Smooth Control of Soft Robots by Reinforcement Learning with Policy Regularization

Vladimir Sivtsov[1], Athanasios Papanikolaou[1], Ivan Markovic[1], Ivan Petrovic[1], Enrica Zereik[2], and Fabio Bonsignorio[1]

[1] University of Zagreb Faculty of Electrical Engineering and Computing, Unska ulica 3, Zagreb, 10000, Croatia, email: {name.lastname}@fer.unizg.hr
[2] Italian National Research Council, Institute of Marine Engineering (CNR-INM), 16149, Genoa, Italy , email: {enrica.zereik}@cnr.it

Abstract. Deep reinforcement learning has achieved promising results in various soft robotic control problems. However, its use in real-world scenarios is still limited. One of the reasons is the occurrence of jerky actions during training that affect the learned control policies and result in excessive oscillations in trajectories. In practice, using robots that implement such actions leads to excessive energy consumption, premature wear, and unsafe behavior. Existing methods have addressed this problem in various ways, such as reward engineering, different exploration techniques, and policy regularization. However, the challenge persists as obtaining high-performing policies that implement smooth control actions for complex, high-dimensional control problems of soft morphological structures is difficult. We propose a new regularization method for action policy in reinforcement learning for compliant robots. The proposed method maximizes the normalized-cross correlation between consecutive actions using a penalization term in the optimization loop. We evaluate the performance of our method on soft robotic tasks in simulation on the suite of EvoGym (OpenAI Gym-compatible benchmark for soft robots) and show that it achieves a better balance between smoothness of actions and high rewards, outperforming existing methods.

1 Introduction

The control of soft robots still poses significant challenges, [1]. In 2012, Calisti et al. [2] created the first flexible robot that could move and manipulate objects. Jones et al. [3] introduced a steady-state model of a continuous robot that is actuation-ignorant. Boyer et al. [4] provided an approximation of the distributed force and torque acting on the automata, but it makes no mention of the actuators that might produce them. Tendon-driven continuum robotics was approached using a continuous, geometrically correct method (Renda et al., [5]). In this context, Deep Reinforcement Learning (DRL) has emerged as a promising alternative. DRL is particularly useful in high-dimensional problems for sophisticated soft morphologies, where conventional control methods

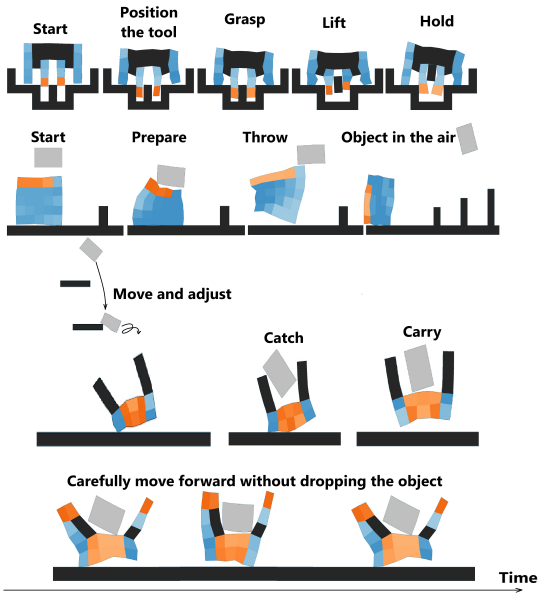


Fig. 1: A visual overview of the performance. From top to bottom: Lifter, Thrower, Catcher, Carrier

struggle to achieve the control goal. Despite significant improvements in DRL, there are still many open challenges [6]. One of the factors that prevents these methods from being widely used in practice is the lack of smoothness of actions, which leads to excessive energy consumption, premature wear and unsafe behaviour. Sampling inherently random control actions or adding Gaussian noise for stochastic policies has proven to be very effective in simulation [7]. However, applying it to real-world robotic systems often leads to: i) poor exploration of the environment; ii) the possibility of damaging the actuators in ordinary use; and iii) shaking trajectories [8].

Several approaches have been proposed to improve the balance between noisy exploration and control quality. Reward engineering [9] can help to encourage the learning agent to take smoother actions, but it requires a careful design of the reward function and is time-consuming. It needs extensive trial and error and is of limited generalization as the reward functions are constructed individually for different problems and in some cases lead the learning agent to unexpected (unintuitive) ways of maximizing it. Direct filtering of the output as it is used in the classical control is also not practical because deep learning control is fundamentally different [10]. Including state or action history in filtering breaks the Markov assumption, which is fundamental for RL algorithms. On the other hand, keeping the history of this data in a state matrix significantly increases the problem dimensionality, and thus the computational difficulty. There have been attempts to integrate filters into the learning system in different ways, for instance via learning with human intervention and a low-pass Butterworth fil-

ter [11], and by purposely designed neural networks that serve as a low-pass filter [12]. Some approaches focused on performing exploration in the parameter space rather than the action space, showing that it is a viable option [13]. State-Dependent Exploration (SDE) [14] was introduced to obtain a better balance between exploration in parameter and action space, by replacing exploration noise with a state-dependent exploration function, which resulted in smoother exploration. It was later extended to generalized State-Dependent Exploration (gSDE) [15] by introducing more general features and consistently resampling the noise. There have been attempts to regularize the actor policy with penalty terms which would help to achieve smooth policies. Conditioning for Action Policy Smoothness (CAPS) [10] uses regularization terms (with constant regularization gains) of the difference between actions taken at two consecutive steps (temporal smoothness) and of the difference between actions taken for similar states (spatial smoothness). L2c2 [16] employs distance-dependent regularization gains and regularizes both the policy and the value function approximations. Grad-CAPS [17] introduces regularization on the first-order derivative of the policy function. Overall, regularization-based approaches are often simpler than others, yet effective.

In this work, we propose a new regularization method for high-dimensional continuous control with reinforcement learning to improve the smoothness of learning for robotic agents with complex morphological structures. The proposed method enables to avoid some disadvantages other methods are prone to, such as being slow in exploration due to temporal soft constraints throughout the rollout or having spatially smooth policies, which might encourage failure in environments where due to certain boundary states present in the environment, the robot has to take significantly different actions for similar by spatial metrics observations. We evaluate the proposed method in the EvoGym simulation environment [18] - OpenAI Gym-compatible [19] benchmark for the evolution and control of soft robots. Fig. 1 illustrates the learned control policy for the Lifter robot in EvoGym. Overall, the proposed method is simple yet effective and in many cases outperforms other state-of-the-art methods in terms of rewards, while being smoother in terms of the L_1 norm of the time derivative of actions. The main contributions of this paper are summarized as follows:

- We propose a new action policy regularization method that helps to learn smoother policies for DRL for soft robot control
- We evaluate and analyze different approaches with respect to the trade-off between smoothness and obtained rewards.
- We demonstrate that the proposed method outperforms existing approaches in terms of smoothness, while achieving better or competitive performance measured by rewards.

2 Background

2.1 Smoothness and RL Algorithms

RL algorithms build on Markov Decision Processes (MDP) [20], where at each timestep the learning agent: i) obtains the state s by interacting with the en-

vironment, ii) chooses and takes the action a taking into account its policy π and iii) receives reward r. In the modern deep RL architectures, the action policy π is parametrized by weights θ of a neural network that predicts actions and is called an actor. The goal is to find a policy that maximizes the performance metric $J(\pi_\theta)$. For this, the state-of-the-art policy gradient optimization methods, including Proximal Policy Optimization (PPO) utilize the advantage function A_{π_θ} and divergence between policies at two different timesteps, which is usually Kullback–Leibler (KL)-divergence [21, 22].

To increase smoothness CAPS [10] adds two regularization terms to the optimization objective:

$$J_{\pi_\theta}^{\text{CAPS}} = J_{\pi_\theta} - \lambda_T L_T - \lambda_S L_s$$
$$L_T = D_T(\pi_\theta(s_t), \pi_\theta(s_{t+1}))$$
$$L_S = D_S(\pi_\theta(s_t), \pi_\theta(\bar{s_t})),$$

where $\bar{s}$ are states drawn from a distribution around s, D are distance measures (in practice Euclidian distance $D(a_1, a_2) = ||a_1 - a_2||_2$) and λ_* are constant scalar coefficients regulating the effect of each term.

Generalized State-Dependent Exploration (gSDE) [15] adds noise to deterministic action as a function of state and follows n-step-based exploration parameters sampling, which results in smoother actions while not significantly limiting the exploration as in SDE [23]. The smoothness is improved because for each period of n-steps the action for a given state is the same and does not fluctuate around the mean value.

2.2 Simulation Environment

EvoGym [18] is a simulation environment for the design and control of 2D voxel-based soft robots. It makes it possible to choose custom morphologies of soft robots and to control them. The morphologies are defined by the $x \times y$ matrix, being x and y the width and the height of the robot, respectively. The building blocks of soft robots are the following (with colors of blocks as in Figure 1):

- 0 - Empty (transparent)
- 1 - Solid non-deformable block (black)
- 2 - Soft deformable block (grey)
- 3 - Horizontal actuator which may expand or shrink along the x-axis (orange)
- 4 - Vertical actuator which may expand or shrink along the y-axis (blue)

The state of the robot s consists of different variables depending on the task e.g. position/velocities of point masses and orientation as described in the EvoGym benchmark [18]. A possible way to measure it in a real-world scenario is via a computer vision system that tracks these points in the robot (some parts can be painted in a different color for simplicity) and estimates the state s from the video stream in real-time. A practical approach to precise real-time local positioning, including orientation angle, is presented in [24].

In this work, we use human-designed intuitive morphologies of soft robots. Although it was shown before that human-designed morphologies in EvoGym are not always the best [18] as evolutionary algorithms may produce counter-intuitive but better-performing morphologies, as we focus on control, manually designed robot morphologies are more suitable for our purposes to have more interpretable results and a better understanding of how an optimal control policy should perform. The RL method that has demonstrated the best results in the EvoGym benchmark is PPO [21]; it has been commonly used in the benchmark in soft robot control and evolution research [25–27].

3 Proposed Method

This section provides details on the proposed method to improve the smoothness of the control while learning high-performing policies in terms of rewards. The regularization term is formulated as an episode-rollout level measure of coherence between consecutive actions. It encourages the action policy to implement temporally correlated actions.

3.1 Policy Regularization with Normalized Cross-Correlation

We propose to regularize the actor policy with the normalized-cross correlation $F(\pi_\theta(s_t), \pi_\theta(s_{t+1}))$ between consecutive actions predicted by the policy:

$$F(\pi_\theta(s_t), \pi_\theta(s_{t+1})) = -\frac{(\pi_\theta(s_t) - \mu_{\pi_\theta(s_t)})(\pi_\theta(s_{t+1}) - \mu_{\pi_\theta(s_{t+1})})}{\sigma_{\pi_\theta(s_t)}\sigma_{\pi_\theta(s_{t+1})} + \varepsilon}, \tag{1}$$

$$\mu_{\pi_\theta(s_t)} = \frac{1}{b}\sum_{0}^{b}\pi_\theta^{(b)}(s_t) \text{ and } \sigma^2_{\pi_\theta(s_t)} = \frac{1}{b-1}\sum_{0}^{b}(\pi_\theta^{(b)}(s_t) - \mu_{\pi_\theta(s_t)})^2,$$

where b denotes the number of dimensions in the control signal vector.

The value of the normalized cross-correlation is taken with the minus sign in order to allow adding the term to the minimization problem with a positive hyperparameter α i.e. to have the optimized loss in the form of a weighted sum of policy loss, entropy loss, value loss and the normalized cross-correlation loss. The ε is a small number used to prevent numerical instabilities due to division by 0 in cases when the action for all actuators is the same and thus $\sigma_{\pi_\theta(s)} = 0$. In practice, during training, F is not expected to be 0, especially for high-dimensional continuous problems, as the sampling strategies for exploration imply differences in temporal actions to enable learning.

The normalized cross-correlation is maximized during training i.e. the performance metric $J(\pi_\theta)$ becomes:

$$J^{\text{Ours}}_{\pi_\theta} = J_{\pi_\theta} - \alpha F(\pi_\theta(s_t), \pi_\theta(s_{t+1})). \tag{2}$$

The performance metric J_{π_θ} depends on the reinforcement learning algorithm. The generality of the proposed method allows to use it, in principle, for any reinforcement learning algorithm, however, in this work we focus on PPO as it overperforms other methods in the soft robots benchmark. PPO maximizes the clipped surrogate objective function:

$$J_{\pi_\theta} = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}.$$

The ratio $r_t(\theta)$ is commonly used in trust-region methods as it helps to balance between exploring new policies while not deviating too much from the current policy. It is multiplied by the advantage function $\hat{A}_t$. The clipping mechanism governed by ϵ attempts to prevent the ratio from moving out of the interval defined in clipping. The minimum between the clipped and unclipped objectives represents the pessimistic bound which is used as the final objective. Together with the proposed regularization, the surrogate objective function becomes:

$$J^{\text{Ours}}_{\pi_\theta} = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] + \\ + \alpha \frac{(\pi_\theta(s_t) - \mu_{\pi_\theta(s_t)})(\pi_\theta(s_{t+1}) - \mu_{\pi_\theta(s_{t+1})})}{\sigma_{\pi_\theta(s_t)}\sigma_{\pi_\theta(s_{t+1})} + \varepsilon}.$$

The hyperparameter α controls the strength of the regularization effect. Instead of directly minimizing smoothness-related metrics such as the temporal difference between actions or the spatial regularization for similar states, we make the hypothesis that consecutive actions do not necessarily have to be very similar, but they should be cross-correlated. The intuition behind it is that this would help to improve the coordination of the body of the robot and thus lead to more effective actions, while naturally decreasing temporal differences when needed. An improvement in the performance that supports this hypothesis is shown in Section 4, which discusses the results. The mean value of the actions $\mu_{\pi_\theta(\cdot)}$ is calculated across all actuators of the robot at a given timestep and subtracting it from the policy prediction produces zero-normalized values.

The regularization of temporal or spatial actions proposed in the literature is effective in terms of making actions smooth, but it also has disadvantages. The minimum of the objective is achieved when the actions are the same in time or the same for similar states of the robot. Strong regularization on the temporal term causes the robot to be slow overall and leads to a decrease in exploration and eventually lower rewards when using the learned policies. At the same time, strong regularization of the spatial regularization term would lead to misleading penalization in environments that have certain boundary states for which the actions for similar states should be completely different. Such a case is also present in our experimental suite, in particular, in the Lifter-v0 environment. The robot takes the object out of the hole of the size 3×2 by following the main

stages shown in Fig. 1. Throughout the performance and at the last stage of holding, the tool still partially resides inside the hole as it helps to hold the two 'arms' of the tool together i.e. preventing them from pulling apart. Arms would be pulled apart if the robot would stretch to its maximum height, which would provoke the fall of the object back into the hole and a significant negative reward. Thus, the actions should be significantly different for similar states where 1 - the tool is at the border of the hole, but is still inside of it and 2 - the tool is outside at the border of the hole. One of the advantages of the proposed method is that it does not penalize such actions at the borderline.

3.2 Measuring smoothness

Approaches to improve the smoothness of the learned control policies in most cases directly contribute to mitigating unnecessary oscillations, which result in desired changes in the frequency spectrum; however, the oscillations measured by frequency spectrum metrics may not always need to be small throughout the whole episode for some problems: an intuitive example of when a good control policy would have high oscillations leading to worse smoothness performance in terms of frequency spectrum analysis, but being good in terms of temporal difference analysis is the problem of removing water from the robot body surface. A few quick motions with the robot body, similar to how a dog shakes off the water after swimming, and then standing still (not moving) would allow a robot to achieve the goal while having a quick action with high oscillations, which is more effective in terms of achieving the goal. Taking this into account, we choose a norm of temporal difference in actions as a metric for measuring smoothness. The approach of measuring smoothness with similar temporal-based metrics has already found some use in the smooth reinforcement learning literature [16, 28, 17]. Such smoothness metric is also related to energy efficiency, as high smoothness mitigates the unnecessary oscillations and leads the robot to spend less energy for control, overall. If the actions are stationary, they will not cause any robot movement, and thus the actions are fully smooth.

In order to measure the smoothness, we leverage the following total variation for discrete signals metric:

$$S^{-1} = \sum_{i=1}^{n-1} \sum_{m=1}^{k} |a_{m_{i+1}} - a_{m_i}|, \tag{3}$$

where k is the number of actuators in the robot, n is the number of steps in an episode, and S^{-1} is stated in the inverse form so that higher values of S correspond to smoother trajectories, i.e. S^{-1} is 'unsmoothness'.

The maximum smoothness is achieved when there is no movement, i.e. $S^{-1} = 0$. On the other hand, the minimum is when the actions are switching between the boundaries of minimum and maximum values. As shown in Section 2.2, in EvoGym simulation $\forall a \in A \implies a \in [0.6, 1.6]$. In this case, the worst possible smoothness according to (3) is $k(n-1)$.

Table 1: Comparing performance on EvoGym soft robots control benchmark

Environment	Algorithm	Train		Evaluation			
		mean reward	S^{-1} (unsmoothness)	mean reward	S^{-1} (unsmoothness)	max reward	$S^{-1}_{\text{max reward}}$
Lifter-v0	gSDE	-0.70 ± 0.08	3810 ± 346	-0.58 ± 0.34	355 ± 93	0.00	248
	Vanilla PPO	0.84 ± 0.58	1101 ± 53	1.61 ± 0.77	826 ± 258	2.43	1151
	CAPS	0.67 ± 0.72	978 ± 65	1.21 ± 1.12	676 ± 291	2.39	894
	Ours	**1.07 ± 1.03**	**774 ± 144**	**1.42 ± 1.32**	**438 ± 226**	**2.93**	**540**
Carrier-v0	gSDE	0.11 ± 0.23	3028 ± 543	0.48 ± 0.79	345 ± 208	2.23	512
	Vanilla PPO	6.49 ± 0.71	1211 ± 67	6.76 ± 0.90	496 ± 70	8.30	560
	CAPS	6.12 ± 0.60	1183 ± 88	6.56 ± 1.28	493 ± 123	8.03	667
	Ours	**6.19 ± 0.63**	**970 ± 77**	**6.25 ± 1.78**	**362 ± 72**	**8.33**	**346**
Catcher-v0	gSDE	-4.10 ± 0.54	2415 ± 389	-4.30 ± 1.96	178 ± 101	-0.16	302
	Vanilla PPO	-3.14 ± 0.46	715 ± 56	-3.32 ± 1.51	249 ± 69	-0.39	286
	CAPS	-3.64 ± 0.43	675 ± 45	-3.80 ± 1.3	193 ± 47	-0.77	201
	Ours	**-3.80 ± 0.55**	**419 ± 35**	**-4.06 ± 1.74**	**140 ± 63**	**0.28**	**126**
Thrower-v0	gSDE	1.28 ± 0.25	8348± 837	1.47 ± 0.35	166 ± 31	1.69	183
	Vanilla PPO	1.00 ± 0.64	1668 ± 166	1.09 ± 0.85	224 ± 51	3.04	175
	CAPS	1.13 ± 0.38	1764 ± 151	1.31 ± 0.41	223 ± 30	1.77	223
	Ours	**0.87 ± 0.47**	**942 ± 163**	**0.95 ± 0.54**	**110 ± 45**	**1.75**	**118**

4 Experimental Results

We compared the performance of the proposed method with state-of-the-art methods for conventional non-smooth (vanilla) and smooth (gSDE, CAPS) control based on PPO, which have shown great success in the EvoGym benchmark as discussed in Section 2.2; the complete results are reported in Table 1. The algorithms for comparison are vanilla (basic) PPO implementation, PPO that uses generalized State-Dependent Exploration (gSDE), and PPO with CAPS regularization. As a baseline, we used the set of reliable implementations of reinforcement learning algorithms Stable Baselines 3 [29]. It includes the implementations of the vanilla PPO and gSDE. For consistency and reproducibility, all models were trained with the same base hyperparameters for PPO for 1 million total environment steps. Apart from the basic PPO hyperparameters, CAPS requires two additional hyperparameters that control the strength of the regularization, while the proposed method requires just one additional hyperparameter.

In particular, CAPS has tunable hyperparameters λ_*, which control the effect of regularization terms for temporal (L_T) and spatial (L_S) smoothness individually. High values for the regularization coefficients led to policies with low rewards that were not able to achieve success in benchmark tasks. On the other hand, low values prevented the improvement of the policy in terms of smoothness. At the beginning of training both L_T and L_S are close to 0; hence, in practice, for $\lambda_* < 1$ the regularization starts to take place actively after a number of timesteps, typically close to the middle of the training process. We follow a similar procedure as in [10], running several unregularized and regularized agents with different random states and regularization values, in order to choose hyperparameters that provide the best tradeoff between smoothness and performance on rewards, and found $\lambda_T = 0.015$ and $\lambda_s = 0.9$ being able to provide improvement in smoothness but still allowing the model to achieve high rewards and to succeed in benchmark tasks. Varying the strength of regularization of one of the parameters while keeping the other constant results also in the change of the other term i.e. the terms are indirectly related. Changing the parameter $\lambda_T = 0.015$ (which regulates the strength of L_T) to $\lambda_T = 0.15$, while keeping λ_S constant, results in the decrease of L_S from 0.0025 ± 0.0004

to 0.0015 ± 0.0003. Stronger temporal regularization implies smoother actions which indeed are confirmed by the empirical results: the metric S^{-1} decreased from 676 ± 291 to 298 ± 89 in evaluation, however, the maximum achieved reward across all agents with different random initialization also dropped from 2.39 to 1.58 i.e. the smoothness is improved at a cost of a significant decrease in the performance rewards-wise. The aforementioned changes in performance are also observed in a similar way during training. The strategy of selection of hyperparameters that worked best consisted of performing a run without CAPS and looking at the balance of terms in the loss function, in particular the policy loss, in the middle of training and then choosing λ that would make CAPS regularization terms approximately equal. Making the regularization terms approximately equal to the policy loss from the beginning would result in very smooth actions but poor performance in terms of rewards. As for gSDE, the default configuration provided by Stable Baselines 3 was used i.e. noise sampling at the beginning of rollout.

The proposed regularization method requires just one hyperparameter (i.e. α from (2)) that governs how strong the regularization is. We found that the effective yet intuitive way of choosing it is by balancing it with the policy loss. The proposed regularization term can be calculated at the first training step and it is easy to calculate what α should be by choosing it in a way to make αF approximately equal to the policy loss in the optimization loop. Running PPO to fill the rollout buffer size, i.e. until the first iteration of optimization, allows determining the policy loss value at the beginning of the training. For instance, in the case of Lifter-v0, the policy loss was determined in this way for a given random state and it appeared to be equal to -0.0169. The regularization term appeared to be $F = -24.4914$. Thus the hyperparameter α was calculated as $\alpha = \frac{-0.0169}{-24.4914} \approx 0.0007$ in a way to make the proposed regularization term F approximately equal to the policy loss. There is no requirement to perform several runs with different random steps to determine the hyperparameter α with the enhanced precision. We found that the policy loss typically has a low standard deviation for runs with different random states; for instance, for the Lifter-v0 the policy loss with standard deviation is -0.018 ± 0.002, yet with the chosen value of $\alpha = 0.0007$ based on the policy loss -0.0169, the regularization method performed competitively across all environments. Parameter ε is used to prevent numerical instabilities and can be any small number. We chose it to be 10^{-8}, however, in practice, the actions provided by the policy were always different in time, resulting in non-zero σ_{π_θ} in (1), and thus instabilities would not have occurred even if ε had been 0.

We evaluate the performance of the proposed method in simulation on challenging problems of high-dimensional soft robots control. It was tested on soft robots that have 10-25 actuators of different types as shown in Section 2.2. We report results with standard deviations for 10 different random initializations. The proposed method consistently achieves higher or competitive rewards while implementing smoother actions, both during the exploration and during evaluation. The performance on the Lifter-v0 task is showcased in Fig. 2. The solid

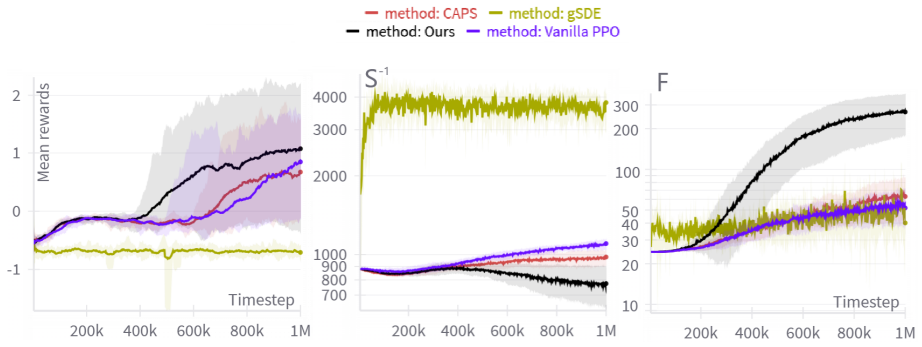


Fig. 2: Mean episode rewards, unsmoothness S^{-1} and normalized-cross correlation F during training of Lifter-v0 robot. The proposed method achieves higher rewards overall and does it faster with the smoothest actions.

line corresponds to the mean value, while the area around it is the standard deviation. The plot on the left shows the mean rewards, while the middle plot demonstrates the values of the unsmoothness metric S^{-1} according to (3). While gSDE generally provides consistent exploration within parts of the episode because of resampling of the exploration noise every n steps, it does not succeed in the Lifter task, leading to large action changes according to the chosen metric S^{-1} in an attempt to discover promising areas of state-action-reward space. CAPS executes actions with slightly lower values of S^{-1} than in vanilla PPO. The proposed method executes the actions with the lowest values according to the metric S^{-1}, i.e., resulting in the smoothest trajectory. The value of the proposed regularization term (F) between different algorithms is shown at the right.

Overall, with the proposed method, the learning agent finds the actions that lead to the solution of the problem faster than other methods while being smoother. CAPS regularization has a slightly worse performance in comparison with vanilla PPO, but it uses smoother actions for exploration. In this task, gSDE fails to produce a good learning policy while implementing energy-intensive actions. The proposed method has the highest values of the F function as it directly maximizes it. CAPS does not directly maximize it, but still slightly surpasses other methods. The complete results for both training and evaluation are shown in Table 1.

5 Conclusions

In this paper, we proposed a new regularization method for actor policy that is easy to include in the optimization and that does not impact the value function. It is based on maximizing normalized cross-correlation between two consecutive actions. The method has just one tunable hyperparameter α which governs the strength of regularization. An appropriate value of α can be found by looking at the values of the policy loss after filling the rollout buffer data for the first

policy optimization step and choosing α to make the product with the proposed regularization term loss αF approximately equal to the policy loss.

The proposed method achieves high performance in terms of reward while implementing smoother actions in comparison with other state-of-the-art algorithms. Future work will explore methods of smooth and energy-efficient learning and control via advanced regularization methods, information geometry and physics-informed methods.

Acknowledgment

This work was supported by the H2020 project AIFORS under Grant Agreement No 952275.

References

1. D. Rus and M. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467—-475, 2015.
2. M. Calisti, A. Arienti, F. Renda, G. Levy, B. Hochner, B. Mazzolai, P. Dario, and C. Laschi, "Design and development of a soft robot with crawling and grasping capabilities," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4950–4955, 2012.
3. B. A. Jones, R. L. Gray, and K. Turlapati, "Three dimensional statics for continuum robotics," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2659–2664, 2009.
4. F. Boyer, M. Porez, and W. Khalil, "Macro-continuous computed torque algorithm for a three-dimensional eel-like robot," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 763–775, 2006.
5. F. Renda, M. Cianchetti, M. Giorelli, A. Arienti, and C. Laschi, "A 3d steady-state model of a tendon-driven continuum soft manipulator inspired by the octopus arm," *Bioinspiration; Biomimetics*, vol. 7, p. 025006, May 2012.
6. G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, p. 2419–2468, apr 2021.
7. Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*, pp. 1329–1338, PMLR, 2016.
8. J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Advances in neural information processing systems*, vol. 21, 2008.
9. S. Ibrahim, M. Mostafa, A. Jnadi, and P. Osinenko, "Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications," *arXiv preprint arXiv:2408.10215*, 2024.
10. S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko, "Regularizing action policies for smooth control with reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1810–1816, 2021.
11. S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," *arXiv preprint arXiv:2002.08550*, 2020.
12. W. Wang, J. Duan, X. Song, L. Xiao, L. Chen, Y. Wang, B. Cheng, and S. E. Li, "Smooth filtering neural network for reinforcement learning," *IEEE Transactions on Intelligent Vehicles*, pp. 1–10, 2024.

13. M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," *arXiv preprint arXiv:1706.01905*, 2017.
14. T. Rückstieß, M. Felder, and J. Schmidhuber, "State-dependent exploration for policy gradient methods," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II 19*, pp. 234–249, Springer, 2008.
15. A. Raffin, J. Kober, and F. Stulp, "Smooth exploration for robotic reinforcement learning," in *Conference on robot learning*, pp. 1634–1644, PMLR, 2022.
16. T. Kobayashi, "L2c2: Locally lipschitz continuous constraint towards stable and smooth reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4032–4039, IEEE, 2022.
17. I. Lee, H.-G. Cao, C.-T. Dao, Y.-C. Chen, I. Wu, *et al.*, "Gradient-based regularization for action smoothness in robotic control with reinforcement learning," *arXiv preprint arXiv:2407.04315*, 2024.
18. J. Bhatia, H. Jackson, Y. Tian, J. Xu, and W. Matusik, "Evolution gym: A large-scale benchmark for evolving soft robots," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2201–2214.
19. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.
20. E. A. Feinberg and A. Shwartz, *Handbook of Markov decision processes: methods and applications*, vol. 40. Springer Science & Business Media, 2012.
21. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.
22. J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015.
23. H. van Hoof, D. Tanneberg, and J. Peters, "Generalized exploration in policy search," *Machine Learning*, vol. 106, no. 9, pp. 1705–1724, 2017.
24. V. Sivtsov, A. Ramos, S. Shavetov, and A. Zhivitskii, "Surface vessel: Local positioning system based on computer vision," in *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, vol. 1, pp. 1484–1488, IEEE, 2022.
25. H. Dong, J. Zhang, and C. Zhang, "Leveraging hyperbolic embeddings for coarse-to-fine robot design," *arXiv preprint arXiv:2311.00462*, 2023.
26. Y. Wang, S. Wu, T. Zhang, Y. Chang, H. Fu, Q. FU, and X. Wang, "Preco: Enhancing generalization in co-design of modular soft robots via brain-body pre-training," pp. 478–498.
27. Y. Wang, S. Wu, H. Fu, Q. Fu, T. Zhang, Y. Chang, and X. Wang, "Curriculum-based co-design of morphology and control of voxel-based soft robots," in *The Eleventh International Conference on Learning Representations*, 2023.
28. X. Song, J. Duan, W. Wang, S. E. Li, C. Chen, B. Cheng, B. Zhang, J. Wei, and X. S. Wang, "Lipsnet: a smooth and robust neural network with adaptive lipschitz constant for high accuracy optimal control," in *International Conference on Machine Learning*, pp. 32253–32272, PMLR, 2023.
29. A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.