Enhancing Control Policy Smoothness by Aligning Actions with Predictions from Preceding States

Kyoleen Kwak and Hyoseok Hwang*

Department of Artificial intelligence, Kyung Hee University, Republic of Korea
{2007kkl,hyoseok}@khu.ac.kr

Abstract

Deep reinforcement learning has proven to be a powerful approach to solving control tasks, but its characteristic high-frequency oscillations make it difficult to apply in real-world environments. While prior methods have addressed action oscillations via architectural or loss-based methods, the latter typically depend on heuristic or synthetic definitions of state similarity to promote action consistency, which often fail to accurately reflect the underlying system dynamics. In this paper, we propose a novel loss-based method by introducing a transition-induced similar state. The transition-induced similar state is defined as the distribution of next states transitioned from the previous state. Since it utilizes only environmental feedback and actually collected data, it better captures system dynamics. Building upon this foundation, we introduce Action Smoothing by Aligning Actions with Predictions from Preceding States (ASAP), an action smoothing method that effectively mitigates action oscillations. ASAP enforces action smoothness by aligning the actions with those taken in transition-induced similar states and by penalizing second-order differences to suppress high-frequency oscillations. Experiments in Gymnasium and Isaac-Lab environments demonstrate that ASAP yields smoother control and improved policy performance over existing methods.

Code — https://github.com/AIRLABkhu/ASAP

Introduction

Reinforcement learning (RL) has seen tremendous advances in recent years (Wang et al. 2024). Beginning with discrete methods, RL has evolved into Deep Reinforcement Learning (DRL) through the integration of deep neural networks and has been extended to continuous action spaces, becoming a leading solution for control problems (Williams 1992; Lillicrap et al. 2015). Building on these advances, RL is widely applied to robot and machine control with impressive results (Levine et al. 2016), and these efforts aim to deploy learned policies in real-world settings (Tobin et al. 2017).

Despite these successes, deploying RL in real-world settings remains challenging, with high-frequency action oscillations posing a significant obstacle. When combined with hardware constraints in the real-world, such action oscillations can dramatically reduce component lifespan and cause excessive wear (de la Presilla et al. 2023). In addition, action oscillations produce non-smooth actions, and these non-smooth actions can directly cause poor user experience, or in severe cases, safety problems. This issue arises because the actor is overly sensitive to minor state perturbations, which yields large action deviations (Chen et al. 2021).

Research on mitigating action oscillations in RL has primarily focused on achieving smoother policy outputs by regulating output variation with respect to input fluctuations. Existing approaches fall into two main categories: architectural methods and loss penalty methods. Architectural methods aim to prevent abrupt output changes by modifying the network structure, typically applying Spectral Normalization (Takase et al. 2022) or Multi-dimensional Gradient Normalization (Song et al. 2023) to the actor network. However, these methods introduce additional computational cost during inference and often suffer from significant performance variability across environments (Christmann et al. 2024).

On the other hand, loss penalty methods incorporate oscillation regularization terms directly into the policy loss. This approach does not alter the actor structure and thus avoids extra inference-time computation. A key idea behind this method is to enforce action consistency across similar states, making the definition of similar states crucial. Initial approaches employed fixed distributions (Mysore et al. 2021), while more recent methods introduced adaptive boundaries (Kobayashi 2022). However, these approaches rely on heuristic neighborhood definitions and fill the region with synthetic rather than observed states. As a result, the similar states do not accurately reflect the actual state distribution. This mismatch undermines the theoretical guarantees and leads to significant declines in performance.

To overcome these limitations, in this paper, we propose a novel loss penalty based RL action smoothing method called regulate action oscillation via **A**ction **S**moothing by **A**ligning Actions with **P**redictions from Preceding States (ASAP). ASAP introduces a transition-induced definition of similar states, under the assumption that states transitioning from the same previous state are similar. We theoretically prove that this definition induces a bounded neighborhood, thereby providing a principled foundation for action smoothing based on similar states. Building on this foun-

*Corresponding author (hyoseok@khu.ac.kr)

dation, ASAP introduces a spatial term that constrains action changes. This spatial term constrains action changes across similar states by reducing the discrepancy between a predicted next action from the preceding state and the actual action taken. In addition, ASAP adopts a temporal term that penalizes second-order differences to suppress high-frequency oscillations (Lee et al. 2024). Unlike prior methods that rely on synthetic or heuristically defined neighborhoods, our approach is trained exclusively on empirically collected transitions, ensuring fidelity to the real state distribution and alignment with system dynamics.

We evaluated ASAP's policy performance and oscillation in Gymnasium (Towers et al. 2024), and Isaac-Lab environments (Mittal et al. 2023). Through experiments with various RL algorithms, we confirmed that ASAP effectively reduces action oscillations across multiple tasks while preserving the performance compared to other methods.

This work's main contributions are as follows:

- We introduce **ASAP**, a new action smoothing method that combines the action constraint using transition-induced similar state with a penalty on second-order action variations.
- We define similar states derived from the maximum-change bound of the transition dynamics, and theoretically prove that they form a bounded neighborhood for action smoothing.
- Extensive experiments demonstrate that ASAP effectively suppresses action oscillations while preserving policy performance over existing methods.

Related Work

Many studies have focused on reducing the Lipschitz constant of the actor network to constrain its output changes to mitigate the action oscillations. Related studies are broadly categorized into two approaches:

Architectural Methods. Architectural methods alter the network itself to satisfy Lipschitz constraints. Gogianu et al. showed that Spectral Normalization can stabilize reinforcement learning by controlling Lipschitz smoothness (Gogianu et al. 2021). LipsNet (Song et al. 2023) enforces global and local Lipschitz continuity via Multi-dimensional Gradient Normalization by regularizing the Jacobian's operator norm. LipsNet++ (Song et al. 2025) adds a Fourier filter and a Lipschitz controller with Jacobian regularization to decouple noise and smoothness.

Loss Penalty Methods. Loss penalty methods add Lipschitz regularizers directly to the RL loss. Conditioning for Action Policy Smoothness (CAPS) (Mysore et al. 2021) adds temporal smoothing between adjacent states and spatial smoothing over a fixed range neighborhood of similar states to limit actor output variation. Local Lipschitz Continuous Constraint (L2C2) (Kobayashi 2022) dynamically adjusts the range of similar states based on the current state, thereby implementing the concept of local Lipschitz continuity in reinforcement learning. Gradient-based CAPS (Grad-CAPS) (Lee et al. 2024) proposed a new approach by applying Lipschitz constraints on derivatives instead of conventional Lipschitz constraints from a temporal perspective. All of these methods share the network architecture of the base model and only increase its loss, so they incur additional computation during training while performing operations identical to the base model in inference.

Background

Lipschitz Continuity

Definition 1 (Global Lipschitz Continuity)**.** *A function $f : \mathcal{X} \to \mathcal{Y}$ is called Lipschitz continuous with global Lipschitz constant $K_{\text{glob}} \in \mathbb{R}_+$ if, for all $x^{(1)}, x^{(2)} \in \mathcal{X}$,*

$$d_Y(f(x^{(1)}),\ f(x^{(2)})) \ \leq\ K_{\text{glob}}\, d_X(x^{(1)},\ x^{(2)}), \qquad (1)$$

Definition 2 (Local Lipschitz Continuity)**.** *A function $f : \mathcal{X} \to \mathcal{Y}$ is locally Lipschitz continuous if, for every point $x_i \in \mathcal{X}$, there exist a neighborhood $U_i \subseteq \mathcal{X}$ and a local Lipschitz constant $K_{\text{loc}}(U_i) \in \mathbb{R}_+$ such that*

$$d_Y(f(x_i^{(1)}),\ f(x_i^{(2)})) \ \leq\ K_{\text{loc}}(U_i)\, d_X(x_i^{(1)},\ x_i^{(2)}), \quad \forall\, x_i^{(1)},\ x_i^{(2)} \in U_i, \qquad (2)$$

here, d_X and d_Y are the distance metrics on the spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively.

Lipschitz constraint means constraining the Eq. 1 or Eq. 2 to be satisfied. A lower Lipschitz constant K (whether K_{glob} or K_{loc}) implies smaller output variations for given input changes, enhancing the robustness to noise and resulting in smoother outputs. Since a single global constant cannot optimally fit all regions of the input space, a global Lipschitz constraint can prevent reaching the optimal solution.

Lipschitz Constraints in Reinforcement Learning

This section describes how Lipschitz constraints have been applied in reinforcement learning to achieve action smoothness. After CAPS decomposed Lipschitz constraints for policy into temporal and spatial constraints, most subsequent works have extended their methods based on these two axes.

Temporal Smoothness. For every timestep t in an episode, for each consecutive input sequence, the network's outputs satisfy the Lipschitz constraint such as

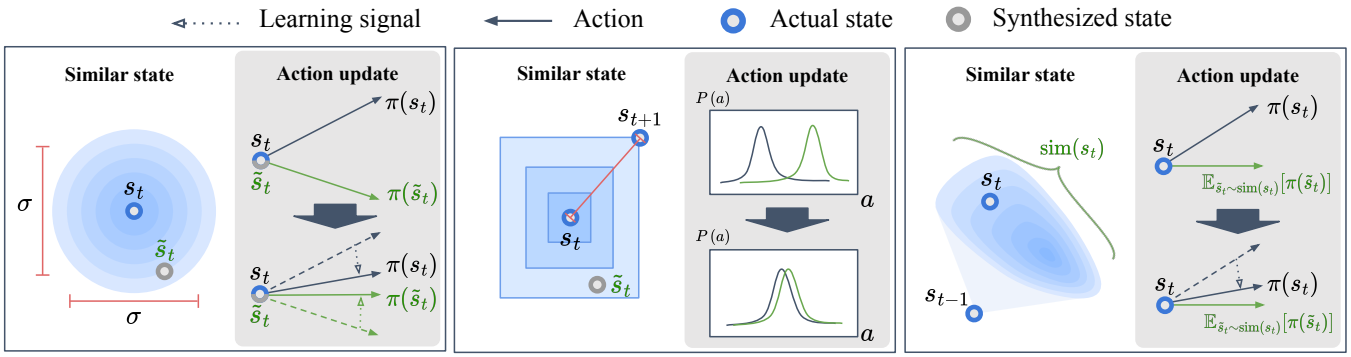
$$d_A(f(s_t),\ f(s_{t+1})) \ \leq\ K\, d_T(s_t,\ s_{t+1}) \ =\ K, \qquad (3)$$

where $s_t \in \mathcal{S}$ is the state, $a_t = f(s_t) \in \mathcal{A}$ is the action, and $f\colon \mathcal{S} \to \mathcal{A}$ is the actor. Here, d_A measures distance in the action space $\mathcal{A}$, and d_T is the temporal distance metric. In reinforcement learning, since the intervals between timesteps are typically assumed to be uniform, the temporal distance metric d_T is fixed to 1.

Spatial Smoothness. For each state s_t, the network satisfies a Lipschitz constraint:

$$d_A(f(s_t),\ f(\tilde{s}_t)) \leq K\, d_S(s_t,\ \tilde{s}_t), \qquad (4)$$

where d_A and d_S denote distances in the action and state spaces, and $\tilde{s}_t$ is a similar state. The definition of $\tilde{s}_t$ varies by method (Figure 1): CAPS samples from $\mathcal{N}(s_t, \sigma)$, enforcing global Lipschitz constraint, while L2C2 sets $\tilde{s}_t = s_t + (s_{t+1} - s_t) \cdot u$ to reflect local Lipschitz constraint. However, because neither definition matches the true state distribution, additional synthetic states must be generated to populate the similar state region.



(a) CAPS (b) L2C2 (c) ASAP

Figure 1: Definition of similar states and action update mechanisms for (a) CAPS, (b) L2C2, and (c) ASAP. (a) CAPS samples states from a Gaussian around the current state and enforces action consistency between real and sampled states. (b) L2C2 constructs similar states by projecting the difference between the next state and the current state in multiple directions and minimizes divergence between the similar and actual policy distributions. (c) ASAP uses the environment's transition distribution to define similar states and aligns the action with the expected policy output under this distribution.

Method

Overview

We define transition-induced similar states via the transition function and demonstrate that they form a bounded neighborhood, providing a basis for spatial regularization. This yields a Lipschitz constraint on the composite function $f \circ T$, from which a spatial loss for the actor is derived. Based on this foundation, we propose ASAP, which (i) enforces action consistency through a spatial loss and (ii) penalizes second-order action differences via a temporal loss. We also describe the training procedure of our method.

Similar State Based on Transition Distribution

We define the distribution of the next states transitioning from the same previous state s_t as the distribution of similar states, as shown in Figure 1c.

Definition 3 (Similar State Distribution)**.** *Given a state s_t, the distribution of similar states is defined as*

$$\mathrm{sim}(s_t) \;=\; P\big(\cdot \mid s_{t-1}\big), \tag{5}$$

where $P(\cdot \mid s_{t-1})$ denotes the transition distribution from s_{t-1}.

To show that the above definition yields a locally bounded neighborhood suitable for imposing a local Lipschitz constraint, we introduce the following assumptions.

Assumption 1 (Local Lipschitz Continuity of the Transition Function with respect to Noise)**.** *For the state transition function*

$$s_t = T(s_{t-1}, a_{t-1}, \xi_{t-1}), \tag{6}$$

we assume that, for each state-action pair (s_{t-1}, a_{t-1}), the transition function is Lipschitz continuous with respect to the noise $\xi_t \in \Xi$ acting on that pair, i.e.,

$$\begin{aligned} &d_S(T(s_{t-1},\, a_{t-1},\, \xi_{t-1}^{(1)}),\; T(s_{t-1},\, a_{t-1},\, \xi_{t-1}^{(2)})) \\ &\quad\le K_\xi(s_{t-1},\, a_{t-1})\, d_\Xi(\xi_{t-1}^{(1)},\, \xi_{t-1}^{(2)}), \end{aligned} \tag{7}$$

where $K_\xi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ is the local noise-sensitivity constant at $(s_{t-1},\, a_{t-1})$, and d_Ξ measures distance in the noise space Ξ.

The real-world dynamics corresponding to the transition function $T(\cdot)$ in Assumption 1 are typically C^1 under most conditions, and this C^1 property is commonly assumed in many system and controller designs (Spong, Hutchinson, and Vidyasagar 2006; Featherstone 2008; Camacho and Alba 2013). It is also well known that any C^1 function is locally Lipschitz continuous (Sontag 1998).

Assumption 2 (Bounded Noise)**.** *The noise ξ_t satisfies a hard bound $d_\Xi(\xi_t,\, 0) \le \sigma_\xi$.*

Although real-world noise is not strictly bounded, sensor filtering, digital clamping, and physical constraints keep it effectively within limits, which underpins many robust/optimal control theories (Doyle 1996; Camacho and Alba 2013).

Lemma 1 (Spatially Bounded Neighborhood)**.** *Under Assumptions 1–2, any two states $s_t^{(1)},\; s_t^{(2)} \sim \mathrm{sim}(s_t)$ satisfy*

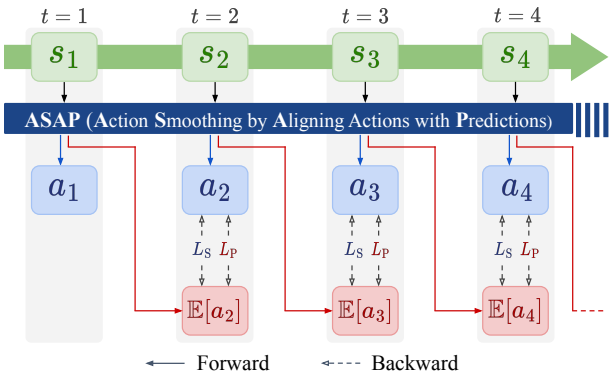
$$d_S(s_t^{(1)},\; s_t^{(2)}) \le 2K_\xi(s_{t-1},\, a_{t-1})\sigma_\xi. \tag{8}$$

Proof. By the Lipschitz continuity assumption on the transition function,

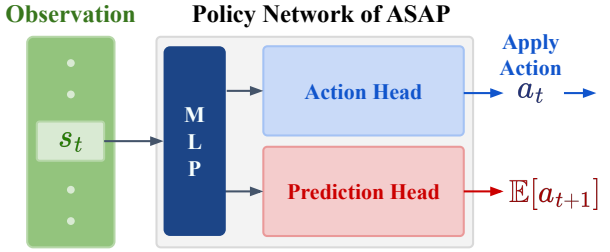
$$\begin{aligned} &d_S(s_t^{(1)},\; s_t^{(2)}) \\ &= d_S(T(s_{t-1},\, a_{t-1},\, \xi_{t-1}^{(1)}),\; T(s_{t-1},\, a_{t-1},\, \xi_{t-1}^{(2)})) \\ &\le K_\xi(s_{t-1},\, a_{t-1})\, d_\Xi(\xi_{t-1}^{(1)},\, \xi_{t-1}^{(2)}). \end{aligned} \tag{9}$$

Under the assumption of bounded noise $d_\Xi(\xi_{t-1},\, 0) \le \sigma_\xi$, we have $d_\Xi(\xi_{t-1}^{(1)},\; \xi_{t-1}^{(2)}) \le 2\sigma_\xi$, which yields the desired result. □

Therefore, according to Definition 3 and Lemma 1, the distance between all similar state pairs is guaranteed to be bounded by $2K_\xi\sigma_\xi$, forming a bounded region suitable for applying the local Lipschitz constraint.



(a)



(b)

Figure 2: (a) The update procedure of ASAP and (b) The implementation architecture of ASAP.

Furthermore, by Definition 3, our similar state distribution coincides exactly with the true transition kernel $P_{\text{real}}(\cdot \mid s_{t-1})$, thereby ensuring fidelity to the system dynamics.

Composite Function Based Lipschitz Constraint

Theorem 1 (Composite Lipschitz Constraint)**.** *Under Assumptions 1-2, for all* $(s_{t-1}, a_{t-1}) \in \mathcal{S} \times \mathcal{A}$ *and* $\xi_{t-1}^{(1)}, \xi_{t-1}^{(2)} \in \Xi$*, it holds that*

$$d_A(f \circ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(1)}),\ f \circ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(2)})) \leq K\, d_S(T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(1)}),\ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(2)})), \quad \forall \xi_1, \xi_2 \in \Xi. \tag{10}$$

Therefore, the composite function $f \circ T$ *is locally Lipschitz continuous in the noise* ξ*, with local Lipschitz constant*

$$K_{\text{comp}}(s_{t-1}, a_{t-1}) = K\, K_\xi(s_{t-1}, a_{t-1}). \tag{11}$$

In this theorem, T represents the transition function, and f represents the actor.

Theorem 1 can be reduced to loss optimization problem such as

$$L = \left\| \pi_\phi(s_t) - \mathbb{E}_{\tilde{s}_t \sim P(\cdot \mid s_{t-1})}\left[\pi_\phi(\tilde{s}_t)\right] \right\|_2^2. \tag{12}$$

All proofs for this paper can be found in the Appendix A. $\mathbb{E}_{\tilde{s}_t \sim P(\cdot \mid s_{t-1})}[\pi_\phi(\tilde{s}_t)]$ denotes the expectation of the policy outputs over all next similar states $\tilde{s}_t$ reachable from the predecessor state s_{t-1}. In other words, L is designed to minimize the difference between the policy output $\pi_\phi(s_t)$ for a state s_t randomly sampled from the transition distribution $s_t \sim P(\cdot \mid s_{t-1})$, and the average policy output over the same distribution, $\mathbb{E}_{\tilde{s}_t \sim P(\cdot \mid s_{t-1})}\left[\pi_\phi(\tilde{s}_t)\right]$. This mechanism propagates the local Lipschitz property of the transition function T through the composite function $f \circ T$, thereby enforcing the local Lipschitz constraint on the actor network f during training.

Action Smoothing via Predictions from Preceding States

In this section, we introduce ASAP, a novel RL action smoothing method using loss-penalty that integrates the spatial smoothness term grounded in the similar state definition based on transition (Definition 3) with the temporal smoothness term proposed by Grad-CAPS (Lee et al. 2024). We define the ASAP policy loss as

$$J_{\pi_\phi}^{\text{ASAP}} = J_{\pi_\phi} + \lambda_S L_S + \lambda_P L_P + \lambda_T L_T, \tag{13}$$

which comprises four terms. J_{π_ϕ} denotes the standard RL actor loss used in algorithms such as Proximal Policy Optimization (PPO) (Schulman et al. 2017) and Soft Actor-Critic (SAC) (Haarnoja et al. 2018).

The spatial loss L_{S} is defined as

$$L_{\text{S}} = \left\| \pi_\phi(s_t) - \texttt{stopgrad}(\pi_{\text{P}}(s_{t-1})) \right\|_2^2, \tag{14}$$

which, as shown in Figure 1c and 2a, minimizes the difference between the current action $\pi_\phi(s_t)$ and the next expected action $\pi_{\text{P}}(s_{t-1})$ predicted from the previous state s_{t-1}. The prediction loss L_{P} is expressed as

$$L_{\text{P}} = \left\| \pi_{\text{P}}(s_{t-1}) - \texttt{stopgrad}(\pi_\phi(s_t)) \right\|_2^2, \tag{15}$$

and trains the predictor π_{P} to closely mimic the policy output $\pi_\phi(s_t)$, illustrated in Figure 2a. Because π_ϕ generates actual actions while π_P provides target action expectations, they differ in objectives, learning signals, and influencing factors and thus require separate learning strengths. Therefore, we fix the target values, divide the loss into Eq. 14 and Eq. 15, and assign distinct strength parameters to each component.

L_T is the temporal Lipschitz penalty directly adopted from the Grad-CAPS (Lee et al. 2024) temporal loss and defined as

$$L_{\text{T}} = \left\| \frac{a_{t+1} - 2a_t + a_{t-1}}{\tanh(a_{t+1} - a_{t-1}) + \epsilon} \right\|_2^2, \tag{16}$$

where ϵ is small positive constant added for numerical stability to avoid division by zero. L_{T} constrains the second-order change in action over time, and the denominator $\tanh(a_{t+1} - a_{t-1})$ provides robustness to variations in the action scale. The second-order difference enables more flexible action changes while maintaining stability. The regularization strength of each term is controlled by the hyperparameters $\lambda_S, \lambda_P, \lambda_T > 0$.

Training Method. Predictor training faces a "moving target" issue, as shifting outputs hinder smoothness optimization and can degrade returns. Although soft updates help stabilize training (Lillicrap et al. 2015; Fujimoto, Hoof, and

Meger 2018), their lag in adapting to rapidly changing distributions can impair performance in on-policy methods like PPO. This issue can be addressed by increasing the number of parallel environments to collect more data under the same policy and by reducing the predictor loss weight λ_P.

Experiment

Experimental Setup

Simulation Framework. We evaluated ASAP using two simulation frameworks: **Gymnasium** (Towers et al. 2024) and **Isaac-Lab** (Mittal et al. 2023). Gymnasium, integrated with the MuJoCo (Todorov, Erez, and Tassa 2012) is a widely adopted benchmark for continuous RL; we used it to empirically validate ASAP's theoretical foundations and evaluate performance. To assess applicability to realistic robotic scenarios, we assessed our method on Isaac-Lab, a high-fidelity simulator.

Implementation Details. All Gymnasium experiments were implemented using Stable Baselines3 (SB3) (Raffin et al. 2021). The experiments were conducted under both PPO and SAC settings, with baselines including original SAC/PPO, CAPS (Mysore et al. 2021), L2C2 (Kobayashi 2022), and Grad-CAPS (Lee et al. 2024) (hereafter GRAD). Isaac-Lab experiments were built on rl-games (Makoviichuk and Makoviychuk 2021) with domain randomization and observation noise for realism, and were compared against original PPO. We fixed preset random seeds for training and evaluation. Full details of the experiment setup are provided in Appendix B. Figure 2b shows our ASAP implementation. To implement the predictor, we added a prediction head to the actor's MLP in addition to the action head. The action head is trained with L_S (Eq. 14), while the prediction head is trained with L_P (Eq. 15).

Evaluation Metrics. Quantitative evaluation uses two metrics: **Cumulative Return (*re*)** and **Smoothness Score (*sm*)**. The cumulative return serves as a standard metric for assessing overall task performance. To quantify action oscillations, we adopted a smoothness measurement method based on the FFT frequency spectrum introduced in prior work (Mysore et al. 2021; Christmann et al. 2024) :

$$Sm \;=\; \frac{2}{n\, f_s} \sum_{i=1}^{n} M_i\, f_i, \tag{17}$$

where f_i and M_i denote the frequency and amplitude of the i-th component, and f_s denotes the sampling frequency. This metric computes a weighted average over the frequency components, reflecting both their magnitudes and frequencies. Higher values indicate the presence of more high-frequency components, while lower values indicate fewer high-frequency components and smoother motion.

Effectiveness on Transition-induced Similar State

We conducted an experiment to empirically validate the effectiveness of the proposed transition-induced similar state (Definition 3). Specifically, a SAC policy was fine-tuned using a predictor trained via supervised learning on a fixed replay buffer. The details are described in Appendix B.

Environment	SAC(Base)		SAC(Predictor)	
	re ↑	*sm* ↓	*re* ↑	*sm* ↓
LunarLander	280.1 (19.5)	0.296 (0.063)	278.8 (20.2)	0.227 (0.041)
Reacher	-3.61 (1.40)	0.051 (0.017)	-3.57 (1.42)	0.048 (0.015)
Hopper	3330 (412)	0.857 (0.061)	3478 (182)	0.712 (0.085)
Walker	4467 (435)	0.836 (0.131)	4419 (641)	0.715 (0.138)

Table 1: Cumulative return (*re*) and smoothness score (*sm*) of SAC before and after fine-tuning with predictor across Gymnasium. Standard deviations shown in parentheses.

As shown in Table 1, this fine-tuning procedure consistently reduced the smoothness score, with an average improvement of 15.1%. In particular, environments with more pronounced high-frequency oscillations, such as Hopper and Walker, exhibited reductions of up to 16.9% and 14.4%, respectively. Meanwhile, the cumulative return remained stable without significant changes. These results suggested that the transition-induced similar state effectively suppressed action oscillations while preserving task performance.

Evaluation on Gymnasium

Table 2 presents the results under the PPO setting. ASAP achieved the highest cumulative return in 3 out of 6 environments and recorded the best smoothness score in 5 environments. On average, the proposed method reduced *sm* by 43.3% compared to the PPO baseline, with particularly large improvements observed in Hopper (89.5%) and Walker (80.4%). In the Hopper environment, the slight decrease in *re* appears to result from excessive spatial smoothing in regions requiring rapid action changes. Nonetheless, our approach minimized the reduction in *re*, preserving reward stability. In contrast, CAPS and L2C2 reduced *sm* in Hopper by 83.5% and 21.3%, respectively, both of which are smaller than the reduction achieved by the proposed method, and they experienced a larger drop in *re*. This suggested that the heuristic similar state definitions employed by these methods failed to form neighborhood structures that adequately satisfy the optimal Lipschitz constraint.

ASAP also showed strong performance in the SAC setting (Table 3), achieving the highest *re* in 3 out of 6 environments and the best *sm* in 4 environments. On average, the proposed method reduced *sm* by 27.9% compared to the SAC baseline, while maintaining a similar level of *re*. Notably, our approach consistently matched or outperformed Grad-CAPS in both *re* and *sm*, suggesting that the Grad-based temporal term and our spatial regularization in ASAP complement each other effectively. These results demonstrated that the proposed method achieves a desirable balance between high reward and smoothness.

Applicability in Robotic Scenarios

As shown in Table 4, ASAP was successfully applied to robotics scenarios, improving or maintaining cumulative re-

Method	LunarLander		Pendulum		Reacher		Ant		Hopper		Walker	
	re ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓
PPO Base	172.8 (100.2)	0.309 (0.071)	**-169.4** (93.4)	0.383 (0.102)	-5.67 (2.12)	0.034 (0.015)	1434 (634)	1.497 (0.391)	**2902** (929)	1.709 (0.524)	2654 (1127)	1.764 (0.511)
CAPS	187.1 (104.0)	0.246 (0.054)	-238.2 (172.6)	0.323 (0.105)	-5.20 (1.83)	**0.028** (0.014)	2185 (837)	1.259 (0.253)	2362 (981)	0.281 (0.057)	2179 (1154)	0.565 (0.111)
L2C2	197.6 (95.5)	0.292 (0.069)	-203.0 (164.7)	0.387 (0.094)	**-4.67** (1.93)	0.039 (0.015)	1393 (749)	1.459 (0.425)	2345 (1048)	1.344 (0.631)	2014 (1102)	1.686 (0.604)
GRAD	188.0 (105.6)	0.298 (0.070)	-199.1 (161.4)	0.336 (0.119)	-5.48 (2.16)	0.031 (0.015)	2419 (804)	1.121 (0.172)	2737 (901)	0.193 (0.034)	1967 (1189)	**0.342** (0.082)
ASAP	**200.7** (105.6)	**0.221** (0.061)	-178.6 (100.8)	**0.318** (0.119)	-5.44 (1.99)	**0.028** (0.013)	**2574** (814)	**1.092** (0.215)	2691 (1004)	**0.179** (0.034)	**3128** (1045)	0.345 (0.094)

Table 2: Cumulative return (*re*) and smoothness score (*sm*) on Gymnasium benchmark under PPO setting. Higher *re* and lower *sm* are better. Bold indicates best, and underlined the second-best, per environment. Standard deviations shown in parentheses.

Method	LunarLander		Pendulum		Reacher		Ant		Hopper		Walker	
	re ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓
SAC Base	279.2 (18.1)	0.289 (0.052)	**-145.6** (69.6)	0.421 (0.141)	**-3.57** (1.41)	0.051 (0.016)	4239 (1552)	1.715 (0.475)	3349 (385)	0.856 (0.062)	4476 (392)	0.823 (0.128)
CAPS	251.0 (87.2)	0.259 (0.057)	-149.4 (68.9)	**0.304** (0.119)	**-3.57** (1.38)	0.046 (0.014)	4532 (1526)	1.739 (0.485)	3413 (33)	0.793 (0.085)	4320 (302)	0.815 (0.081)
L2C2	249.3 (90.4)	0.349 (0.122)	-146.2 (69.1)	0.383 (0.145)	**-3.57** (1.38)	0.051 (0.016)	4125 (1420)	1.811 (0.395)	**3455** (53)	0.893 (0.092)	4472 (558)	0.857 (0.183)
GRAD	250.9 (86.7)	0.248 (0.055)	**-145.6** (69.6)	0.345 (0.123)	-3.62 (1.47)	**0.043** (0.013)	3938 (2483)	1.423 (0.410)	3190 (628)	0.588 (0.115)	4339 (445)	0.612 (0.061)
ASAP	**280.9** (19.5)	**0.185** (0.038)	-145.7 (68.9)	0.338 (0.107)	-3.61 (1.43)	0.045 (0.015)	**4966** (1467)	**1.226** (0.267)	3448 (313)	**0.498** (0.077)	**4665** (494)	**0.578** (0.096)

Table 3: Cumulative return (*re*) and smoothness score (*sm*) on Gymnasium benchmark under SAC setting. Higher *re* and lower *sm* are better. Bold indicates best, and underlined the second-best, per environment. Standard deviations shown in parentheses.

turn while lowering smoothness across all four tasks. In particular, in the Franka reach task, it improved *re* and decreased the *sm* by 31.3%, yielding substantial improvements in both metrics. Figure 3 depicts the magnitude of the action difference for joint 1 between consecutive steps. The mean error between the end effector and the target also decreased by 30.3%, from 0.90 cm to 0.62 cm, suggesting that smoothing constraints are beneficial for tasks requiring high precision and sustained stability. In other tasks that prioritize dynamic interaction with the environment over fine-grained precision, such as Repose-Cube and Velocity, no significant reward improvement was achieved. Nevertheless, the consistent reward performance indicated that ASAP does not compromise the agent's responsiveness in these tasks. However, in environments such as Lift-Cube, we observed an increase in the variance of *re*, which we attribute to the oscillation suppression process constraining exploration and resulting in inadequate learning in a subset of random seeds.

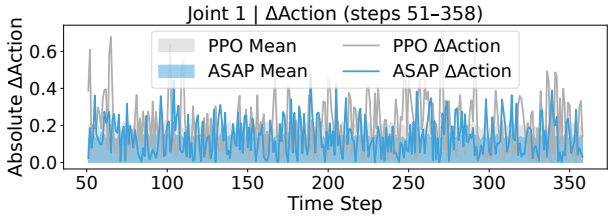


Figure 3: Action change (Δa) of Joint-1 in the Franka reach task for original PPO and ASAP. Solid line represents median Δa run; shaded area indicates mean across all seeds.

Compatibility with Architectural Method

We evaluated the compatibility of ASAP with architectural methods by integrating it with LipsNet under the SAC framework. For comparison, we adopted the original LipsNet (Song et al. 2023) and LipsNet+CAPS (Christmann et al. 2024) as baselines, and conducted experiments on the Hopper and Walker environments. As shown in Table 5, our method significantly improved the *re* and *sm* of LipsNet across all environments. In particular, the smoothness metric *sm* was reduced by 54.5% on Hopper and by 47.0% on Walker, indicating a substantial improvement in action stability. Compared to the LipsNet+CAPS variant, our approach achieved 41.4% and 27.1% lower *sm* on Hopper and Walker, respectively. These results demonstrated that ASAP can synergize with the architectural method.

Method	Reach-Franka		Lift-Cube-Franka		Repose-Cube-Allegro		Anymal-Velocity-Rough	
	re↑	*sm*↓	*re*↑	*sm*↓	*re*↑	*sm*↓	*re*↑	*sm*↓
PPO Base	0.380 (0.233)	0.959 (0.113)	**136.1** (15.1)	2.315 (0.873)	31.90 (13.09)	2.658 (0.612)	**16.69** (2.66)	3.502 (0.285)
ASAP	**0.525** (0.195)	**0.658** (0.065)	134.0 (32.2)	**0.926** (0.305)	**32.82** (15.91)	**2.363** (0.558)	16.09 (2.92)	**2.861** (0.306)

Table 4: Cumulative return (*re*) and smoothness score (*sm*) for PPO Base and PPO + ASAP on the Isaac-Lab environment. Higher *re* and lower *sm* are better. Bold indicates the best per environment. Standard deviations shown in parentheses.

Method	Hopper		Walker	
	re↑	*sm*↓	*re*↑	*sm*↓
LipsNet	2798 (665)	0.838 (0.143)	3942 (462)	0.915 (0.178)
Lips+CAPS	3096 (21)	0.650 (0.046)	3464 (1377)	0.665 (0.143)
Lips+ASAP	**3105** (45)	**0.381** (0.021)	**4475** (542)	**0.485** (0.103)

Table 5: Measurements of *re* and *sm* in SAC setting when combined with LipsNet.

L_S	L_T	Hopper		Walker	
		re↑	*sm*↓	*re*↑	*sm*↓
-	GRAD	**2963** (690)	0.241 (0.051)	2659 (1176)	0.541 (0.188)
CAPS	GRAD	2264 (1033)	0.201 (0.044)	2303 (1132)	0.467 (0.118)
L2C2	GRAD	2925 (705)	0.227 (0.035)	2500 (980)	0.519 (0.151)
ASAP	GRAD	2691 (1004)	**0.179** (0.034)	**3128** (1045)	**0.345** (0.094)

Table 6: Measurements of *re* and *sm* when the temporal term is fixed to GRAD and combined with various spatial terms.

Ablation Study

We verified that our spatial term represents the most effective strategy and evaluated its performance when combined with various temporal terms. The experiments were conducted using PPO as the baseline algorithm and were limited to the Walker and Hopper environments.

Comparison with Other Spatial Methods. We conducted experiments under four settings, fixing the temporal term to Grad-CAPS, as shown in Table 6. The CAPS spatial term reduced *sm* by 15.1% but caused a significant drop in *re*. The L2C2 term showed minimal effect, reducing *sm* by only 4.9% with a slight decrease in *re*. In contrast, our spatial term reduced *sm* by 30.9 %, achieving the largest oscillation reduction, while stably maintaining *re*. These results demonstrated that when the Grad-CAPS temporal term is combined with our proposed spatial term, it yields superior performance in reducing oscillations and preserving rewards compared to other methods.

L_S	L_T	Hopper		Walker	
		re↑	*sm*↓	*re*↑	*sm*↓
-	-	2902 (929)	1.709 (0.524)	2654 (1127)	1.764 (0.511)
-	CAPS	2962 (797)	0.371 (0.073)	2249 (1174)	0.872 (0.209)
ASAP	CAPS	**2973** (706)	0.235 (0.047)	2357 (1282)	0.389 (0.076)
-	GRAD	2963 (690)	0.241 (0.051)	2659 (1176)	0.541 (0.188)
ASAP	GRAD	2691 (1004)	**0.179** (0.034)	**3128** (1045)	**0.345** (0.094)

Table 7: Measurements of *re* and *sm* when the spatial term is fixed to ASAP and combined with various temporal terms.

Compatibility with Temporal Methods. We conducted experiments under five combination settings, as shown in Table 7, to evaluate whether our proposed spatial term is compatible with various temporal terms. When integrated with the CAPS temporal regularization, *sm* exhibited substantial improvements of 36.6% in Hopper and 55.3% in Walker, accompanied by consistent gains in *re* across both environments. Similarly, incorporating the Grad-CAPS temporal term led to *sm* improvements of 25.7% in Hopper and 36.2% in Walker, while *re* also improved, except for a marginal decline in Hopper. These results showed that when our spatial term is combined with a temporal term, it reduces oscillations without significant interference.

Conclusion and Limitation

Conclusion. In this paper, we introduced ASAP, a novel action-smoothing method for reinforcement learning. ASAP mitigates action oscillations by leveraging the transition function's Lipschitz continuity and penalizing second-order action differences. Through extensive experiments, we demonstrated that our method effectively maintains policy performance while improving action smoothness.

Limitation. While ASAP defines similar states based on observation noise to better reflect real transition dynamics, the presence of high noise levels may enlarge the resulting neighborhood, potentially diminishing the effectiveness of the local Lipschitz constraint and spatial regularization. This issue can be mitigated by adjusting the spatial loss weight λ_S.

Acknowledgements

This work was supported by the Technology Innovation Program(RS-2025-25453780, Development of a National Humanoid AI Robot Foundation Model for Multi-Task Applications) funded by the Ministry of Trade Industry & Energy(MOTIE, Korea), and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) under Grant RS-2022-00155911, Artificial Intelligence Convergence Innovation Human Resources Development (Kyung Hee University), and in part by Convergence security core talent training business support program under Grant IITP-2023-RS-2023-00266615).

References

Camacho, E. F.; and Alba, C. B. 2013. *Model Predictive Control*. Springer Science & Business Media.

Chen, C.; Tang, H.; Hao, J.; Liu, W.; and Meng, Z. 2021. Addressing action oscillations through learning policy inertia. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7020–7027.

Christmann, G.; Luo, Y.-S.; Mandala, H.; and Chen, W.-C. 2024. Benchmarking Smoothness and Reducing High-Frequency Oscillations in Continuous Control Policies. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 627–634. IEEE.

de la Presilla, R.; Wandel, S.; Stammler, M.; Grebe, M.; Poll, G.; and Glavatskih, S. 2023. Oscillating rolling element bearings: A review of tribotesting and analysis approaches. *Tribology International*, 188: 108805.

Doyle, J. 1996. Robust and optimal control. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 2, 1595–1598 vol.2.

Featherstone, R. 2008. *Rigid Body Dynamics Algorithms*. Springer.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Gogianu, F.; Berariu, T.; Rosca, M. C.; Clopath, C.; Busoniu, L.; and Pascanu, R. 2021. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, 3734–3744. PMLR.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. Pmlr.

Kobayashi, T. 2022. L2c2: Locally lipschitz continuous constraint towards stable and smooth reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4032–4039. IEEE.

Lee, I.; Cao, H.-G.; Dao, C.-T.; Chen, Y.-C.; and Wu, I.-C. 2024. Gradient-based Regularization for Action Smoothness in Robotic Control with Reinforcement Learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 603–610. IEEE.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39): 1–40.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Makoviichuk, D.; and Makoviychuk, V. 2021. rl-games: A High-performance Framework for Reinforcement Learning. https://github.com/Denys88/rl_games.

Mittal, M.; Yu, C.; Yu, Q.; Liu, J.; Rudin, N.; Hoeller, D.; Yuan, J. L.; Singh, R.; Guo, Y.; Mazhar, H.; Mandlekar, A.; Babich, B.; State, G.; Hutter, M.; and Garg, A. 2023. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robotics and Automation Letters*, 8(6): 3740–3747.

Mysore, S.; Mabsout, B.; Mancuso, R.; and Saenko, K. 2021. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1810–1816. IEEE.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268): 1–8.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Song, X.; Chen, L.; Liu, T.; Wang, W.; Wang, Y.; Qin, S.; Ma, Y.; Duan, J.; and Li, S. E. 2025. LipsNet++: Unifying Filter and Controller into a Policy Network. In *Forty-second International Conference on Machine Learning*.

Song, X.; Duan, J.; Wang, W.; Li, S. E.; Chen, C.; Cheng, B.; Zhang, B.; Wei, J.; and Wang, X. S. 2023. LipsNet: A smooth and robust neural network with adaptive Lipschitz constant for high accuracy optimal control. In *International Conference on Machine Learning*, 32253–32272. PMLR.

Sontag, E. D. 1998. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer, 2nd edition.

Spong, M. W.; Hutchinson, S.; and Vidyasagar, M. 2006. *Robot Modeling and Control*. Wiley.

Takase, R.; Yoshikawa, N.; Mariyama, T.; and Tsuchiya, T. 2022. Stability-certified reinforcement learning control via spectral normalization. *Machine Learning with Applications*, 10: 100409.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 23–30.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; De Cola, G.; Deleu, T.; Goulao, M.; Kallinteris, A.; Krimmel, M.; KG, A.; et al. 2024. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.

Wang, X.; Wang, S.; Liang, X.; Zhao, D.; Huang, J.; Xu, X.; Dai, B.; and Miao, Q. 2024. Deep Reinforcement Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4): 5064–5078.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8: 229–256.