2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Gradient-based Regularization for Action Smoothness in Robotic Control with Reinforcement Learning

I Lee*[1], Hoang-Giang Cao*[2], Cong-Tinh Dao[1], Yu-Cheng Chen[1], and I-Chen Wu†[1,3]

[1]*Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan*
[2]*Ming Chi University of Technology, Taiwan*
[3]*Research Center for IT Innovation, Academia Sinica, Taiwan*

***Abstract*— Deep Reinforcement Learning (DRL) has achieved remarkable success, ranging from complex computer games to real-world applications, showing the potential for intelligent agents capable of learning in dynamic environments. However, its application in real-world scenarios presents challenges, including the jerky problem, in which jerky trajectories not only compromise system safety but also increase power consumption and shorten the service life of robotic and autonomous systems. To address jerky actions, a method called conditioning for action policy smoothness (CAPS) was proposed by adding regularization terms to reduce the action changes. This paper further proposes a novel method, named Gradient-based CAPS (Grad-CAPS), that modifies CAPS by reducing the difference in the gradient of action and then uses displacement normalization to enable the agent to adapt to invariant action scales. Consequently, our method effectively reduces zigzagging action sequences while enhancing policy expressiveness and the adaptability of our method across diverse scenarios and environments. In the experiments, we integrated Grad-CAPS with different reinforcement learning algorithms and evaluated its performance on various robotic-related tasks in DeepMind Control Suite and OpenAI Gym environments. The results demonstrate that Grad-CAPS effectively improves performance while maintaining a comparable level of smoothness compared to CAPS and Vanilla agents.**

I. Introduction

In recent years, deep reinforcement learning (DRL) has achieved numerous milestones in many challenging computer games, ranging from traditional board games like Go, as demonstrated by AlphaGo [15], to complicated real-time strategy (RTS) video games such as Starcraft with AlphaStar [18] and Dota2 with OpenAI Five [13]. DRL has also been applied to various real-world control tasks, such as robotics and autonomous systems. This holds the promise of intelligent agents capable of learning to make decisions through interactions within complex and dynamic environments.

However, applying DRL to real-world applications presents some challenges, particularly when it results in a jerky trajectory ([12], [9], [16], [3]). During the training process, DRL primarily focuses on maximizing accumulated rewards, typically without enforcing constraints related to trajectory smoothness. Thus, lack of trajectory smoothness gives rise to critical issues, especially in autonomous systems such as robot manipulation or autonomous driving, where unpredictable behaviors are likely to be dangerous. Such unstable control not only compromises the safety of the system but also significantly increases power consumption, resulting in mechanical wear and tear, reducing energy efficiency, and, consequently, the service life of the robot or vehicle. Tackling the problem of jerky behaviors emphasizes the importance of stability and control issues in implementing DRL in real-world scenarios. This effort enhances system performance and ensures the safe, reliable, and sustainable operation of robotic and autonomous systems across various applications.

To address the jerky problem, a straightforward approach is to use reward engineering. Reward engineering is based on prior human knowledge about the tasks to design a specific reward function to penalize unsmooth trajectory ([8], [10], [4]). However, this approach is task-specific, limiting its generalizability to apply to other tasks.

Another approach is to use DRL with a hierarchical network structure. The objective is to optimize the overall episode reward while simultaneously mitigating control or action oscillations, as demonstrated in the works of Yu et al. (2021) [19] and Chen et al. (2021) [5]. Mysore *et al.* proposed conditioning for action policy smoothness (CAPS) for solving jerky actions in low-dimensional input by adding regularization terms [12]. Based on CAPS, Cao *et al.* proposed Image-based regularization for action smoothness (I-RAS) [3], focusing on high-dimensional input scenarios. Furthermore, based on Lipschitz constraints, recent contributions such as Locally Lipschitz Continuous Constraint (L2C2) [9] and LipsNet [16] aim to ensure the policy function slowly changes to obtain the smoothness with small value of K-Lipschitz constants.

This paper proposes a novel approach to improve policy smoothness in DRL, named Gradient-based Conditioning for Policy Action Smoothness (Grad-CAPS). Grad-CAPS has several advantages compared to prior research. Our method reduces the difference in the gradient of action, like the first-order derivative of the policy function, enhancing its ability to identify and smoothen zigzagging action sequences. We also introduce displacement normalization to effectively regularize the action sequences regardless of the action scale across various scenarios and environments. Our Grad-CAPS can serve as a new condition to other methods that based on Lipschitz constraints like CAPS, L2C2, or LipsNet to obtain smoothness behaviors. As a regularization technique, our

*Equal contribution.
†Correspondence.

method can be incorporated into existing DRL algorithms. Overall, our Grad-CAPS leads to smoother behaviors while maintaining comparable performance to other methods.

The main contributions of this paper are summarized as follows:

- We propose Grad-CAPS, a method with regularization on the first-order derivative of the policy function, which effectively reduces zigzagging trajectories.
- We introduce displacement normalization to enable our method to adapt to invariant action scales and generalize across diverse environments.
- In our experiments, we demonstrate that integrated Grad-CAPS with different reinforcement learning algorithms clearly outperforms CAPS and other methods without CAPS on various robotic-related tasks while preserving both the expressiveness and smoothness of the policy.

II. Backgrounds

A. Conditioning for Action Policy Smoothness (CAPS)

Mysore *et al.*[12] proposed a regularization technique to minimize the action change, resulting in a smoother trajectory. The method is constructed by two regularization terms: 1) temporal smoothness term, and 2) spatial smoothness term.

The temporal smoothness term L_{temp} is the difference in action taken on two consecutive states s_t and s_{t+1}. Minimizing the term L_{temp} is equivalently to reduce the difference between chronologically adjacent actions. The spatial smoothness term L_{spat} is the difference between two actions taken on state s and s', where s' is a similar state to s and is generated by sampling from a normal distribution Φ centered around state s. Cao *et al.* [3] used domain randomization to generate s' when dealing with high-dimensional input. CAPS minimizes these two regularization terms while maximizing the original objective of reinforcement learning algorithm J_π for a given policy π, formalized as follows.

Let ($\mathcal{A}$, $d_\mathcal{A}$) and ($\mathcal{S}$, $d_\mathcal{S}$) be two metric spaces, where both $\mathcal{A}$ and $\mathcal{S}$ are action and state spaces respectively. The distance metric $d_\mathcal{A}$ for $\mathcal{A}$ is based on the Euclidean distance, namely, for two actions a_1 and a_2,

$$d_\mathcal{A}(a_1, a_2) = ||a_1 - a_2||_2. \tag{1}$$

The distance metric $d_\mathcal{S}$ for $\mathcal{S}$ is based on the cardinal distance in an episode, namely, in a sequence of states, $\{s_1, s_2, ...\}$,

$$d_\mathcal{S}(s_t, s_{t+k}) = k. \tag{2}$$

Thus, two consecutive states $d_S(s_t, s_{t+1})$ is simply one.

An RL method decides an action for a given state s, based on a policy π. Namely, the action is taken by $a = \pi(s)$ in a deterministic manner, or by sampling on the distribution of a policy $a \sim \pi(\cdot|s)$ stochastically. The actions considered in the above regularization terms for CAPS are derived deterministically. Thus, $\pi(s)$ serves a mapping function from state s to action a. The objective function of CAPS is to maximize J_π^{CAPS} as follows:

$$J_\pi^{\text{CAPS}} = J_\pi - \lambda_t L_{temp} - \lambda_s L_{spat}, \tag{3}$$

$$L_{temp} = d_\mathcal{A}(a_t, a_{t+1}), \tag{4}$$

$$L_{spat} = d_\mathcal{A}(a_t, a'_t) \tag{5}$$

$$\text{where} \quad a'_t = \pi(s'_t) \quad \text{and} \quad s'_t \sim \Phi(s_t).$$

The regularization weights λ_t and λ_s are used to control the impact of L_{temp} and L_{spat}, respectively.

The experiments for CAPS reported the significant influence of temporal smoothness on action smoothness compared to spatial smoothness. So, this paper focuses exclusively on temporal smoothness and omits spatial smoothness.

B. Lipschitz Constraints

Some of the previous works [12], [3], [9], [16] also investigated to use the Lipschitz constraint to obtain the smoothness trajectory. Let us review Lipschitz constraints as follows.

Definition II.1. (Lipschitz Constraints) Let (X, d_X), (Y, d_Y) be two metric spaces, where d_X and d_Y denote the distance metrics on sets X and Y, respectively. A function f: $X \to Y$ is called *K-Lipschitz continuous* if there exists a real constant $K \geq 0$ such that, $\forall x_1, x_2 \in X$,

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2), \tag{6}$$

A smaller constant value for K indicates a smoother function f. While computing the exact values of Lipschitz constants is proven to be an NP-hard problem [14], many studies utilize various regularization techniques to approximate the optimal value of Lipschitz constraints K, thereby achieving smoother functions.

Temporal smoothness in CAPS can also be viewed from Lipschitz constraints, where (S, d_S) and (A, d_A) correspond to (X, d_X) and (Y, d_Y) respectively, and π serves as f. Suppose that temporal smoothness satisfies K-Lipschitz continuity. Thus,

$$L_{temp} = d_\mathcal{A}(a_t, a_{t+1}) \leq K d_S(s_t, s_{t+1}). \tag{7}$$

Since $d_\mathcal{A}(a_t, a_{t+1}) = ||a_t - a_{t+1}||_2$ and $d_S(s_t, s_{t+1}) = 1$ as above, the following holds: $||a_t - a_{t+1}||_2 \leq K$.

When CAPS minimizes the difference in action for smoothness, it also implicitly makes the Lipschitz constant K smaller in general. However, a small K causes the issue of *over-smoothing*, namely, leading to a loss of expressiveness, as reported in the work [9]. For this issue, this paper proposes a regularization method over the first-order derivative of the policy function that obtains both smoothness and expressiveness of the action trajectory.

III. Our Approach

A. Gradient-based Condition for Action Policy Smoothness (Grad-CAPS)

As discussed in subsection II-A above, CAPS made the policy excessively smooth by minimizing action changes. In terms of Lipschitz constraints, CAPS enforces the policy function π with a small K, therefore losing the capability to handle situations requiring the agent to change actions with

agility. For this issue, this paper proposes a new regularization approach called Gradient-based Condition for Action Smoothness (Grad-CAPS), which minimizes the differences of action changes, that is, the first-order derivative of action, instead of actions.

Before discussing our method, we first define the first-order Lipschitz constraints over f', the first-order derivative of the function f, as below:

Definition III.1. (First-order Lipschitz Constraints) Let (X, d_X), (Y, d_Y) be two metric spaces, as in Definition II.1. Let f be a function: $X \to Y$, and f' be a first-order derivative of f: $X \to \Delta Y$. Thus, $(\Delta Y, d_{\Delta Y})$ is a metric space, where $d_{\Delta Y}$ denotes the distance metric on ΔY. A function f is called first-order K-Lipschitz continuous and f' is K-Lipschitz continuous, if there is a constant $K \geq 0$ satisfying the following:

$$d_{\Delta Y}(f'(x_1), f'(x_2)) \leq K d_X(x_1, x_2). \tag{8}$$

As described above, we introduce a novel temporal smoothness approach by minimizing the change in the first-order derivative of action instead of the change in actions. As in subsection II-A where $\pi(s)$ is defined as a mapping function from $\mathcal{S}$ to $\mathcal{A}$, we define $\pi'(s)$ to be a mapping function from $\mathcal{S}$ to $\Delta\mathcal{A}$, the first-order derivative of action space $\mathcal{A}$, which are action changes between two consecutive time steps, namely,

$$\pi'(s_t) = a_t - a_{t-1}. \tag{9}$$

Let Δa_t denote $\pi'(s_t)$. Then, Δa_t forms a space $\Delta\mathcal{A}$, and ($\Delta\mathcal{A}$, $d_{\Delta\mathcal{A}}$) is a metric space, where $d_{\Delta\mathcal{A}}$ is the distance metric in the space $\Delta\mathcal{A}$ based on the Euclidean distance. Namely, for two action changes Δa_{t1} and Δa_{t2}:

$$d_{\Delta\mathcal{A}}(\Delta a_{t1}, \Delta a_{t2}) = ||\Delta a_{t1} - \Delta a_{t2}|| \tag{10}$$

Instead, temporal smoothness loss in Grad-CAPS is defined as:

$$\begin{aligned} L_{temp} &= d_{\Delta\mathcal{A}}(\Delta a_t, \Delta a_{t+1}) \\ &= ||(a_t - a_{t-1}) - (a_{t+1} - a_t)||_2 \end{aligned} \tag{11}$$

From the viewpoint of first-order Lipschitz constraint, suppose that Grad-CAPS temporal smoothness satisfies the first-order Lipschitz continuity. Thus,

$$L_{temp} = d_{\Delta\mathcal{A}}(\Delta a_t, \Delta a_{t+1}) \leq K d_{\mathcal{S}}(s_t, s_{t+1}) \tag{12}$$

Since $d_{\mathcal{S}}(s_t, s_{t+1})$ is 1 and $d_{\Delta\mathcal{A}}(\Delta a_t, \Delta a_{t+1}) = ||(a_t - a_{t-1}) - (a_{t+1} - a_t)||_2$ from above, the formula now becomes:

$$||(a_t - a_{t-1}) - (a_{t+1} - a_t)||_2 \leq K \tag{13}$$

In contrast to CAPS, which obtains smoothness by reducing the difference in action, our Grad-CAPS focuses on minimizing the difference in action change. Fig. 1 illustrates the distinction between CAPS and Grad-CAPS. Based on our definition, Grad-CAPS leads to improved regularization for distinguishing zigzagging sequences and larger effective action sequences compared to CAPS. Fig. 2 illustrates the advantages of Grad-CAPS in recognizing

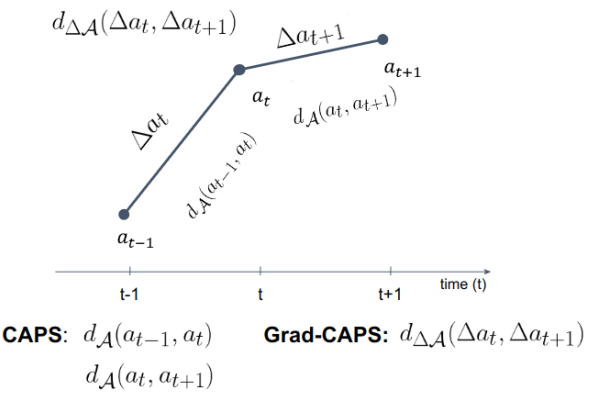


Fig. 1: Differences of temporal smoothness loss between CAPS and Grad-CAPS.

zigzag trajectories over CAPS. We can observe that CAPS fails to distinguish between a zigzagging sequence and a sequence of actions with stable changes, whereas our Grad-CAPS allows for substantially stable changes in action while penalizing zigzagging patterns. In short, CAPS defines a smooth trajectory as a slow change in action, while Grad-CAPS characterizes it as a stable change in action.

B. Displacement Normalization

Theoretically, our Grad-CAPS aims to reduce the difference in action changes as in (11).

Nonetheless, optimizing the gradient of action change that satisfies Equation 13 still suffers the issue of over-smoothing of the policy and a loss of expressiveness, like CAPS, for the following reason.

To minimize the term L_{temp}, the learning system tends to optimize the model in two ways. First, simply minimize the difference of consecutive Δa_t and Δa_{t+1}, as we expected in Table I.

Second, the system is distracted to minimize all Δa_t. For example, let L_{temp} in (11) satisfy the first-order K-Lipschitz constraints, as in Equation 12. For achieving this, one way is simply to enforce to satisfy $K/2$-Lipschitz constraints, instead. Since $K/2$-Lipschitz constraints are satisfied, we have the following conditions:

$$||a_t - a_{t-1}||_2 \leq K/2, \; ||a_{t+1} - a_t||_2 \leq K/2$$

Based on vector triangle inequality, obtain

$$\begin{aligned} L_{temp} &= ||(a_t - a_{t-1}) - (a_{t+1} - a_t)||_2 \\ &\leq ||a_t - a_{t-1}||_2 + ||a_{t+1} - a_t||_2 \\ &\leq K/2 + K/2 \leq K. \end{aligned} \tag{14}$$

That is, we can satisfy the first-order K-Lipschitz constraints by satisfying the $K/2$-Lipschitz constraints. Although this is less over-smoothing, the issue still exists.

To address this problem, we introduce displacement normalization to encourage the network to focus on optimizing

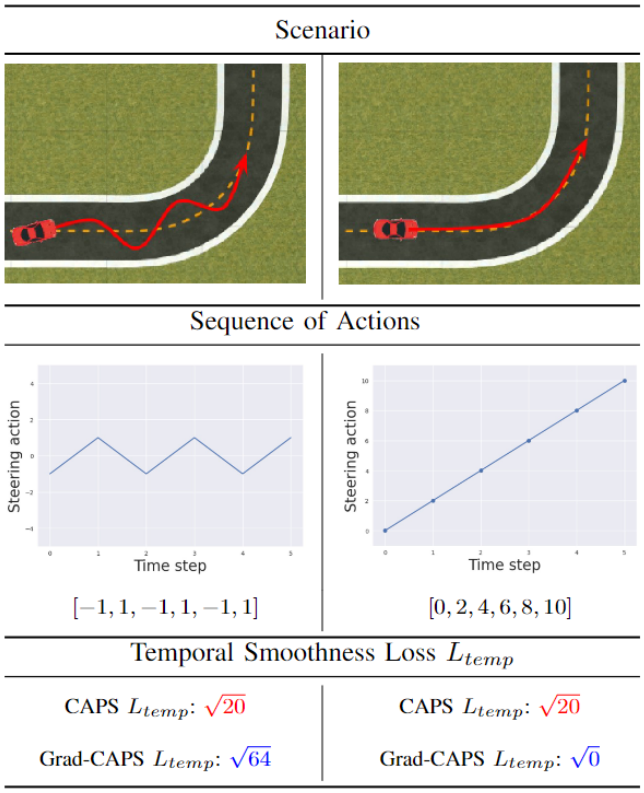


Fig. 2: Two cases: one with a zigzagging sequence of actions (left) and the other with a sequence with constant action changes (right). The upper part shows corresponding car racing scenarios, and the lower part shows corresponding losses for CAPS and Grad-CAPS. The cases show that CAPS fails to distinguish two sequences, while Grad-CAPS encourages stable action changes and penalizes zigzagging patterns.

the differences in the gradient of action instead of being distracted to optimize the differences in action.

We first define the total action displacement δ_t around time t as below:

$$\delta_t = \Delta a_{t+1} + \Delta a_t = (a_{t+1} - a_t) + (a_t - a_{t-1}) = a_{t+1} - a_{t-1} \tag{15}$$

Then, we divide the temporal smoothness loss in Equation 11 by action displacement δ_t:

$$L_{temp} = d_{\Delta\mathcal{A}}(\frac{\Delta a_t}{\delta_t}, \frac{\Delta a_{t+1}}{\delta_t}) = \left\| \frac{\Delta a_t - \Delta a_{t+1}}{\delta_t} \right\|_2 \tag{16}$$

Note that we add a small positive constant ϵ into denominator $D(= \delta_t)$ to prevent from dividing zero, as follows.

$$L_{temp} = \left\| \frac{\Delta a_t - \Delta a_{t+1}}{\delta_t + \epsilon} \right\|_2 \tag{17}$$

Besides, a small denominator D also lets the whole term go to a huge number in practice. To solve this problem, we also apply the $tanh$ function to $1/D$ to limit the value range to $[-1, 1]$ for stability. Thus our temporal regularization term in our Grad-CAPS becomes:

$$L_{temp} = \|\Delta a_t - \Delta a_{t+1}\|_2 \tanh(\left\| \frac{1}{(\delta_t + \epsilon)} \right\|_2) \tag{18}$$

Normalizing the temporal loss function over the displacement enables many advantageous features in our proposed method. First, Grad-CAPS regularizes the action sequence regardless of the action scale from different scenarios or environments. Second, it magnifies the loss of the zigzagging pattern. The low displacement in the zigzagging sequence magnifies the gradient loss by dividing the action displacement. Table I shows examples of the benefit of adding displacement normalization.

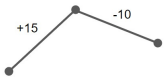
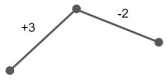
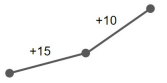
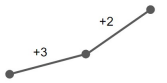
Action Sequence	Grad-CAPS Loss without normalization	Grad-CAPS Loss with normalization
	$\sqrt{(15-(-10))^2}$ $= \sqrt{625}$	$\sqrt{(\frac{15-(-10)}{5})^2}$ $= \sqrt{25}$
	$\sqrt{(3-(-2))^2}$ $= \sqrt{25}$	$\sqrt{(\frac{3-(-2)}{1})^2}$ $= \sqrt{25}$
	$\sqrt{(15-10)^2}$ $= \sqrt{25}$	$\sqrt{(\frac{15-10}{25})^2}$ $= \sqrt{\frac{1}{25}}$
	$\sqrt{(3-2)^2}$ $= \sqrt{1}$	$\sqrt{(\frac{3-2}{5})^2}$ $= \sqrt{\frac{1}{25}}$

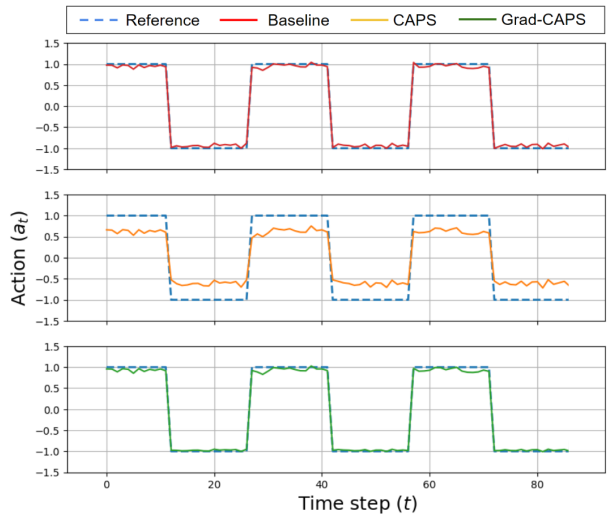
TABLE I: Examples of sequences of actions for comparison of the temporal smoothness losses with and without normalization on different action scales. Grad-CAPS with displacement normalization magnifies the loss of the zigzagging pattern, demonstrating its ability to handle different action scales and penalizing the zigzagging pattern.

IV. Experiment Results

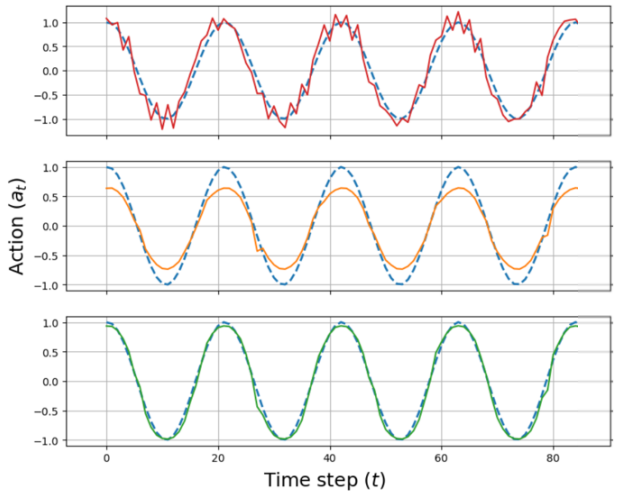
A. Experiment Setup

In the experiment, we compare the performance of the following agents: (a) Baseline agent, using Vanilla SAC algorithm. (b) CAPS agent.(c) Grad-CAPS agent (ours), as described in section III. These agents are tested across three tasks of varying difficulty levels: trajectory tracking in subsection IV-B, DeepMind Control Suite in subsection IV-C, and OpenAI Gym in subsection IV-D.

To evaluate the performance of the agent, we use the following metrics: (a) Average reward: the average accumulative reward achieved by an agent over 10 episodes, evaluated with the best policy. (b) Action fluctuation: Similar to that in [9], [16], we also evaluate the smoothness of the policy with L_2 norm of temporal action change, $\left\|a_t - a_{t-1}\right\|_2$, namely, the average of all action changes. A smaller value



(a) Square wave tracking experiment.



(b) Cosine wave tracking experiment.

Fig. 3: The referenced trajectories: (a) a square wave and (b) a cosine wave. The CAPS agent tends to over-smooth the action, leading to a loss of expressiveness in tracking the reference path. The Grad-CAPS agent performs better in following the reference path while maintaining smoothness.

of action fluctuation usually refers to smoother behavior. However, excessive smoothing does not always result in good performance.

B. Trajectory Tracking

In this section, we aim to assess the fundamental capability of the agent to generate precise action sequences or patterns, simulating the moving trajectory in general autonomous controlling tasks. The objective of this task is to follow the referenced trajectory accurately.

The trajectory tracking task is designed as follows. The observation at time t is the current position of the agent pos_t and the generated target point tar_t. From pos_t and tar_t, we calculate the distance from the agent's position to the generated target point denoted as $dist_t$. The state representation input to the agent is a $[2 \times 1]$ dimensional vector, defined as $s_t = [dist_t, pos_t]$. For a given state representation s_t, the agent is required to predict the next target point $\overline{tar}_{t+1}$. The reward is the negative of the difference between the predicted point $\overline{tar}_{t+1}$ and the next generated target point tar_{t+1}, i.e., $-||\overline{tar}_{t+1} - tar_{t+1}||$.

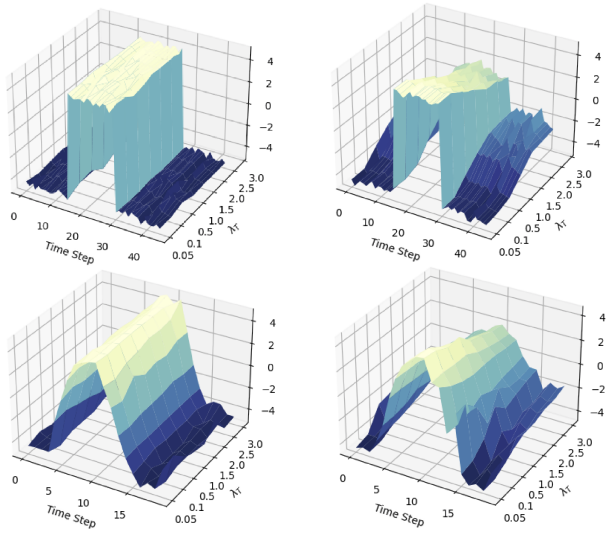
Method	Square Wave ↑	Cosine Wave ↑
Baseline	-14.3 ± 0.6	-16.6 ± 2.6
CAPS	-18.6 ± 0.7	-20.7 ± 2.5
Grad-CAPS	**-14.3 ± 0.5**	**-14.2 ± 1.8**

(a) The average reward

Method	Square Wave ↓	Cosine Wave ↓
Baseline	0.15 ± 0.4	0.29 ± 0.19
CAPS	**0.14 ± 0.2**†	**0.09 ± 0.03**†
Grad-CAPS	0.14 ± 0.4	0.17 ± 0.12

(b) Action fluctuation. † denotes the excessive smoothness that causes severe degradation in the agent's performance.

TABLE II: Results of trajectory tracking shown in Fig. 3.



(a) Grad-CAPS (b) CAPS

Fig. 4: Temporal loss weight λ_T ablation study on wave tracking experiments.

We design two referenced trajectories: square and cosine waves, which serve as toy problems. The square wave goal is to evaluate the expressiveness of the policy, while the cosine wave is to test the smoothness of the policy. Through these tests, we analyze the ability of agents to generate actions that align with specific patterns effectively.

Table II compares the results of different agents for the toy examples. From the table, Grad-CAPS clearly outperforms the other two in terms of average rewards. We observe that excessive regularization of action differences in the CAPS agent leads to a significant loss in policy expressiveness

Algorithms	Environments			
	Reacher	Cartpole	Walker	Ball-In-Cup
SAC - Vanilla	977.20 ± 10.30	856.45 ± 0.31	559.11 ± 15.00	976.49 ± 22.92
SAC - CAPS	980.56 ± 11.87	807.50 ± 6.39	498.37 ± 86.77	977.55 ± 15.27
SAC - Grad-CAPS	**982.20 ± 11.40**	**856.60 ± 0.39**	**561.15 ± 12.97**	**979.79 ± 16.62**
TD3 - Vanilla	936.94 ± 6.99	**860.27 ± 0.97**	186.11 ± 20.33	978.35 ± 12.26
TD3 - CAPS	909.34 ± 3.79	849.45 ± 0.45	555.90 ± 12.59	978.18 ± 13.02
TD3 - Grad-CAPS	**985.10 ± 9.30**	854.99 ± 0.32	**663.10 ± 14.70**	**980.04 ± 12.38**
DDPG - Vanilla	865.14 ± 8.37	877.74 ± 0.28	34.91 ± 14.27	977.78 ± 18.77
DDPG - CAPS	928.82 ± 9.64	857.94 ± 0.24	627.72 ± 18.10	974.94 ± 14.29
DDPG - Grad-CAPS	**928.26 ± 4.24**	**880.13 ± 0.36**	**670.05 ± 19.20**	**980.21 ± 13.16**
D4PG - Vanilla	970.10 ± 13.63	879.77 ± 0.65	752.97 ± 09.92	987.30 ± 13.12
D4PG - CAPS	982.00 ± 10.68	881.44 ± 0.73	748.59 ± 08.37	**987.60 ± 10.36**
D4PG - Grad-CAPS	**989.10 ± 14.85**	**882.49 ± 0.34**	**779.87 ± 05.46**	983.30 ± 15.49

TABLE III: The average reward on DMControl. The higher value of the average reward shows a better performance.

Algorithms	Environments			
	Reacher	Cartpole	Walker	Ball-In-Cup
SAC - Vanilla	24.10 ± 2.90	0.53 ± 0.03	20.70 ± 0.50	14.10 ± 1.71
SAC - CAPS	6.50 ± 0.40	0.48 ± 0.03	**2.74 ± 0.3**	1.30 ± 0.13
SAC - Grad-CAPS	**6.37 ± 0.28**	**0.41 ± 0.03**	4.90 ± 0.21	**0.60 ± 0.27**
TD3 - Vanilla	67.38 ± 19.32	**0.97 ± 0.46**	35.94 ± 20.23	46.18 ± 7.99
TD3 - CAPS	40.39 ± 19.03	1.61 ± 0.19	37.04 ± 0.73	**39.29 ± 1.00**
TD3 - Grad-CAPS	**66.20 ± 22.15**	2.14 ± 0.71	**33.04 ± 0.74**	43.59 ± 1.17
DDPG - Vanilla	64.94 ± 19.11	**1.34 ± 0.70**	33.82 ± 24.29	39.56 ± 0.96
DDPG - CAPS	95.79 ± 21.29	8.14 ± 0.38	**31.20 ± 1.02**	10.32 ± 7.71
DDPG - Grad-CAPS	**55.45 ± 22.62**	2.13 ± 1.34	35.81 ± 0.90	**2.04 ± 4.46**
D4PG - Vanilla	45.47 ± 23.09	1.56 ± 0.02	27.55 ± 9.62	18.40 ± 0.74
D4PG - CAPS	**32.17 ± 05.71**	1.92 ± 0.05	**23.40 ± 5.61**	18.93 ± 2.57
D4PG - Grad-CAPS	46.14 ± 31.26	**1.25 ± 0.01**	24.28 ± 3.74	**18.38 ± 1.31**

TABLE IV: Action fluctuation on DMControl. The values are in units of 10^{-2} their original values. The lower value of action fluctuation shows a better smoothness trajectory.

in both tests, though the agent has the least action fluctuation. The baseline agent performs well in the square wave tracking task; however, in the cosine wave tracking task, the lack of smoothness regularization results in a zigzagging trajectory when following the designed trajectory. In contrast, our regularization method outperforms other methods while effectively reducing zigzagging in actions and enabling the agent to stabilize action changes.

Fig. 3 visualizes the predicted trajectory following the reference path of different agents. From the observation of both square wave and cosine wave tracking tasks, Grad-CAPS performs well by following the reference path while maintaining a reasonably smooth trajectory.

Based on the above task for tracking square and cosine waves, we further investigate how the regularization weight λ_T affects the performance of Grad-CAPS. (Note that the regularization weight λ_S is ignored in this paper.) In our experiment, we let λ_T vary from 0.05 to 3.0. Fig. 4 shows the results for different λ_T by stacking the waves predicted by the agents in a 3D space. From this figure, the performance of our Grad-CAPS agent in wave tracking remains consistent regardless of different weights. Thus, the Grad-CAPS agent demonstrates reliable tracking capabilities. On the other hand, the CAPS agent exhibits distinct behaviors based on the weight settings. The CAPS agent performs better as weights become lower, but it produces more zigzagging patterns at the same time. On the other hand, higher weights result in a loss of expressiveness for the CAPS agent, indicating a decline in its ability to capture and represent the wave dynamics effectively. Overall, the above highlights the robustness of our Grad-CAPS agent in wave tracking tasks, while it is sensitive for the CAPS agent to choose a temporal weight for the performance. In the rest of our experiments, we set the $\lambda_T = 1$ as in [12].

C. DeepMind Control Suite

In this experiment, we evaluated the performance of our Grad-CAPS with a more complex environment, the DeepMind Control Suite [17] (referred to as DMControl in this paper). The DMControl provides a set of well-designed continuous control tasks involving interactions with diverse robotic systems, such as manipulating robotic arms

or maintaining balance in dynamic scenarios. Specifically, we selected four tasks from DMControl as follows:

For Cartpole, we choose Swingup, whose objective is to swing up and balance a pole attached to a cart. For Reacher, we choose Easy, whose objective is to control two joint motors of a robot arm to maneuver the endpoint toward a designated position without considering the complex dynamic influences present in other environments. For Ball-In-Cup, we choose Catch, whose objective is to control an actuated planar receptacle to swing and catch a ball attached to its bottom. For Walker, we choose Run, whose objective is to run forward as fast as possible, which requires the agent to make more rapid and aggressive action changes.

We compare four different reinforcement learning algorithms, including SAC [7], TD3 [6], DDPG [11], and D4PG [2], without smoothness terms (denoted as Vanilla) and with smoothness terms (denoted as CAPS and Grad-CAPS).

The results are presented in Table III and Table IV, showing the average reward and action fluctuation of different agents in the DMControl environment. The findings indicate that adding the smoothness term Grad-CAPS generally outperforms other configurations (Vanilla and CAPS) in terms of average rewards and smoothness values across different algorithms and tasks. We also observe that the SAC agent has a better smoothness value compared to TD3, DDPG, and D4PG. Grad-CAPS consistently demonstrates improved outcomes when combined with various reinforcement learning algorithms and tasks for controlling robot tasks, indicating its potential for enhancing the performance of the agent while maintaining comparable smoothness behaviors. This showcases a promising approach for practical application in robotic scenarios where the smooth action trajectory of agents is essential. Demonstration results are provided in the accompanying video of this paper.

D. OpenAI Gym

In this experiment, we select three tasks from OpenAI Gym [1]: Half-Cheetah, Humanoid, and Car-Racing. These tasks are used to evaluate the performances of agents on challenging high-dimensional robotics control tasks and car racing in complex dynamic environments. Note that in Car-Racing we only train and evaluate on a single map.

We implement and compare the SAC agents without and with smoothness terms. The average rewards and smoothness values of different methods in the OpenAI Gym environment are reported in Table V. We observe that CAPS significantly emphasizes smoothness, resulting in the lowest smoothness scores in Half-Cheetah and Humanoid. However, focusing on smoothness reduces its ability to adapt to tasks requiring action changes, reducing overall performance in task completion compared to other methods. In contrast, our Grad-CAPS agent still balances between performance and expressive action, therefore outperforming in all tasks while maintaining comparable smooth trajectories to CAPS. Additionally, in Car-Racing, where achieving smoother actions is crucial for enhancing performance, our Grad-CAPS agent distinctly demonstrates better trajectory smoothness as in Fig. 5.

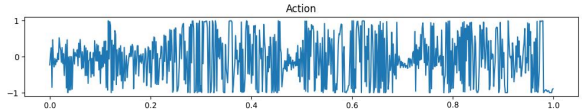
Method	Half-Cheetah	Humanoid	Car Racing
Vanilla	13060 ± 123	7530 ± 428	917
CAPS	10126 ± 15	7076 ± 212	932
Grad-CAPS	**13092 ± 97**	**8014 ± 390**	**942**

(a) The average reward on OpenAI Gym environments. The higher value of the average reward shows a better performance.

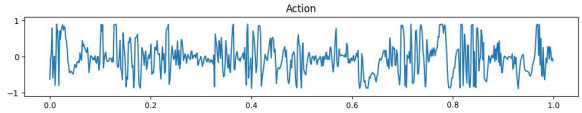
Method	Half-Cheetah	Humanoid	Car-Racing
Vanilla	6.76 ± 0.02	1.13 ± 0.02	0.35
CAPS	**4.11 ± 0.02**	**0.69 ± 0.02**	0.15
Grad-CAPS	6.08 ± 0.02	0.75 ± 0.01	**0.08**

(b) Action fluctuation on OpenAI Gym environments. The values are in units of 10^{-2}. The lower value of action fluctuation shows a better smooth trajectory.

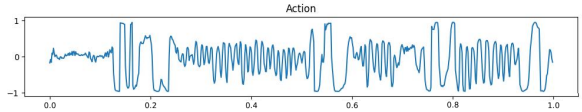
TABLE V: Results of SAC agent with different regularization terms on OpenAI Gym environments.



(a) SAC - Vanilla.



(b) SAC - CAPS.



(c) SAC - GradCAPS

Fig. 5: Steering actions during a track of different agents in Car Racing environment. Grad-CAPS clearly obtains smoother steering action compared to other methods.

V. Conclusion

In this paper, we addressed the critical issue of jerky trajectories in DRL and propose Grad-CAPS, a regularization method designed to reduce the problem of zigzagging actions. Grad-CAPS allows the agent to expressively change the action to adapt to the environment while maintaining a smooth trajectory, balancing enhanced performance and smooth behavior execution. We also introduced displacement normalization for adaptability to action scale, generalizing the use of our method across a wide range of applications. Our experiments demonstrate that Grad-CAPS effectively enhances the performance of agents while maintaining a comparable level of smoothness compared to other methods across various reinforcement learning algorithms. This improvement is beneficial for reinforcement learning techniques for autonomous systems where jerky actions can lead to inefficiencies or safety concerns.

REFERENCES

[1] MoJuCo - Gym Documentation. Accessed on 14/03/2024.

[2] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients, 2018.

[3] Hoang-Giang Cao, I Lee, Bo-Jiun Hsu, Zheng-Yi Lee, Yu-Wei Shih, Hsueh-Cheng Wang, and I-Chen Wu. Image-based regularization for action smoothness in autonomous miniature racing car with deep reinforcement learning, 2023.

[4] Ignacio Carlucho, Mariano De Paula, Sen Wang, Yvan Petillot, and Gerardo G. Acosta. Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning. *Robotics and Autonomous Systems*, 107:71–86, 2018.

[5] Chen Chen, Hongyao Tang, Jianye Hao, Wulong Liu, and Zhaopeng Meng. Addressing action oscillations through learning policy inertia. In *the Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

[6] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.

[7] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[8] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, oct 2017.

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

[10] William Koch. *Flight Controller Synthesis Via Deep Reinforcement Learning*. PhD thesis, Boston University, 2019.

[11] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[12] Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1810–1816, 2021.

[13] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

[14] Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation, 2019.

[15] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[16] Xujie Song, Jingliang Duan, Wenxuan Wang, Shengbo Eben Li, Chen Chen, Bo Cheng, Bo Zhang, Junqing Wei, and Xiaoming Simon Wang. LipsNet: A smooth and robust neural network with adaptive Lipschitz constant for high accuracy optimal control. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32253–32272. PMLR, 23–29 Jul 2023.

[17] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018.

[18] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

[19] Haonan Yu, Wei Xu, and Haichao Zhang. Taac: Temporally abstract actor-critic for continuous control. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29021–29033. Curran Associates, Inc., 2021.