters, is competitive across architectures, matches or exceeds the adaptive baselines in most settings, and provides its largest improvements on long-horizon tasks while being computationally cheaper and model-agnostic.

2 Background and Related Work

Robot manipulation as a POMDP. We model robot manipulation as a partially observed Markov decision process (POMDP) (Kaelbling et al., 1998), where the agent maintains a state $X_t = (x_t, x_t^{\text{env}}) \in \mathcal{X}$ comprising the robot state x_t and object state x_t^{env}, receives observations $o_t \in \mathcal{O}$ from cameras, and executes actions $u_t \in \mathcal{U}$. The environment evolves stochastically according to unknown transition dynamics $X_{t+1} \sim p(\cdot \mid X_t, u_t)$. Since the full state X_t is not directly observable, the agent must reason and act based on partial observations.

Vision-language-action models. VLA models allow tasks to be specified and solved through natural language instructions, by parameterizing a policy $\pi(u_t \mid o_t)$ on top of a pre-trained vision-language backbone, which provides rich semantic representations of observations and task instructions. Early VLAs, such as OpenVLA (Kim et al. (2024)), tokenize actions and treat control as sequence prediction. More recent systems adopt flow-matching and diffusion-based action heads (Chi et al., 2025) that generate entire action trajectories jointly, enabling smoother, higher-frequency control: π_0 (Black et al., 2024) and $\pi_{0.5}$ (Black et al. (2025a)) combine a pre-trained VLM with a flow-matching action expert, while GR00T N1.5 (Bjorck et al., 2025) uses a dual-system architecture coupling a VLM with a diffusion transformer.

Action chunking. Rather than predicting a single action for a task-observation tuple, recent VLAs generate a chunk of H future actions at once, and the first $r \leq H$ of the actions in a chunk are executed before replanning (Zhao et al. (2023); Chi et al. (2025)). Action chunking improves temporal coherence, reduces latency, and prevents the jerky motion associated with per-step replanning (Chi et al., 2025). It has become standard in flow-matching VLAs, where generating full trajectories in one forward pass is natural. Despite the prevalence of action chunking in VLAs, the replanning horizon r is usually chosen heuristically via grid search, with little theoretical understanding of what determines its optimal value (Black et al. (2024); Kim et al. (2025)). We address this gap through a modeling framework that characterizes the optimal replanning horizon through a class of functions meant to capture the dependence of the failure rate on the replanning horizon, and we instantiate an observable that predicts the minimizer of the failure rate. Prior work has also compared overlapping action predictions. BID (Liu et al. (2025)) uses backward coherence between overlapping

portions of N candidate chunks to select the best one at a fixed execution horizon. DGR instead uses disagreement between consecutive chunks to adapt the execution horizon online, without sampling additional candidates. Related theoretical work provides sufficient conditions on executed chunk length for stability (Zhang et al. (2025)) or models this tradeoff (Liu et al., 2025), but does not show the existence of an optimal horizon. Our analysis should be understood as a simplified surrogate model tailored to motivate horizon adaptation.

Adaptive replanning. A rapidly growing line of recent and concurrent work studies adaptive execution horizons. AAC (Liang et al., 2026) estimates entropy from N sampled trajectories, AutoHorizon (Wang et al., 2026) uses action self-attention, and similarly DVAC (Feng et al., 2026) measures variance across intermediate flow-matching estimates. AdaHorizon (Chopra et al., 2025) and PI-VLA (Jian et al., 2026) instead measure disagreement between two distinct, trained action heads: one predicting discrete actions and the other continuous actions, so the disagreement is between the same actions; PI-VLA additionally uses state-prediction error. These methods require multiple samples, model-internal signals, or specialized heads and training. Our approach starts from the tradeoff between open-loop drift and replanning inconsistency. We first model this tradeoff, then use overlap disagreement to predict a good static replanning horizon offline, and finally use the same signal to adapt the horizon online. Since disagreement is computed directly from consecutive action chunks, DGR requires no additional action heads or calls, nor architectural assumptions.

3 Characterization of Drift-Inconsistency Tradeoff

Modern VLA policies commonly generate a sequence of H actions (planning horizon) upon inference, of which a chunk of $r \leq H$ of actions is executed (control horizon) before replanning. A small r involves high total inference cost and potential inconsistency between subsequent plans, while a large r is detrimental to reactivity. The chunk size r is fixed in most systems, although it may be adapted during execution depending on task complexity, uncertainty, or environment interaction.

3.1 Replanning Policy

It is convenient to think of chunked execution as a two-level decision process: a *VLA policy* generates an action plan of horizon H, and a *replanning policy* selects the number of actions r to be executed after which replanning should occur. At replanning step k, corresponding to time $t_k = \sum_{i=0}^{k-1} r_i$, the VLA policy receives o_{t_k} and generates action sequence $u^{(k)} = (u_0^{(k)}, u_1^{(k)}, \ldots, u_{H-1}^{(k)})$, where H is the planning horizon. The replanning policy $\rho : \mathcal{H}_k \to \{1, .., H\}$ maps the execution history $\mathcal{H}_k$ to a replanning horizon r_k. The first r_k actions in $u^{(k)}$ are executed, followed by replanning based on a new observation $o_{t_k+r_k}$. The replanning policy implicitly predicts a future trajectory: $\hat{x}_{t_k+j}^{(k)} = \tilde{x}_{t_k} + \sum_{i=0}^{j-1} u_i^{(k)}$, where $\tilde{x}_{t_k}$ is an estimate of the current state x_{t_k} and $\hat{x}_{t_k+j}^{(k)}$ is the predicted system state after executing $j \leq H$ actions from the action sequence $u^{(k)}$. By construction, this sequence forms a planned trajectory that reconstructs the VLA training target, rather than representing the actual system dynamics.

Let us denote the expected task success rate starting from state x obtained under a replanning policy ρ by $S^\rho(x)$. The adaptive replanning problem consists of finding $\rho^* \in \arg\max_\rho S^\rho(x)$, which we refer to as the *Dynamic Optimal* policy. Most existing VLA systems instead use a policy restricted to a fixed replanning horizon over the entire episode, i.e., a policy $\rho_s \in \mathcal{P}_s = \{\rho : \rho(\mathcal{H}) = \rho(\mathcal{H}')\}$. We refer to an optimal fixed policy $\rho_s^* \in \arg\max_{\rho \in \mathcal{P}_s} S^\rho(x)$ as *Static Optimal* policy.

We use three related quantities to study the problem. In Section 3.2, we define the surrogate cost $C(r)$ that captures the dependence of $1 - S^\rho$ on r for $\rho \in \mathcal{P}_s$. In Section 3.3, we construct an empirical estimate $\widehat{L}_o(r)$ of this surrogate and use it as a proxy for task failure. Finally, in Section 4, we use $\widehat{L}_o(r)$ to derive a greedy online controller.

3.2 Existence of Optimal Fixed Replanning Horizon

For ease of exposition, let us consider the set $\mathcal{P}_s$ of static replanning policies, i.e., replanning occurs at times $t_k = kr$ and an episode of length T consists of $K = T/r$ replanning steps. Our goal is

to characterize the tradeoff between replanning inconsistency and open-loop execution drift and to provide a simplified framework for the analysis of the effect of the replanning horizon r on execution performance that will motivate our adaptive controller.

Open-loop drift. During execution, the system evolves according to the actual environment dynamics. The deviation between the planned state and the true state accumulates due to process noise, modeling error, partial observability, and unmodeled interactions with the environment. As many dynamical systems exhibit compounding sensitivity to prediction errors, the difference between the predicted and the actual state grows with the number of open-loop steps, and so does the task-relevant execution error. Let us define the *instantaneous* expected drift cost $\phi(j) := \mathbb{E}\left[\ell_{\text{drift}}\left(x_{kr+j}, \hat{x}^{(k)}_{kr+j}\right)\right]$, where ℓ_{drift} is a task-relevant measure of deviation between the true state and the predicted state, for example the l_2 norm, and the expectation is taken over rollouts.

Assumption 1 (Monotonic and unbounded drift cost)**.** *The function*

$$\phi : \mathbb{N} \to \mathbb{R}_{\geq 0}$$

satisfies:

1. $\phi(0) = 0$,
2. $\phi(j+1) > \phi(j)$ *for all* $j \geq 0$,
3. $\phi(j) \to \infty$ *as* $j \to \infty$.

Assumption 1 expresses the intuition that predictions become less reliable as they are executed farther open-loop. Since the task success depends on the execution of all actions in a chunk, we define the cumulative expected drift cost over a chunk of length r: $\Phi(r) := \sum_{j=0}^{r-1} \phi(j)$. As we show in the Appendix, $\Phi(r)$ is strictly increasing and convex in r.

Replanning inconsistency. While frequent replanning reduces execution drift, it also introduces inconsistency between consecutive plans. Given a state x_k, observation noise and generative inference can cause the newly predicted future trajectory to differ substantially from the continuation of the previous plan even without drift. Consequently, two consecutive plans may disagree about actions intended for the same future physical time. This inconsistency is inherent to the VLA policy, and is independent of the replanning policy. Denoting the expected cost introduced by each replanning event by σ, we thus make the following assumption.

Assumption 2. *The replanning inconsistency* $\sigma > 0$ *is independent of* r.

We can combine the two terms to characterize the tradeoff between open-loop drift and replanning inconsistency through the expected total cost

$$C(r) = \frac{T}{r}\left(\sigma + \Phi(r)\right). \tag{1}$$

The expected total cost captures the opposing effect that r has on the open-loop drift and on the total replanning inconsistency. For ease of notation, let us define

$$R(r) := \sum_{j=0}^{r-2} \left(\phi(r-1) - \phi(j)\right) = (r-1)\left(\phi(r-1) - \frac{\Phi(r-1)}{r-1}\right), \qquad R(1) = 0. \tag{2}$$

Observe that if $\sigma = 0$ then $R(r) = \frac{r(r-1)}{T}\left[C(r) - C(r-1)\right]$, i.e., it is the change of the cost when incrementing the chunk size $r-1$. The optimal horizon r^* is reached precisely when this marginal cost exceeds the replanning inconsistency cost σ. In fact, under Assumptions 1 and 2 the following result holds.

Proposition 1 (Optimal replanning horizon)**.** *There exists a minimizer*

$$r^* = \min(\arg\min_r C(r)),$$

given by $r^* = \min\{r \geq 1 : R(r+1) \geq \sigma\}$. *Furthermore,* $C(r)$ *decreases for* $r < r^*$ *and increases for* $r \geq r^* + 1$. *If* $R(r^*+1) = \sigma$, *then* $r^* + 1$ *is also a minimizer, otherwise the minimizer is unique.*

Proof sketch. The forward difference $C(r+1) - C(r)$ has the same sign as $R(r+1) - \sigma$. Furthermore, $R(r) \geq 0$ and is strictly increasing in r. Their difference therefore changes sign exactly once. The complete proof is in the Appendix. □

On the practical finite domain $r \in \{1, \ldots, H\}$, existence of a minimizer is immediate. Proposition 1 establishes the stronger and operationally useful result: the total cost is unimodal.

3.3 Overlap Disagreement

The total cost is not observable, alas. We now define a quantity that is observable and can be computed offline based on consecutive action chunks. Observe that any two consecutive replanning steps result in predicted actions corresponding to the same future physical times. If planning was deterministic and consistent and execution was perfectly predictable then the two plans would agree on the actions corresponding to these shared future times; in practice this is not true. For replanning steps $k-1$ and k we can define the overlap disagreement $D_j^{(k)} := u_j^{(k)} - u_{r+j}^{(k-1)}$, where $0 \leq j \leq L-1$ and $0 < L \leq H - r$ is the overlap length. Intuitively, $D_j^{(k)}$ should be strongly correlated with $\ell_{\text{drift}}\left(\hat{x}_{kr+j}^{(k)}, \hat{x}_{kr+j}^{(k-1)}\right)$ due to the underlying POMDP structure and with σ due to the uncertainty of the estimate $\tilde{x}_{kr}$ and due to the inherent randomness of action sequences $u^{(k)}$. We define the weighted aggregate overlap disagreement

$$\mathcal{L}_{\text{overlap}}(r) = \mathbb{E}\left[r \sum_{j=1}^{L-1} \|D_j^{(k)}\|_2^2 \right]. \tag{3}$$

The expectation is taken over time, task distribution, environment stochasticity and policy stochasticity. The factor r accounts for the fact that a disagreement is kept for r steps. We exclude the term corresponding to $j = 0$ by design to prevent the metric from being disproportionately influenced by necessary immediate responses to the latest observation. In our experiments we observed that using only the first overlap term ($L = 2$) already provides a highly predictive signal; we call this the aggregate overlap disagreement, defined as

$$\hat{\mathcal{L}}_{\text{o}}(r) = \mathbb{E}\left[r \|u_1^{(k)} - u_{r+1}^{(k-1)}\|_2^2 \right]. \tag{4}$$

As our results show in Section 5.2, the aggregate overlap disagreement (4) is compatible with the model described in Section 3.2, and its minimizer is a good predictor of the static optimal replanning horizon. It can, however, not be computed online.

4 Adaptive Chunking via Overlap Disagreement

Although the aggregate overlap disagreement can only be computed offline, its predictive power suggests that it can be used as the objective to minimize for finding the optimal replanning horizon. This motivates us to define the instantaneous overlap disagreement at replanning step $k \geq 1$ as

$$\ell_k = \sum_{j=1}^{L-1} \|D_j^{(k)}\|_2^2, \tag{5}$$

and define our objective as

$$\min_{\rho} \; \mathcal{L}_{\text{on}}(\rho) := \mathbb{E}_\rho \left[\frac{1}{K_\rho - 1} \sum_{k=1}^{K_\rho - 1} r_k \ell_k \right] \quad \text{with} \quad \sum_{k=0}^{K_\rho - 1} r_k = T,\; 1 \leq r_k \leq H - L,\; r_k = \rho(\mathcal{H}_k). \tag{6}$$

Note that K_ρ is induced by the replanning policy. For the constant policy $r_k = r$, $K^\rho = \frac{T}{r}$ and the objective reduces to (3).

Exact optimization of ρ would require long-horizon reasoning, either through reinforcement learning or explicit prediction of future trajectories. Instead, we seek a greedy decision rule that uses only

the information available at the current replanning event. Rather than selecting $r_k \in \{1, \dots, H\}$, we restrict the controller to retaining, increasing, or decreasing r_k by a fixed step Δ. This restriction keeps the decision complexity low, prevents abrupt changes, but allows the controller to explore the feasible range through successive updates. We thus define the action set $\mathcal{A} = \{-1, 0, +1\}$, and thus $r_k(a) = r_{k-1} + a\Delta$. To define the greedy comparison, fix a continuation policy $\bar{\rho}$ and let $\mu_k^a = \mathbb{E}_{a,\bar{\rho}}\left[\left(\sum_{j=k+1}^{K_{\bar{\rho}}^a - 1} r_j\right)^{-1} \sum_{j=k+1}^{K_{\bar{\rho}}^a - 1} r_j \ell_j\right]$ denote the mean disagreement over the remainder of the episode conditioned by taking action a and then following $\bar{\rho}$, where $K_{\bar{\rho}}^a$ is the resulting number of replannings. Let $\widehat{\mathcal{L}}_k^a(\mu_k)$ be the greedy predicted objective by taking taking action a and then following $\bar{\rho}$. Define the difference between increasing and decreasing the horizon as $d_k := \widehat{\mathcal{L}}_k^{+1}(\mu_k) - \widehat{\mathcal{L}}_k^{-1}(\mu_k)$. As shown in Appendix A.3, under local approximations $\mu_k := \mu_k^{+1} \approx \mu_k^{-1}$ and $K_{\bar{\rho}} := k_{\bar{\rho}}^{+} \approx k_{\bar{\rho}}^{-}$, we can write $d_k = c_k\left(\ell_k - \mu_k\right)$, where $c_k > 0$. If $d_k > 0$ the optimal action is $a_k = -1$, on the other hand, if $d_k < 0$ the optimal one is $a_k = +1$. One could potentially train a model to predict μ_k to choose what action to take. Instead, we maintain an exponential moving average (EMA) of the disagreement observed up to replanning step k: $m_k = (1-\beta)m_{k-1} + \beta\ell_k$, with $m_1 = \ell_1$ and $0 < \beta < 1$, and make use of the following

Assumption 3 (Directional calibration). *For some* $0 < \alpha < \min\left(1, \frac{1-\beta}{\beta}\right)$,

$$\mathbb{E}\left[\operatorname{sign}(\ell_k - m_k) d_k \,|\, |\ell_k - m_k| > \alpha m_k\right] > 0. \tag{7}$$

We validate Assumption 3 in Section 5.3. Intuitively, Assumption 3 means that, in expectation, if the current disagreement ℓ_k is greater than the past mean m_k, then l_k should also be greater than the future mean μ_k, and vice versa. We provide a stronger and more interpretable sufficient condition in Appendix A.3. Importantly, the comparisons are between the current disagreement ℓ_k with the average m_k, both of which are available online, and so Assumption 3 leads to the control rule

$$a_k = \begin{cases} -1, & \ell_k > (1+\alpha)m_k, \\ 0, & |\ell_k - m_k| \le \alpha m_k, \\ +1, & \ell_k < (1-\alpha)m_k, \end{cases} \qquad r_k = \operatorname{clip}_{[1, H-L]}\left(r_{k-1} + a_k\Delta\right). \tag{8}$$

Observe that by construction whenever $a_k \neq 0$ chosen using 8, it holds that

$$\mathbb{E}\left[\widehat{\mathcal{L}}_k^{a_k}(\mu_k) - \widehat{\mathcal{L}}_k^{-a_k}(\mu_k) \,\middle|\, |\ell_k - m_k| > \alpha m_k\right] < 0. \tag{9}$$

Thus, the controller selects the better directional update in expectation. When the signal lies inside the dead zone, neither direction is supported by Assumption 3, so DGR maintains the current horizon to avoid oscillation. The complete derivation and pseudocode are provided in Appendix A.3.

The proposed policy attempts to approximate ρ^* without requiring task-specific knowledge or additional training, it introduces negligible computational overhead and is VLA architecture-agnostic. We refer to the proposed method as Disagreement-Guided Replanning (DGR) algorithm.

5 Numerical Results

5.1 Evaluation Methodology

Policies and tasks: We evaluated DGR on four VLA architectures: OpenVLA-OFT ($H = 8$) (Kim et al., 2025), π_0 ($H = 50$) (Black et al., 2024), $\pi_{0.5}$ ($H = 10$) (Black et al., 2025a), and GR00T N1.5 ($H = 16$) (Bjorck et al., 2025), using checkpoints from the official repositories. We used the LIBERO benchmark (Liu et al., 2023) for the evaluation, which consists of four task suites, each containing 10 similar manipulation tasks. Each task is evaluated 50 times, and we report mean success rates along with standard errors across runs. We also provide a failure mode on the RoboCasa benchmark (Nasiriany et al., 2024), and an extension to overcome it. Finally, we stress test the algorithm under an adversarial patch attack, and a real world push-T task with CloudGripper (Zahid & Pokorny, 2024). For simplicity, our algorithm uses unnormalized translational DoFs.
Baselines: We use three baselines: AAC (Liang et al., 2026), AutoHorizon (Wang et al., 2026) and the Static Optimal policy ρ_s^*. Note that ρ_s^* is not actually available at deployment time: it is found through a complete sweep per suite[1], and as such it is a high bar to beat; for example, for π_0 the

[1] to get it per-task with *sufficient statistics* would too expensive even for research purposes, in the Appendix we give evidence that per suite minimizer is near optimal for all the tasks in the suite

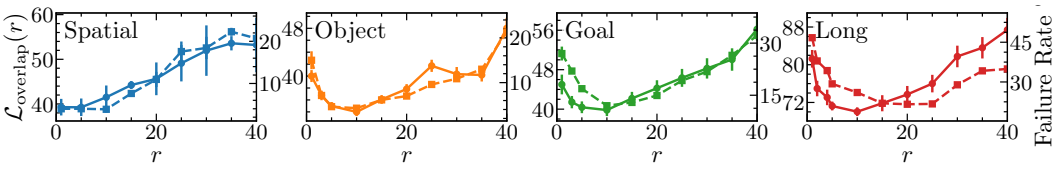

Figure 2: Task failure rate (right axis, dashed) and overlap disagreement (left axis, solid) vs. replanning horizon r for π_0 on LIBERO. The minimizer of overlap disagreement consistently predicts the static optimal r^* that minimizes the task failure rate.

chosen value is given by the minimizer of Figure 2. For DGR, replanning decisions are made online at inference time without task-specific tuning. In contrast, all baseline methods are tuned for their respective settings using the parameters provided by the authors, except AutoHorizon on π_0 that uses the same parameters as $\pi_{0.5}$. Unless otherwise specified, the hyperparameters for DGR are $(\beta, \alpha, \Delta, L) = (0.2, 0.25, 2, 2)$, found to be near-optimal on $\pi_{0.5}$-LIBERO-Long-1. All evaluations were performed on a Dell PowerEdge R7515 server with a 32 core AMD EPYC 7543P CPU and an NVidia A30 GPU.

5.2 Fixed Replanning Horizon

We validate two links between the proposed proxy and the theory: if its minimizer tracks the optimal replanning horizon, and if its inferred cost components satisfy the structural assumptions of the surrogate class.

We first evaluate the task performance for fixed replanning horizons. Figure 2 shows the overlap disagreement 4 and task failure rate as a function of the replanning horizon r. We observe a strong correlation between the two curves across all environments: the minimizer of the overlap disagreement objective consistently matches the replanning horizon that minimizes the task failure rate; when there is discrepancy, the failure rate is relatively flat near the minimum. Importantly, this correlation also holds at the individual task level rather than only after aggregation across several tasks. We provide further results in the Appendix, in particular per-task results.

The theoretical framework of Section 3.2 characterizes a family of cost functions decomposing into a constant replanning term σ and an increasing drift term $\Phi(r)$; any member of this family has a minimizer r^*. Since $\hat{\mathcal{L}}_{\text{o}}(r)$ empirically recovers r^*, we ask whether its U-shaped profile is compatible with the specific modeled tradeoff. To answer this question, we impose the structural decomposition and verify the consistency with the Assumption 2 and Assumption 1 on σ and $\phi(r)$, respectively. Since $\Phi(1) = \phi(0) = 0$ in our discrete-time model, the overlap disagreement at $r = 1$ carries very little drift, and thus we can use the approximation $\sigma \approx \frac{\hat{\mathcal{L}}_{\text{o}}(1)}{T}$. Across all evaluated task suites, which differ substantially in typical episode length T and value of $\hat{\mathcal{L}}(1)$, we measured values between 0.28 and 0.32 for π_0, *suggesting* σ is an intrinsic property of the policy, not of the replanning schedule, independent of both r and the task. Similarly, we obtain ~ 0.011 for GR00T and ~ 0.029 for $\pi_{0.5}$.

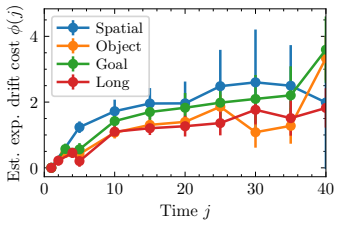

Figure 3: Estimated drift cost $\phi(j)$, derived from the overlap disagreement curves across four LIBERO task suites. The increasing profile is consistent with Assumption 1.

We can now express the estimated drift component as $\sum_{j=0}^{r-1} \phi(j) = \Phi(r) \approx \frac{r}{T}\hat{\mathcal{L}}_{\text{o}}(r) - \hat{\sigma}$ and use it to estimate $\phi(j)$. The results are shown in Figure 3 for four LIBERO tasks. Taking into account experimental uncertainties, the curves are increasing, consistent with Assumption 1.

Table 1: Adaptive replanning performance for LIBERO tasks.

Model	Method	Spatial	Object	Goal	Long
π_0	Our	$\underline{95.8} \pm 0.9$	$\mathbf{98.6} \pm 0.6$	$\mathbf{95} \pm 1$	$\mathbf{89} \pm 1$
	AAC	93 ± 1	96.4 ± 0.8	$\underline{92} \pm 1$	$\underline{82} \pm 2$
	AutoHorizon	85 ± 2	89 ± 1	86 ± 2	72 ± 2
	$r = 1$	95.6 ± 0.8	84 ± 2	73 ± 2	54 ± 2
	Static Optimal	$\mathbf{96.0} \pm 0.8$	$\underline{98.0} \pm 0.6$	$\mathbf{95} \pm 1$	79 ± 2
$\pi_{0.5}$	Our	$\underline{98.8} \pm 0.5$	$\mathbf{99.2} \pm 0.4$	$\mathbf{99.0} \pm 0.4$	$\mathbf{95.4} \pm 0.9$
	AAC	98.4 ± 0.5	$\mathbf{99.2} \pm 0.4$	$\underline{98.4} \pm 0.6$	94.2 ± 0.9
	AutoHorizon	98.6 ± 0.5	98.2 ± 0.6	98 ± 0.6	$\underline{94.4} \pm 0.9$
	$r = 1$	98.4 ± 0.5	94 ± 2	90 ± 2	76 ± 2
	Static Optimal	$\mathbf{99} \pm 0.5$	$\underline{98.4} \pm 0.6$	97.4 ± 0.8	92 ± 1
GR00T N1.5	Our	$\mathbf{91} \pm 1$	$\underline{96.6} \pm 0.8$	$\mathbf{95.8} \pm 0.8$	$\mathbf{91} \pm 1$
	AAC	$\underline{90} \pm 1$	$\mathbf{97} \pm 0.7$	$\mathbf{95.8} \pm 0.8$	$\mathbf{91} \pm 1$
	$r = 1$	91 ± 2	91 ± 2	90 ± 2	79 ± 2
	Static Optimal	$\underline{90} \pm 1$	96.0 ± 0.8	$\underline{95.4} \pm 0.7$	$\underline{90} \pm 1$
OpenVLA-OFT	Our	$\mathbf{93} \pm 1$	$\underline{97.8} \pm 0.5$	$\underline{94} \pm 1$	$\underline{94} \pm 1$
	$r = 1$	$\underline{87} \pm 2$	86 ± 2	82 ± 2	64 ± 2
	Static Optimal	$\mathbf{93} \pm 1$	$\mathbf{98.4} \pm 0.5$	$\mathbf{96} \pm 0.9$	$\mathbf{95} \pm 1$

Table 2: Hyperparameter sensitivity analysis on π_0-LIBERO-long.

Δ	Success (%)	β	Success (%)	α	Success (%)	L	Success (%)	r_0	Success (%)
1	84 ± 2	0.05	86 ± 2	0.05	87 ± 2	2	88 ± 1	1	89 ± 1
2	88 ± 1	0.10	85 ± 2	0.10	86 ± 2	3	87 ± 2	3	87 ± 2
5	88 ± 2	0.20	89 ± 1	0.25	89 ± 1	4	88 ± 1	5	86 ± 2
7	84 ± 2	0.50	84 ± 2	0.35	87 ± 2	5	86 ± 2	10	88 ± 2
10	82 ± 2	1.00	55 ± 2	0.45	85 ± 2	6	87 ± 2	40	80 ± 2

5.3 Adaptive Replanning

Table 1 shows the task success rate for various VLAs and task suites. We can observe that DGR matches or exceeds the baselines in most settings and gives its largest gains on LIBERO-Long. On the LIBERO-Long task suite DGR outperforms the Static Optimal on all diffusion-based VLA architectures, by up to 10 percentage points. Importantly, the performance is equal or better without hyperparameter tuning. Notably, DGR also performs well with OpenVLA-OFT, a non-diffusion architecture, demonstrating its model-agnostic nature. For the suite with the biggest improvement, the per-task breakdown is: compared to Static Optimal for π_0 ($\pi_{0.5}$), DGR improves on 7/10 (6/10), ties on 2/10 (1/10), and degrades on 1/10 (3/10). We provide additional results, such as per-task results and optimals, in the Appendix. RoboCasa presents an additional challenge: its embodiment comprises multiple independent kinematic groups (i.e., separate arms), which may each have a different optimal replanning horizon (see the Appendix). Since VLAs generate a single chunk covering all degrees of freedom simultaneously, we provide an extension of DGR to multiple kinematic groups based on inpainting Black et al. (2025b) in Appendix E.1. The mean success rate for [DGR/$r = 1$/Static Optimal/DGR Extended] is: 41 ± 2, 31 ± 2, 43 ± 1 and 48 ± 1.The signal is also stable under an adversarial patch attack: for DGR / clean-tuned r^* / attack-tuned r^*, success rates are 74/54/69 (Spatial), 53/53/55 (Object), 51/52/52 (Goal), and 33/33/33 (Long). Finally, on the real-world Push-T task with CloudGripper, $r = 1$, $r = 10$, $r = 40$, and DGR achieve 10%, 65%, 20%, and 70% success rate, respectively. This shows stability and within task adaptation. For details, see Appendix F and G.

Table 2 provides a sensitivity analysis of DGR to the hyperparameters Δ, α, β and L. The results show that the success rate is stable across a broad practical range of the hyperparameter values, and DGR consistently outperforms all baselines on π_0-long for a broad range of hyperparameter values, although we can see some degradation.

Table 3: Computational cost comparison on LIBERO-Long. The values are the ratio of the cost of AAC divided by that of DGR; higher ratio implies DGR is more efficient. "Calls" is the average number of policy evaluations per episode, it is also a ratio. Tested on NVIDIA A30.

Model	FLOPs/call	Calls	Total FLOPs
π_0 (cached)	2.65	0.8	2
$\pi_{0.5}$ (cached)	1.32	1.5	2
GR00T N1.5 (cached)	2.30	1.3	3
π_0 ($N = 20$)		0.8	15
$\pi_{0.5}$ ($N = 20$)	20	1.5	30
GR00T N1.5 ($N = 20$)		1.3	27

To test the validity of Assumption 3 we approximated μ_k by $\widehat{\mu}_k = \sum_{i>k} r_i \ell_i / \sum_{i>k} r_i$. Our evaluation on π_0-Libero-long showed that the inequality $\mathrm{sign}(\ell_k - m_k) d_k > 0$, is satisfied for 90% of the updates.

We finally compare the computational cost of DGR and the best adaptive baseline. AAC requires multiple policy evaluations during inference in order to estimate the entropy used for deciding the replanning horizon, its computational cost thus scales with the number N of sampled trajectories. In contrast, our method and AutoHorizon require no additional forward passes beyond standard VLA policy inference.Table 3 shows the relative computational cost between two versions of AAC and our approach. Without VLM caching, the computational cost of AAC scales approximately proportionally with the number of samples N. With VLM caching, the computational cost of AAC grows substantially slower because the VLM is run once, and only the action head is run N times. Yet, even compared to AAC with caching, DGR not only involves fewer FLOPs per call, but it also requires fewer replannings per episode for $\pi_{0.5}$ and GR00T, yielding 2–3$\times$ fewer total FLOPs than the cached AAC implementation. Thus, DGR achieves equal or better performance as the state of the art adaptive chunking methods at significantly lower computational cost. We provide a more detailed computational cost analysis in the Appendix.

6 Conclusion

We studied execution-horizon selection in action-chunked VLA policies. Under explicit assumptions, our surrogate model formalizes the tradeoff between replanning inconsistency and open-loop drift and yields a unimodal horizon-dependent cost. We used the weighted overlap disagreement as an output-only observable whose minimizer empirically predicts the best fixed horizon, and used it to develop DGR, a lightweight online controller requiring neither training, additional policy evaluations, nor model-internal signals. Across four VLA families, DGR is competitive with or improves upon tuned fixed and adaptive baselines in most settings, with its largest gains on LIBERO-long, while using one hyperparameter configuration. The results extend to visual perturbations and a real-world Push-T task.

Limitations

Our analysis models replanning through a surrogate execution cost designed to prove the tradeoff between open-loop drift and replanning inconsistency. While the overlap disagreement objective empirically tracks the optimal replanning horizon across architectures and tasks, and the correspondence with the two components of $C(r)$ is shown, the links are still empirical. In addition, the proposed adaptive controller inherits this limitation, as it relies on the overlap disagreement; DGR can fail when consecutive plans agree but are consistently incorrect. Visual occlusion may therefore worsen the perfomance, while multimodal yet valid action predictions may produce high disagreement. Performance may also depend on the relative scaling of the action dimensions used to compute disagreement. For multi-group replanning, although the current implementation uses standard inference-time guidance techniques, it remains computationally more expensive than standard DGR. Developing more computationally efficient algorithms for multi-group replanning is therefore an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for code generation, that was subsequently checked by a human. We have not used generative AI tools for ideation, mathematical proofs, or any sort of discovery. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All proofs are given, and the code is attached.

REFERENCES

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proc. of Conference on Robot Learning (CoRL)*, 2025a.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2025b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

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.

Samarth Chopra, Alex McMoil, Ben Carnovale, Evan Sokolson, Rajkumar Kubendran, and Samuel Dickerson. Everydayvla: A vision-language-action model for affordable robotic manipulation. *arXiv preprint arXiv:2511.05397*, 2025.

Xiangdong Feng, Yuxuan Cheng, Chen Shi, Boyao Han, Yuxuan Yan, Yitong Hong, Zhuotao Tian, and Li Jiang. Denoising tells when to replan: Denoising-variance adaptive chunking for flow-based robot policies. *arXiv preprint arXiv:2606.03847*, 2026.

Yina Jian, Di Tian, Xuan-Jing Chen, Zhen-Yuan Wei, Chen-Wei Liang, and Mu-Jiang-Shan Wang. Pi-vla: Adaptive symmetry-aware decision-making for long-horizon vision–language–action manipulation. *Symmetry*, 18(3):394, 2026.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proc. of Robotics: Science and Systems (RSS)*, 2025.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. In *Proc. of IEEE/CVF Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2026.

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.

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

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

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.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. VLA knows its limits. *arXiv preprint arXiv:2602.21445*, 2026.

Muhammad Zahid and Florian T Pokorny. Cloudgripper: An open source cloud robotics testbed for robotic manipulation research, benchmarking and data collection at scale. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12076–12082. IEEE, 2024.

Thomas T Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and exploratory data collection yield exponential improvements in behavior cloning for continuous control. *arXiv preprint arXiv:2507.09061*, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proc. of Robotics: Science and Systems (RSS)*, 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Proofs

A.1 Proof that $\Phi(r)$ is Strictly Increasing and Convex

Proof. Recall that $\Phi(r) = \sum_{j=0}^{r-1} \phi(j)$.

Strictly increasing. $\Phi(r+1) - \Phi(r) = \phi(r) > 0$ for all $r \geq 1$, since ϕ is strictly increasing with $\phi(0) = 0$, so $\phi(r) > \phi(0) = 0$.

Strictly convex. The second difference is $\Phi(r+1) - 2\Phi(r) + \Phi(r-1) = \phi(r) - \phi(r-1) > 0$ for all $r \geq 1$, since ϕ is strictly increasing. □

A.2 Proof of Proposition 1

Proof. We compute the forward difference of $C(r)$ as

$$\begin{aligned} C(r+1) - C(r) &= T\left(\frac{\sigma + \Phi(r+1)}{r+1} - \frac{\sigma + \Phi(r)}{r}\right) \\ &= \frac{T}{r(r+1)}\left(r\,\Phi(r+1) - (r+1)\Phi(r) - \sigma\right). \end{aligned} \tag{10}$$

We can simplify the terms in the parentheses and obtain

$$r\,\Phi(r+1) - (r+1)\Phi(r) = r\phi(r) - \Phi(r) \\ = \sum_{j=0}^{r-1}(\phi(r) - \phi(j)) = R(r+1). \tag{11}$$

Thus, we can write

$$C(r+1) - C(r) = \frac{T}{r(r+1)}\big(R(r+1) - \sigma\big), \tag{12}$$

which shows that the sign of $C(r+1) - C(r)$ equals the sign of $R(r+1) - \sigma$.

Next, we establish that $R(r)$ is strictly increasing by expressing its increment and observing that

$$R(r+1) - R(r) = \sum_{j=0}^{r-1}(\phi(r) - \phi(j)) - \sum_{j=0}^{r-2}(\phi(r-1) - \phi(j)) = r(\phi(r) - \phi(r-1)) > 0, \tag{13}$$

since ϕ is strictly increasing. Furthermore, observe that $R(1) = 0$, $R(r) \geq \phi(r-1)$, and $\phi(r) \to \infty$. Hence, $R(r) \to \infty$ as $r \to \infty$.

Since $R(1) = 0 < \sigma$ and $R(r) \to \infty$, the set $\{r \geq 1 : R(r+1) \geq \sigma\}$ is non-empty. Therefore,

$$r^* = \min\big\{r \geq 1 : R(r+1) \geq \sigma\big\} \tag{14}$$

is well defined and satisfies $r^* \geq 1$. By definition, $R(r+1) < \sigma$ for all $r < r^*$, and $R(r^*+1) \geq \sigma$.

Strict decrease for $r < r^*$. If $r < r^*$ then $r + 1 \leq r^*$ and thus $R(r+1) < \sigma$. Consequently, $C(r+1) - C(r) = \frac{T}{r(r+1)}\,(R(r+1) - \sigma) < 0$.

Behavior at $r = r^*$. Since $R(r^*+1) \geq \sigma$, by 12, we have $C(r^*+1) - C(r^*) = \frac{T}{r^*(r^*+1)}\,(R(r^*+1) - \sigma) \geq 0$.

Strict increase for $r > r^*$. Since R is strictly increasing and $R(r^*+1) \geq \sigma$, we have $R(r+1) > R(r^*+1) \geq \sigma$. Therefore, $C(r+1) > C(r)$.

Conclusion. C is strictly decreasing on $\{1, \ldots, r^*\}$ and strictly increasing on $\{r^*+1, r^*+2, \ldots\}$. Therefore r^* is a global minimizer. **Note on uniqueness.** If $R(r^*+1) > \sigma$, we have $C(r^*+1) > C(r^*)$ and thus r^* is the unique minimizer. If $R(r^*+1) = \sigma$, we have $C(r^*+1) = C(r^*)$ and therefore, r^* and r^*+1 are both minimizers. □

A.3 Derivation of the DGR Controller

Define the accumulated objective numerator and remaining execution budget as

$$A_k = \sum_{i=1}^{k-1} r_i \ell_i, \qquad B_k = T - \sum_{i=0}^{k-1} r_i. \tag{15}$$

Fix a continuation policy $\bar{\rho}$, followed after the current action under both candidates. Then the corresponding future mean disagreement is

$$\mu_k^a = \frac{1}{B_k - r_k(a)}\mathbb{E}_{a,\bar{\rho}}\left[\sum_{i=k+1}^{K_k^a - 1} r_i \ell_i\right]. \tag{16}$$

Thus, $(B_k - r_k(a))\mu_k^a$ is the expected future contribution to the objective after executing the current candidate horizon. Since K_k^a replanning events produce $K_k^a - 1$ overlap measurements, the greedy predicted objective is

$$\widehat{\mathcal{L}}_k^a(\mu_k^a) = \frac{A_k + r_k(a)\ell_k + \big(B_k - r_k(a)\big)\mu_k^a}{K_k^a - 1}. \tag{17}$$

The difference between increasing and decreasing the horizon is therefore

$$\begin{aligned} d_k &:= \widehat{\mathcal{L}}_k^{+1}(\mu_k^{+1}) - \widehat{\mathcal{L}}_k^{-1}(\mu_k^{-1}) \\ &= \frac{A_k + r_k(+1)\ell_k + \big(B_k - r_k(+1)\big)\mu_k^{+1}}{K_k^{+1} - 1} \\ &\quad - \frac{A_k + r_k(-1)\ell_k + \big(B_k - r_k(-1)\big)\mu_k^{-1}}{K_k^{-1} - 1}. \end{aligned} \tag{18}$$

We now apply the local approximations:

$$\mu_k^{+1} \approx \mu_k^{-1} \approx \mu_k, \qquad K_k^{+1} \approx K_k^{-1} \approx K_k. \tag{19}$$

The first assumes that a local change of Δ does not substantially alter the disagreement generated by the common continuation policy. The second assumes that it does not change the total number of replanning events. Since this number is integer-valued, the latter is exact whenever the two candidates lie on the same local plateau.

Under these approximations,

$$\widehat{\mathcal{L}}_k^{a}(\mu_k) \approx \frac{A_k + r_k(a)\ell_k + \big(B_k - r_k(a)\big)\mu_k}{K_k - 1}. \tag{20}$$

Substituting $r_k(a) = r_{k-1} + a\Delta$ yields

$$\begin{aligned} d_k &\approx \frac{\big[r_k(+1) - r_k(-1)\big](\ell_k - \mu_k)}{K_k - 1} \\ &= \frac{2\Delta}{K_k - 1}(\ell_k - \mu_k). \end{aligned} \tag{21}$$

Consequently,

$$d_k \approx c_k(\ell_k - \mu_k), \qquad c_k := \frac{2\Delta}{K_k - 1} > 0. \tag{22}$$

Thus, decreasing the horizon is locally preferable when $\ell_k > \mu_k$, whereas increasing it is preferable when $\ell_k < \mu_k$.

The future mean μ_k is unavailable online. DGR instead compares the current disagreement ℓ_k with the EMA m_k. Whenever

$$|\ell_k - m_k| > \alpha m_k, \tag{23}$$

the selected action satisfies

$$a_k = -\operatorname{sign}(\ell_k - m_k). \tag{24}$$

Therefore,

$$\widehat{\mathcal{L}}_k^{a_k} - \widehat{\mathcal{L}}_k^{-a_k} = -\operatorname{sign}(\ell_k - m_k) d_k. \tag{25}$$

Taking the conditional expectation and applying Assumption 3 gives

$$\mathbb{E}\Big[\widehat{\mathcal{L}}_k^{a_k} - \widehat{\mathcal{L}}_k^{-a_k} \,\Big|\, |\ell_k - m_k| > \alpha m_k\Big] < 0. \tag{26}$$

Stronger sufficient condition. A stronger pointwise sufficient condition is

$$|m_k - \mu_k| \le \varepsilon_\mu \alpha m_k \qquad 0 \le \varepsilon_\mu < 1, \tag{27}$$

on every update event. This condition interprets the past mean disagreement m_k as an estimator of the future mean μ_k, with an error smaller than the dead-zone margin. This is an especially attractive interpretation since m_k is an EMA, so it's biased towards the recent values.

Indeed, using Eq. 22,

$$\begin{aligned} \operatorname{sign}(\ell_k - m_k) d_k &\approx c_k \operatorname{sign}(\ell_k - m_k)\big[(\ell_k - m_k) + (m_k - \mu_k)\big] \\ &\ge c_k\big(|\ell_k - m_k| - |m_k - \mu_k|\big) \\ &> c_k(1 - \varepsilon_\mu)\alpha m_k > 0. \end{aligned} \tag{28}$$

Therefore, Eq. 27 implies directional calibration, although it is not required by DGR.

The derivation assumes that both directional candidates are feasible. At the boundary of $[1, H - L]$, the proposed horizon is clipped as in Eq. 8.

Algorithm 1 DGR

Require: VLA policy π, planning horizon H, overlap length L, threshold α, EMA weight β, step size Δ

$r_0 \leftarrow 1$, $m \leftarrow$ undefined, $k \leftarrow 0$, $t_0 \leftarrow 0$
Generate $U^{(0)} = (u_0^{(0)}, \ldots, u_{H-1}^{(0)}) \sim \pi(\cdot \mid o_{t_0})$
while episode not terminated **do**
    Execute $u_0^{(k)}, \ldots, u_{r_k-1}^{(k)}$
    $t_{k+1} \leftarrow t_k + r_k$
    if episode terminated **then**
        break
    end if
    Generate $U^{(k+1)} \sim \pi(\cdot \mid o_{t_{k+1}})$
    Compute

$$\ell_{k+1} \leftarrow \sum_{j=1}^{L-1} \left\| u_j^{(k+1)} - u_{r_k+j}^{(k)} \right\|_2^2$$

    if m is undefined **then**
        $m \leftarrow \ell_{k+1}$
    else
        $m \leftarrow (1-\beta)m + \beta\ell_{k+1}$
    end if
    if $\ell_{k+1} > (1+\alpha)m$ **then**
        $r_{k+1} \leftarrow r_k - \Delta$
    else if $\ell_{k+1} < (1-\alpha)m$ **then**
        $r_{k+1} \leftarrow r_k + \Delta$
    else
        $r_{k+1} \leftarrow r_k$
    end if
    $r_{k+1} \leftarrow \mathrm{clip}(r_{k+1}, 1, H-L)$
    $k \leftarrow k+1$
end while

Remark on the EMA parameters. The controller compares ℓ_k with the updated EMA $m_k = (1-\beta)m_{k-1} + \beta\ell_k$. The condition for decreasing the horizon satisfies

$$\ell_k > (1+\alpha)m_k \quad \Longleftrightarrow \quad \big[1-\beta(1+\alpha)\big]\ell_k > (1+\alpha)(1-\beta)m_{k-1}. \tag{29}$$

Since the right-hand side is nonnegative, this update is feasible only if

$$\beta < \frac{1}{1+\alpha}, \qquad \text{equivalently} \qquad \alpha < \frac{1-\beta}{\beta}. \tag{30}$$

The increasing update remains feasible for every $0 < \alpha < 1$. We therefore require

$$0 < \alpha < 1, \qquad 0 < \beta < \frac{1}{1+\alpha}. \tag{31}$$

The default values $(\alpha, \beta) = (0.25, 0.2)$ satisfy these conditions. In particular, $\beta = 1$ gives $m_k = \ell_k$, so neither directional update can be triggered.

B Algorithm

The algorithm is described in alg. 1.

C DGR Trajectory Analysis

Figure 4 shows the replanning horizon r_k selected by DGR over the course of a single episode of π_0-LIBERO-Long task (*turn on the stove and put the moka pot on it*), together with the overlap disagreement ℓ_k and its exponential moving average m_k.

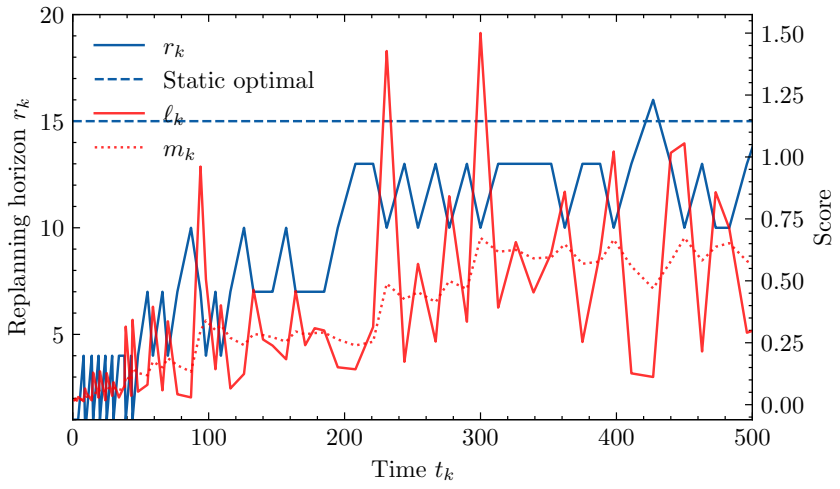

Figure 4: Replanning horizon r_k, overlap disagreement ℓ_k, and its smoothed version m_k as a function of t_k. Evaluated on π_0-LIBERO-long-Task 2 (*turn on the stove and put the moka pot on it.*)

The static optimal horizon (dashed) serves as a reference: DGR does not track it directly but converges to a similar operating range after an initial transient, adapting dynamically as task conditions change throughout the episode. This supports the intended within-episode adaptation.

D Additional Fixed-Horizon Results

We provide task-level fixed-horizon results for π_0 and $\pi_{0.5}$ across all four LIBERO suites. Figures 5 and 6 show the correlation between $\hat{\mathcal{L}}_{\mathrm{o}}(r)$ and task failure rate across replanning horizons, computed per-task. For π_0, correlations are consistently high and positive across most tasks and suites, confirming that the overlap disagreement is a reliable per-task predictor of performance. For $\pi_{0.5}$ (planning horizon $H = 10$), correlations are more variable, likely because of a lower value of H; when negative, the curves are flat.

Figures 7 and 8 compare the minimizer of $\hat{\mathcal{L}}_{\mathrm{o}}(r)$ with the minimizer of the failure rate, at task level. The two minimizer intervals overlap for the large majority of tasks across all suites, validating the proxy at the task level beyond aggregate results.

In Table 4, we can see the success rate per task for π_0-LIBERO. Compared to the suite-level Static Optimal, DGR is better or equal in 30/40 tasks with 12 ties. On the other hand, compared to the task-level Static Optimal, DGR is better or equal in 24/40 tasks with 10 ties. Note that the task-level quantities have a higher uncertainty, so the metrics should be understood as DGR being competitive with an Oracle, rather than being better.

Table 4: Per-task success rate on π_0-LIBERO

Task	Description	DGR	Static	Static Optimal
Goal suite				
1	open the middle drawer of the cabinet	90 ± 4	94 ± 3	100
2	put the bowl on the stove	100	100	100
3	put the wine bottle on top of the cabinet	**100**	98 ± 2	100
4	open the top drawer and put the bowl inside	80 ± 6	$\mathbf{84 \pm 5}$	88 ± 5
5	put the bowl on top of the cabinet	96 ± 3	96 ± 3	94 ± 3
6	push the plate to the front of the stove	$\mathbf{98 \pm 2}$	94 ± 3	96 ± 3
7	put the cream cheese in the bowl	$\mathbf{98 \pm 2}$	96 ± 3	94 ± 4
8	turn on the stove	100	100	100

Continued on next page

Table 4 — continued

Task	Description	DGR	Static	Static Optimal
9	put the bowl on the plate	96 ± 3	$\mathbf{100}$	100
10	put the wine bottle on the rack	$\mathbf{94 \pm 3}$	88 ± 5	92 ± 4
Long suite				
1	put both the alphabet soup and the tomato sauce in the basket	$\mathbf{92 \pm 4}$	84 ± 5	100
2	put both the cream cheese box and the butter in the basket	100	100	92 ± 4
3	turn on the stove and put the moka pot on it	$\mathbf{90 \pm 4}$	80 ± 6	94 ± 3
4	put the black bowl in the bottom drawer of the cabinet and close it	98 ± 2	98 ± 2	82 ± 6
5	put the white mug on the left plate and put the yellow and white mug on the right plate	$\mathbf{82 \pm 5}$	76 ± 6	94 ± 3
6	pick up the book and place it in the back compartment of the caddy	92 ± 4	$\mathbf{94 \pm 3}$	88 ± 4
7	put the white mug on the plate and put the chocolate pudding to the right of the plate	$\mathbf{86 \pm 5}$	84 ± 5	96 ± 2
8	put both the alphabet soup and the cream cheese box in the basket	$\mathbf{96 \pm 3}$	88 ± 5	72 ± 6
9	put both moka pots on the stove	$\mathbf{60 \pm 7}$	8 ± 4	30 ± 6
10	put the yellow and white mug in the microwave and close it	$\mathbf{94 \pm 3}$	80 ± 6	82 ± 5
Object suite				
1	pick up the alphabet soup and place it in the basket	$\mathbf{98 \pm 2}$	92 ± 4	100
2	pick up the cream cheese and place it in the basket	100	100	100
3	pick up the salad dressing and place it in the basket	$\mathbf{100}$	98 ± 2	100
4	pick up the bbq sauce and place it in the basket	$\mathbf{96 \pm 3}$	92 ± 4	100
5	pick up the ketchup and place it in the basket	92 ± 4	$\mathbf{100}$	100
6	pick up the tomato sauce and place it in the basket	$\mathbf{100}$	98 ± 2	96 ± 3
7	pick up the butter and place it in the basket	100	100	100
8	pick up the milk and place it in the basket	100	100	100
9	pick up the chocolate pudding and place it in the basket	100	100	100
10	pick up the orange juice and place it in the basket	100	100	98 ± 2
Spatial suite				
1	pick up the black bowl between the plate and the ramekin and place it on the plate	98 ± 2	$\mathbf{100}$	98 ± 2
2	pick up the black bowl next to the ramekin and place it on the plate	98 ± 2	$\mathbf{100}$	98 ± 2
3	pick up the black bowl from table center and place it on the plate	100	100	98 ± 2
4	pick up the black bowl on the cookie box and place it on the plate	96 ± 3	$\mathbf{98 \pm 2}$	98 ± 2
5	pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate	$\mathbf{96 \pm 3}$	92 ± 4	98 ± 2
6	pick up the black bowl on the ramekin and place it on the plate	$\mathbf{94 \pm 3}$	90 ± 4	98 ± 2
7	pick up the black bowl next to the cookie box and place it on the plate	$\mathbf{100}$	92 ± 4	98 ± 2

Continued on next page

Table 4 — continued

Task	Description	DGR	Static	Static Optimal
8	pick up the black bowl on the stove and place it on the plate	90 ± 4	$\mathbf{96 \pm 3}$	98 ± 2
9	pick up the black bowl next to the plate and place it on the plate	92 ± 4	$\mathbf{98 \pm 2}$	98 ± 2
10	pick up the black bowl on the wooden cabinet and place it on the plate	94 ± 3	94 ± 3	98 ± 2

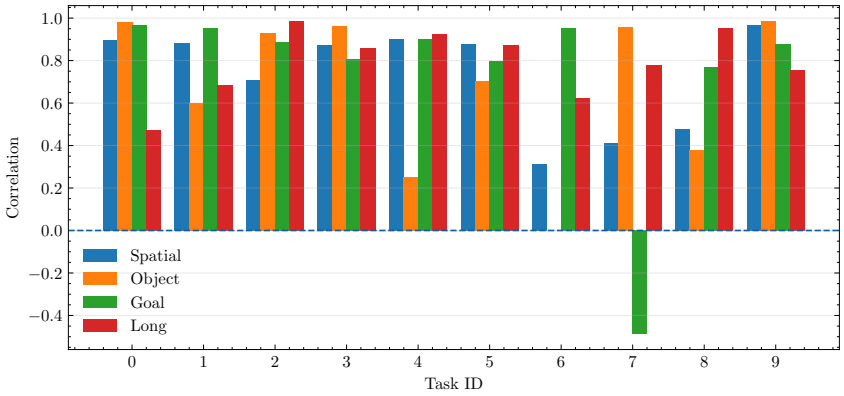

Figure 5: Per-task correlation between overlap disagreement $\hat{\mathcal{L}}_{\mathrm{o}}(r)$ and task failure rate across re-planning horizons, for π_0 on all four LIBERO suites.

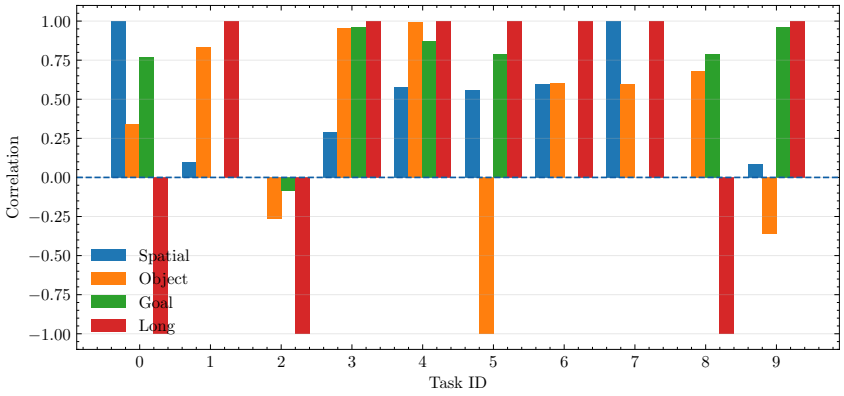

Figure 6: Per-task Spearman correlation for $\pi_{0.5}$ ($H = 10$). Note that whenever the correlation is low or negative, the curves are essentially flat.

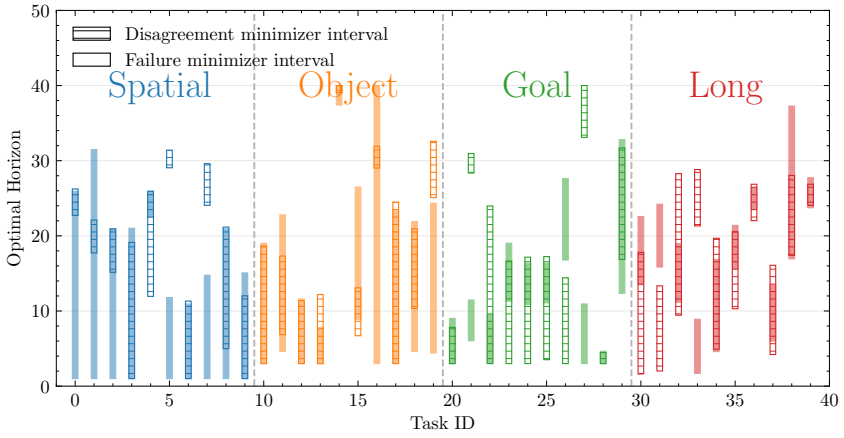

Figure 7: Per-task comparison of the overlap disagreement minimizer interval (hatched) and the failure rate minimizer interval (plain) for π_0. Intervals are estimated using a quantile method (20% of the range); for some tasks the failure rate is essentially flat, thus really wide optimality intervals. Overlap between intervals indicates agreement between the two minimizers.

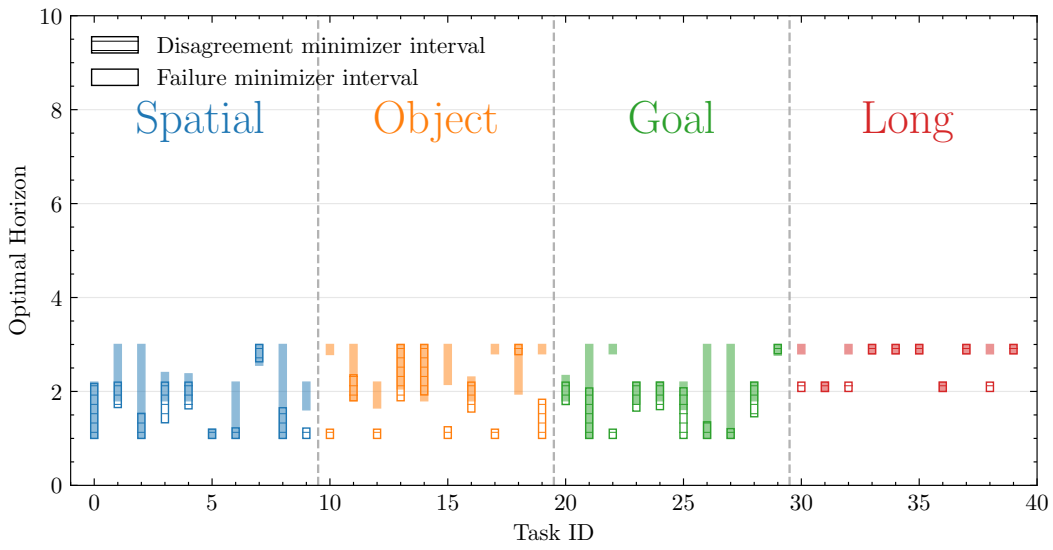

Figure 8: Per-task minimizer comparison for $\pi_{0.5}$.

E RoboCasa Results

We evaluated DGR on a suite based on RoboCasa with GR00T. The specific checkpoint and task details can be found here. The results are shown in Table 5. We observed that DGR occasionally outperforms the static optimal baseline, while it also underperforms on some tasks, resulting in a lower average success rate overall. However, it has a substantially higher average success rate compared to the $r = 1$ static baseline. Given that DGR performs well with GR00T on a closely related robosuite-based benchmark (LIBERO) that differs primarily in embodiment, we conjecture that the primary limitation on RoboCasa is that a humanoid's kinematic groups, such as arms, waist and hands, have heterogeneous drift profiles $\phi_g(j)$ and thus different optimal horizons r_g^*. Figure 9 shows that the per-group overlap disagreements $\ell_k^{(g)}$ are very weakly correlated, evidence that different groups carry uncorrelated signals, and suggesting that there is no common optimal replanning horizon. Computing a single scalar overlap disagreement ℓ_k over all 29 degrees of freedom conflates these groups, producing an adaptation signal that is suboptimal. To solve this employ inpainting. The results are in Table 5. Base DGR wins in 2 out of 20 tasks, Static Optimal in 3/20, and finally DGR+Inpainting wins on 13/20 tasks.

Table 5: Success rates (%) on RoboCasa tasks. Each task was done 50 times, except for inpainting-based DGR which received 500.

Task	$r = 1$	DGR	Static Optimal	DGR+Inpainting
Cup to Drawer & Close	20 ± 6	40 ± 7	38 ± 7	45 ± 2
Potato to Microwave & Close	6 ± 3	24 ± 6	30 ± 6	32 ± 2
Milk to Microwave & Close	44 ± 7	52 ± 7	56 ± 7	57 ± 2
Wine to Cabinet & Close	14 ± 5	40 ± 7	44 ± 7	40 ± 2
Can to Drawer & Close	32 ± 7	38 ± 7	42 ± 7	49 ± 2
Cutting Board to Basket	28 ± 6	54 ± 7	50 ± 7	45 ± 2
Cutting Board to Cardboard Box	28 ± 6	34 ± 7	34 ± 7	43 ± 2
Cutting Board to Pot	36 ± 7	52 ± 7	50 ± 7	68 ± 2
Cutting Board to Tiered Basket	24 ± 6	44 ± 7	36 ± 7	39 ± 2
Placemat to Basket	38 ± 7	44 ± 7	42 ± 7	45 ± 2
Placemat to Bowl	46 ± 7	50 ± 7	58 ± 7	57 ± 2
Placemat to Tiered Shelf	12 ± 5	38 ± 7	38 ± 7	30 ± 2
Plate to Bowl	42 ± 7	38 ± 7	42 ± 7	59 ± 2
Plate to Cardboard Box	36 ± 7	38 ± 7	42 ± 7	48 ± 2
Plate to Pan	38 ± 7	52 ± 7	50 ± 7	57 ± 2
Tray to Cardboard Box	46 ± 7	38 ± 7	46 ± 7	45 ± 2
Tray to Plate	34 ± 7	52 ± 7	56 ± 7	62 ± 2
Tray to Pot	40 ± 7	54 ± 7	58 ± 7	56 ± 2
Tray to Tiered Basket	20 ± 6	18 ± 5	28 ± 6	43 ± 2
Tray to Tiered Shelf	26 ± 6	18 ± 5	28 ± 6	30 ± 2
Mean	31	41	43	**48**

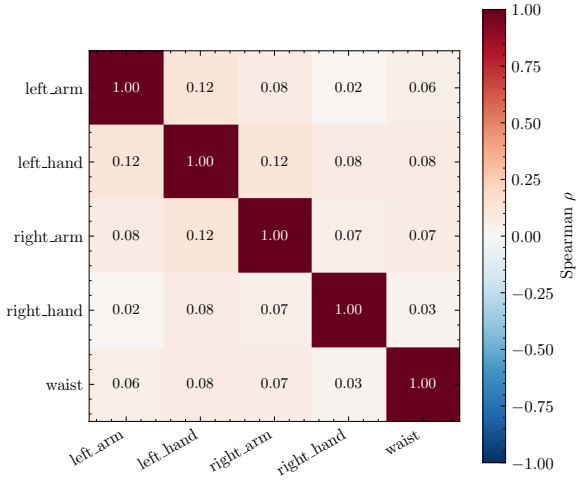

Figure 9: Spearman correlation ρ between per-group overlap disagreement $\ell_k^{(g)}$ for GR00T.

E.1 Multi-Group DGR via Cross-Group Action Inpainting

DGR introduced in Section 4 computes one disagreement score and selects one replanning horizon, which can be restrictive for embodiments with multiple kinematic groups. This is a problem for all the baselines too. For example, the arms, hands, and waist of a humanoid may exhibit different disagreement dynamics and may therefore benefit from different replanning frequencies. Figure 9 shows that the groupwise disagreement signals on RoboCasa are only weakly correlated, motivating independent horizon adaptation.

We consequently extend DGR in two steps. First, each kinematic group maintains an independent disagreement statistic and replanning horizon. Second, when only a subset of groups replans, we use guidance-based action inpainting to condition their newly generated actions on the committed future of the groups that do not replan.

Action-space partition. Let

$$\mathbf{A} \in \mathbb{R}^{H \times D}$$

denote an action chunk produced by the flow-matching policy, where H is the prediction horizon and D is the action dimensionality. We partition the action dimensions into G disjoint kinematic groups $\{\mathcal{I}_g\}_{g=1}^{G}$ satisfying

$$\bigcup_{g=1}^{G} \mathcal{I}_g = \{1, \dots, D\}, \qquad \mathcal{I}_g \cap \mathcal{I}_{g'} = \varnothing \quad \text{for } g \neq g'.$$

For GR00T N1.5 controlling the GR1 humanoid, we use five groups: the left arm (7 DoF), left hand (6 DoF), right arm (7 DoF), right hand (6 DoF), and waist (3 DoF).

Each group g maintains a committed action buffer

$$\mathbf{C}_g \in \mathbb{R}^{H \times |\mathcal{I}_g|},$$

a replanning horizon r_g, an EMA disagreement m_g, and a pointer e_g recording the number of actions executed from $\mathbf{C}_g$. Group g becomes due for replanning when $e_g = r_g$. At a replanning event, we define

$$\mathcal{R} = \{g : e_g = r_g\}, \qquad \mathcal{B} = \{1, \dots, G\} \setminus \mathcal{R},$$

where $\mathcal{R}$ contains the replanning groups and $\mathcal{B}$ contains the buffered, or held, groups.

Per-group DGR update. Let $\widetilde{\mathbf{A}}$ denote the newly generated joint chunk. For every $g \in \mathcal{R}$, we compute the groupwise overlap disagreement

$$\ell^{(g)} = \sum_{j=1}^{L-1} \left\| \widetilde{\mathbf{A}}[j, \mathcal{I}_g] - \mathbf{C}_g[e_g + j, :] \right\|_2^2. \tag{32}$$

Because every group compares its score only with its own moving average, the scores are not aggregated across action groups.

We apply the DGR update independently:

$$r_g^{+} = \mathrm{clip}_{[1, H-L]} \begin{cases} r_g - \Delta, & \ell^{(g)} > (1+\alpha) m_g, \\ r_g + \Delta, & \ell^{(g)} < (1-\alpha) m_g, \\ r_g, & \text{otherwise.} \end{cases} \tag{33}$$

The corresponding moving average is

$$m_g^{+} = (1-\beta) m_g + \beta \ell^{(g)}. \tag{34}$$

For the first observation of a group, we initialize $m_g = \ell^{(g)}$ and leave r_g unchanged. After replanning, we replace the committed buffer of group g by

$$\mathbf{C}_g \leftarrow \widetilde{\mathbf{A}}[:, \mathcal{I}_g], \qquad e_g \leftarrow 0, \qquad g \in \mathcal{R}.$$

The buffers and pointers of held groups remain unchanged.

Asynchronous execution. After updating the replanning groups, the number of actions that can be executed before the next group becomes due is

$$n_{\text{exec}} = \min_{g=1,\dots,G} (r_g - e_g). \tag{35}$$

The composite action at step j is assembled directly from the group-specific buffers:

$$\mathbf{a}_{t+j}[\mathcal{I}_g] = \mathbf{C}_g[e_g + j, :], \qquad j = 0, \dots, n_{\text{exec}} - 1. \tag{36}$$

After execution, every pointer is incremented by n_{exec}. Equation 35 ensures that no group executes beyond its selected horizon.

E.1.1 Cross-Group Action Inpainting

Independently replanning each group introduces a compatibility problem. A replanning arm, for example, may generate a grasping motion that is incompatible with the trajectory already committed by a held hand. We address this using a training-free adaptation of the ΠGDM guidance procedure employed by real-time chunking Black et al. (2025b). The method encourages newly generated actions for groups in $\mathcal{R}$ to remain compatible with the committed futures of groups in $\mathcal{B}$.

Guidance target. Before generating a new chunk, the remaining time until the next held group replans is

$$q = \begin{cases} \min_{g \in \mathcal{B}} (r_g - e_g), & \mathcal{B} \neq \varnothing, \\ 0, & \mathcal{B} = \varnothing. \end{cases} \tag{37}$$

We construct a target $\mathbf{Y} \in \mathbb{R}^{H \times D}$ and binary mask $\mathbf{W} \in \{0,1\}^{H \times D}$. For every held group,

$$\mathbf{Y}[j, \mathcal{I}_g] = \mathbf{C}_g[e_g + j, :], \qquad \mathbf{W}[j, \mathcal{I}_g] = 1, \quad g \in \mathcal{B}, \quad j = 0, \ldots, q-1. \tag{38}$$

All remaining mask entries are zero. Thus, guidance is applied only where a held group has a committed future. We do not constrain actions beyond q, because those groups may themselves replan at that point.

Guided flow integration. Let $\mathbf{Z}^\tau$ denote the noisy action chunk at normalized flow time $\tau \in [0,1]$, initialized as

$$\mathbf{Z}^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Given the policy velocity field $v_\theta(\mathbf{Z}^\tau, o_t, \tau)$, the one-step estimate of the clean action chunk is

$$\widehat{\mathbf{A}}^{\,1} = \mathbf{Z}^\tau + (1-\tau) v_\theta(\mathbf{Z}^\tau, o_t, \tau). \tag{39}$$

We define the masked guidance residual

$$\mathbf{E}^\tau = \mathbf{W} \odot \left(\mathbf{Y} - \widehat{\mathbf{A}}^{\,1} \right). \tag{40}$$

The ΠGDM vector–Jacobian product is computed as

$$\mathbf{g}^\tau = \mathbf{W} \odot \nabla_{\mathbf{Z}^\tau} \left\langle \mathrm{sg}(\mathbf{E}^\tau), \widehat{\mathbf{A}}^{\,1} \right\rangle, \tag{41}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient.

The residual mask inside Equation 41 is inherited from ΠGDM. The additional multiplication by $\mathbf{W}$ after the vector–Jacobian product is our modification: it restricts the direct gradient correction to held action dimensions. The action dimensions of replanning groups continue to evolve through the policy velocity field and respond indirectly as the jointly processed held dimensions are guided toward their committed values.

To avoid a conflict with the EMA coefficient β, we denote the maximum guidance strength by $\gamma_{\max}$. The clipped guidance schedule is

$$\gamma(\tau) = \min\left(\gamma_{\max}, \frac{1-\tau}{\tau\, \rho(\tau)^2} \right), \qquad \rho(\tau)^2 = \frac{(1-\tau)^2}{\tau^2 + (1-\tau)^2}, \tag{42}$$

for $\tau > 0$, with $\gamma(0) = \gamma_{\max}$.

Using N_{flow} Euler steps, the guided update is

$$\mathbf{Z}^{\tau + 1/N_{\text{flow}}} = \mathbf{Z}^\tau + \frac{1}{N_{\text{flow}}} \left[v_\theta(\mathbf{Z}^\tau, o_t, \tau) + \gamma(\tau) \mathbf{g}^\tau \right]. \tag{43}$$

The final sample $\widetilde{\mathbf{A}} = \mathbf{Z}^1$ supplies the new buffers for groups in $\mathcal{R}$. Groups in $\mathcal{B}$ continue executing their original buffers rather than the corresponding generated values. Consequently, their committed actions are enforced exactly, while guidance encourages the newly replanned groups to generate compatible motions. When $\gamma_{\max} = 0$, the procedure reduces to independent groupwise DGR without cross-group guidance.

Unlike standard DGR, guided inpainting requires a vector-Jacobian product at every flow step, thus resulting in higher computational cost for the replanning policy.

Algorithm 2 Multi-Group DGR with Cross-Group Inpainting

Require: Flow policy π_θ, groups $\{\mathcal{I}_g\}_{g=1}^{G}$, $H, L, \alpha, \beta, \Delta$, guidance cap $\gamma_{\max}$

Generate initial joint chunk $\mathbf{A} \sim \pi_\theta(\cdot \mid o_0)$
for $g = 1, \dots, G$ **do**
  $\mathbf{C}_g \leftarrow \mathbf{A}[:, \mathcal{I}_g]$
  $r_g \leftarrow 1, e_g \leftarrow 0, m_g \leftarrow \perp$
end for
while episode not terminated **do**
  $n_{\text{exec}} \leftarrow \min_g(r_g - e_g)$
  Execute n_{exec} composite actions using Equation 36
  $e_g \leftarrow e_g + n_{\text{exec}}$ for every g
  if episode terminated **then**
    break
  end if
  $\mathcal{R} \leftarrow \{g : e_g = r_g\}$
  $\mathcal{B} \leftarrow \{1, \dots, G\} \setminus \mathcal{R}$
  Construct $(\mathbf{Y}, \mathbf{W})$ using Equations 37–38
  Generate $\widetilde{\mathbf{A}}$ using guided flow integration 43
  for all $g \in \mathcal{R}$ **do**
    Compute $\ell^{(g)}$ using Equation 32
    if $m_g = \perp$ **then**
      $m_g \leftarrow \ell^{(g)}$
    else
      Update r_g using Equation 33
      Update m_g using Equation 34
    end if
    $\mathbf{C}_g \leftarrow \widetilde{\mathbf{A}}[:, \mathcal{I}_g]$
    $e_g \leftarrow 0$
  end for
end while

F Adversarial Patch Generation

We optimize a universal untargeted patch against π_0 using observations sampled from LIBERO. Given an observation o, patch transformation τ, and shared flow-matching noise ϵ, the patch p maximizes the distance from the clean action prediction,

$$\max_{p \in [0,1]} \mathbb{E}_{o,\tau} \left[|\pi_\theta(\tau(o, p); \epsilon) - \pi_\theta(o; \epsilon)|_2^2 \right]. \tag{44}$$

We optimize p using signed gradients while randomly varying its position and scale, and apply it only to the external-camera image. The complete implementation and hyperparameters are provided in the repository. DGR provides its clearest advantage on LIBERO-Spatial, improving by 20 percentage points over the clean-tuned fixed horizon and by 5 points over the attack-tuned horizon. On Object, Goal, and Long, it performs similarly to the fixed policies. In particular, all methods reach only 33% on LIBERO-Long, indicating that horizon adaptation cannot recover failures caused by severe degradation of the base policy. These results therefore support robustness to a shifted optimal horizon, rather than claiming that DGR constitutes a general adversarial defense.

G Real-World Push-T Evaluation

We evaluate SmolVLA Shukor et al. (2025) on a physical CloudGripper Push-T setup observed by a fixed fisheye camera. Images are undistorted and mapped to the policy's 96×96 input space. The policy predicts 50-step chunks of absolute two-dimensional pusher setpoints. We compare fixed execution horizons with DGR. For each method, 20 episodes are executed, resetting the environment in between. Success is evaluated by the target coverage attained per episode, normalized as $R = \text{clip}(c/0.95, 0, 1)$. If it is greater than 0.5, then an episode is considered successful. In simulation the finetuned version of SmolVLA reached an average score of ~ 0.6. Figure 10 illustrates

Table 6: Success rate (%) under the adversarial patch. The clean-tuned fixed horizon is selected without the attack, whereas the attack-tuned horizon is selected using attacked rollouts.

Suite	DGR	Clean-tuned r^*	Attack-tuned r^*
Spatial	**74**	54	69
Object	53	53	**55**
Goal	51	**52**	**52**
Long	33	33	33

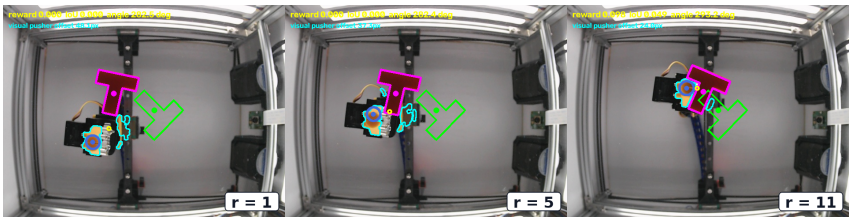

Figure 10: CloudGripper Push-T task

that the replanning horizon adapts substantially even within a single task execution, not just among tasks. In particular, it is low when several different trajectories are possible, and high near contact. $r = 1$, $r = 10$, $r = 40$, and DGR achieve 10%, 65%, 20%, and 70% success rate, respectively. With 20 trials per condition, these correspond to 2, 13, 4, and 14 successful episodes. The poor performance at both $r = 1$ and $r = 40$ reproduces the expected tradeoff between frequent replanning and extended open-loop execution on a physical system. DGR attains the highest numerical success rate, although its difference from $r = 10$ is only one episode. The result should therefore be interpreted as evidence that DGR can match the best-performing tested fixed horizon without selecting that horizon in advance, rather than as evidence of a statistically significant improvement over $r = 10$

H Detailed Computational Cost Analysis

Let F_{VLM} and F_{act} denote the floating point operations (FLOPs) required for one forward pass through the VLM backbone and action head respectively, and let N be the number of trajectory samples used by AAC. Let K_{AAC} and K_{DGR} denote the expected number of replanning steps per episode for AAC and DGR, respectively.

DGR requires one VLM call and one action head call per replanning step, plus a negligible vector subtraction for the overlap disagreement:

$$\mathcal{F}_{\mathrm{DGR}} = K_{\mathrm{DGR}}\,(F_{\mathrm{VLM}} + F_{\mathrm{act}}). \tag{45}$$

Each of the N samples requires a full forward pass:

$$\mathcal{F}^{\mathrm{full}}_{\mathrm{AAC}} = K_{\mathrm{AAC}} \cdot N\,(F_{\mathrm{VLM}} + F_{\mathrm{act}}). \tag{46}$$

The FLOPs ratio per call is therefore:

$$\frac{\mathcal{F}^{\mathrm{full}}_{\mathrm{AAC}}/K_{\mathrm{AAC}}}{\mathcal{F}_{\mathrm{DGR}}/K_{\mathrm{DGR}}} = N, \tag{47}$$

independent of architecture.

In case of cached implementation, the VLM runs once and its key-value cache is reused across all N action head samples:

$$\mathcal{F}^{\mathrm{cached}}_{\mathrm{AAC}} = K_{\mathrm{AAC}}\,(F_{\mathrm{VLM}} + N \cdot F_{\mathrm{act}}). \tag{48}$$

Defining $\rho := F_{\mathrm{act}}/F_{\mathrm{VLM}}$ as the action-head-to-VLM FLOPs ratio, the per-call FLOPs ratio becomes:

$$\left.\frac{\mathcal{F}^{\mathrm{cached}}_{\mathrm{AAC}}/K_{\mathrm{AAC}}}{\mathcal{F}_{\mathrm{DGR}}/K_{\mathrm{DGR}}}\right|_{\text{per call}} = \frac{F_{\mathrm{VLM}} + N \cdot F_{\mathrm{act}}}{F_{\mathrm{VLM}} + F_{\mathrm{act}}} = \frac{1 + N\rho}{1 + \rho}. \tag{49}$$

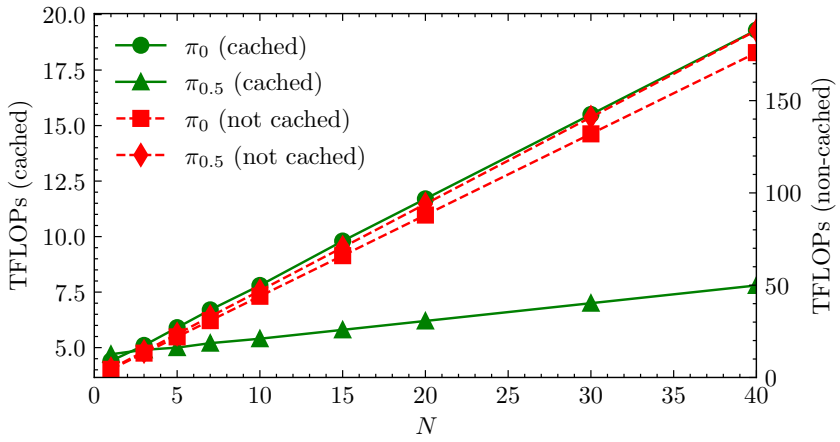

Figure 11: Number of FLOPs as a function of N for the two possible implementations of AAC: cached and non-cached

Since ρ can be estimated from profiling, this equation yields the same per-call ratios reported in Table 3 of the main paper, confirming the validity of this cost analysis.

In addition, we profiled both implementations for varying values of N to validate our analysis; Figure 11 displays FLOPs per call as function of N.