Improving Smooth Control in Model-free Reinforcement Learning

Juan Jose Garau Luis
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
garau@mit.edu

1 Introduction

Model-free Reinforcement Learning (RL) [1] is becoming increasingly popular in many real-world fields, such as robotics, communications, or biotechnology. In addition to the general RL advantages such as fast decision-making and not needing any supervision, model-free RL has proven to perform better than model-based RL in most cases [2]. This paradigm is hence becoming a potential component of future real-world autonomous systems, where learning a model might not always be possible. However, the progress in real-world settings is slow; this is due to different challenging features that are usually not present in common RL simulation frameworks [3, 4]. Non-stationarity and stochasticity are two of these issues, which in the context of real-world control applications such as robotics can lead to non-smooth or jerky policies in the time-space.

Apart from visually-undesired performances, jerkiness in the policy outcome can also impact certain subsytems of the robot, such as sensors and actuators. Different methods such as reward shaping or action filtering have been proposed to address the issue of jerky policies. Despite their promising results, it remains unclear how different approaches compare to each other and, more importantly, trade with respect to the environment's reward. In real-world deployments, robot operators might prefer to implement multiple methods and choose depending on the context in real time instead of relying on a single approach.

This project tries to close this research gap by implementing several strategies to reduce jerk proposed in the literature and doing an extensive comparison both in terms of smoothness and reward. To that end, different robotics-inspired environments are used as benchmarks, and jerk analyses of the results are provided. The findings suggest that the strategy that works best strongly depends on the specific environment considered and reward shaping methods, which can effectively encode the need to reduce jerk, are difficult to tune in a generalized manner. Nevertheless, we show achieving lower jerk while preserving performance is possible and outline different areas of future work.

1.1 Related Work

To try to improve learning outcomes in non-stationary or highly-stochastic environments that might result in jerky policies, researchers have explored different avenues. One approach is to use latent environment representations [5] to reduce input noise, since random noise makes agents more prone to jerky motions [4]. Sampling temporal coherent noise during training has also been proposed to increase robustness against noise [6, 7]. Randomization has also been explored [8], so that policies can better adapt to real-world environments and generalize. Reward shaping has also been proposed, by mimicking smooth reference trajectories [9] or learning additive feedback together with a trajectory generator [10]. Then, researchers in [11] propose a divide-and-conquer method that splits the initial state distribution. Finally, data augmentation [12, 13] has also been considered as a form of increasing robustness and better generalization. Most of these approaches require manual tuning efforts to work. Specific applications in which jerk mitigation is an objective include driving [14] and human-robot co-manipulation [15].

6.884 Computational Sensorimotor Learning - Class project.

2 Methods

We now explain each of the strategies considered in this study. In all cases, we assume the context of an agent interacting with an environment that follows a Markov Decision Process. The goal of the agent is to learn a stochastic policy π that determines the action to take for each environment observation, $a_t \sim \pi(o_t)$. The observation o_t is related to the environment's true state, s_t. Given the focus on smooth trajectories, instead of taking action a_t, the agent might decide to take a different action a'_t that reduces jerkiness. The agent receives a reward r_t after taking action a'_t in state s_t and improves the policy based on that reward.

2.1 Previous action as input to the policy

The first approach consists of using the previous action taken by the agent as an input to the policy, i.e., $o_t = \{s_t, a'_{t-1}\}$. In this case the action taken is the outcome of the policy, $a'_t = a_t$.

2.2 Moving average of the action

In the second method, instead of taking action a_t, we first "filter" it by performing moving averages over past actions. Given a window size M, the action taken is computed as $a'_t = \frac{1}{M}\sum_{i=0}^{M-1} a_{t-i}$. In this case the observation corresponds to the environment's true state, $o_t = s_t$.

2.3 Action difference

In an attempt to reduce the variance of the policy's actions, the third strategy consists in changing the action space of the policy to output the difference between consecutive environment actions. Specifically, $a'_t = a'_{t-1} + a_t$, which in turn entails that $a'_t = \sum_{i=0}^{t} a_i$. Again, the observation corresponds to the environment's true state.

2.4 Reward shaping

Finally, we also consider a reward shaping strategy that can complement each of the presented approaches, since it directly impacts the reward rather than the taken action. In these cases, instead of taking the true environment reward r_t, we add a penalty term based on the observed jerk, i.e., $r'_t = r_t - \beta \cdot J_t$, where J_t is the observed jerk at timestep t and β is a weighing factor. We can combine this strategy with any of the three approaches presented, the only requirement is having access to the observed jerk during training. This might limit the use of this method to simulation-based environments.

3 Results

In this section we show the results of each method introduced in Section 2 for four specific robotics-inspired locomotion environments. A repository with the code for this project can be found at `https://github.com/jjgarau/CSLProject`.

3.1 Environments

Four different use cases are chosen for this project, all of them from PyBullet's [16] environment suite. They are depicted in Figure 1: *Ant*, *Hopper*, *Half Cheetah*, and *Humanoid* environments. The task in these environments is for a robot to learn how to walk and constantly move forward. The number of joints and shape of each robot is different. An episode lasts until the robot falls to the ground. The **state** in these environments consists of multiple variables corresponding to the position of the robot's body, the XYZ velocity of the body, its roll, its pitch, the relative position and velocity of each joint, and additional variables indicating whether each foot of the robot is making contact to the ground or not.

The **action** taken in the environment is represented by n continuous variables and corresponds to the torque exercised in each of the n joints. This number corresponds to 8, 3, 6, and 17 for the *Ant*, *Hopper*, *Half Cheetah*, and *Humanoid* environments, respectively. The **reward** returned by the

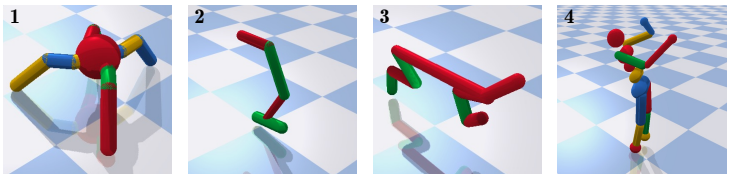

Figure 1: PyBullet environment suite: **1.** Ant, a quadrupedal robot with 8 joints, **2.** Hopper, a jumper robot with 3 joints, **3.** Half Cheetah, a bipedal robot with 6 joints, and **4.** Humanoid, a human-like robot with 17 joints. The specific task in all environments consists of learning how to walk and move forward.

environment is a linear combination of 4 metrics: a bonus for being alive, the distance travelled, the cost of electricity (negative), and the strain on the joints (negative). We set the minimum possible reward to zero.

We then define two different metrics for **jerk** in these environments, corresponding to the jerk in the body and the joints, respectively. To that end, we first keep track of the body XYZ velocities, $\boldsymbol{v}_t$, throughout an episode. We also keep track of the joint relative velocities, $\omega_{i,t}$, for $i = 1, \ldots, n$. Once the episode ends, and assuming $\Delta t = 1$, we compute the accelerations as $\boldsymbol{a}_t = \boldsymbol{v}_t - \boldsymbol{v}_{t-1}$ and $\alpha_{i,t} = \omega_{i,t} - \omega_{i,t-1}$, respectively. We repeat the same procedure to compute the jerk of the body over time, $\boldsymbol{j}_t = \boldsymbol{a}_t - \boldsymbol{a}_{t-1}$, and the joints', $\Omega_{i,t} = \alpha_{i,t} - \alpha_{i,t-1}$. The first metric, the jerk of the body, is then computed as $J_{body} = \max_t \|\boldsymbol{j}_t\|$. The metric corresponding to the jerk in the joints is in turn computed as $J_{joints} = \max_t \left(\max_i |\Omega_{i,t}| \right)$.

3.2 Baseline policy

To better understand the performance of each of the methods, we also compare them against a baseline agent, in which $s_t = o_t$ and $a'_t = a_t$ — i.e., no modifications. All agents are then trained using PPO [17] with parameters $\gamma = 0.99$, $\lambda = 0.97$, and $\epsilon = 0.2$. We train each agent for 200 epochs, each epoch consisting of 4,000 timesteps, and fix a maximum episode duration of 1,000 timesteps. MLP networks with two 64-unit hidden layers and Tanh activation functions are used for both the policy and value networks.

3.3 Recurrent policy

One option that has not been addressed is the use of latent spaces. While the effort required to explore this idea in depth is out of the scope of this project, we also implement and include a RNN-based policy in the comparison to provide an idea of its potential performance. Specifically, we use a 2-layer GRU [18] with a hidden space of size 64. We also train this policy using the PPO algorithm with the parameters described in the previous section.

3.4 Comparison without reward shaping

We first compare the performance of the baseline, the recurrent policy, and the methods presented (Moving average, Previous action, and Action difference) without reward shaping. The evolution of the average return, body jerk, and joint jerk for each method in the *Ant* and *Humanoid* environments is shown in Figure 2 and Figure 3, respectively, with results aggregated over 5 seeds. In the specific case of the Moving average policy, a window size $M = 10$ is used.

Looking at the curves we realize the specific test environment has a significant effect in the results. In the case of the *Ant* environment, when not using reward shaping none of the methods achieves substantial lower jerk compared to the baseline. The Previous action strategy shows lower body jerk and higher return, but the joint jerk is much larger than baseline's, as seen in Table 1. The Action difference method is very unstable during training, the reward is almost zero and the jerk is high.

The curves look very different in the case of the *Humanoid* environment. All policies but Action difference show a natural tendency to reduce jerk while increasing rewards. Among them, in the last epoch the Moving average policy is able to achieve the same return as Baseline's and the lowest joint

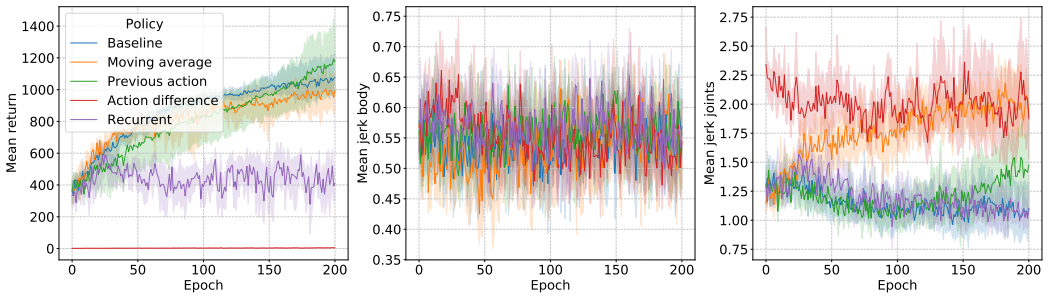

Figure 2: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Ant* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is not used*. Average and 95% CI across five seeds is shown.

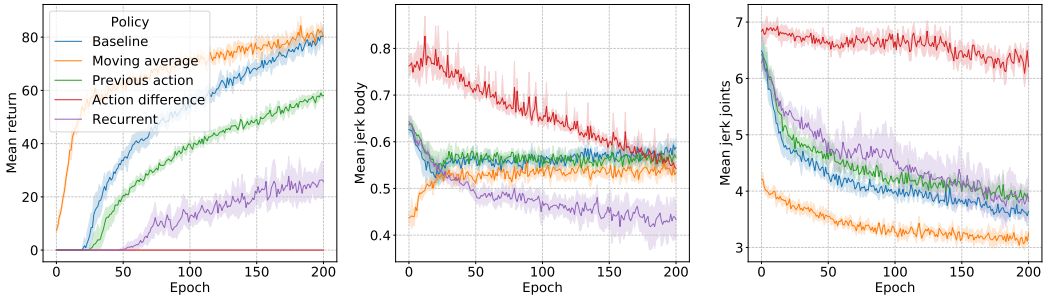

Figure 3: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Humanoid* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is not used*. Average and 95% CI across five seeds is shown.

Table 1: Average return, body jerk, and joints jerk during the last epoch in the *Ant* environment for the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference.

	Without reward shaping					**With reward shaping**				
Last ep.	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**
Return	1077	994	**1176**	5	409	10	48	22	1	**58**
Body	0.56	0.55	**0.53**	0.54	0.57	0.95	0.91	0.89	0.88	**0.70**
Joints	**1.10**	1.93	1.48	1.87	**1.10**	**0.47**	0.64	1.07	1.01	0.92

jerk, although the tendency of the curves indicates other policies could do better if we trained for more epochs. The Baseline is already a strong competitor, as seen in Table 2. The curves for the *Hopper* and *Half Cheetah* environments can be found in Figure 10 and Figure 12 in the appendix, respectively. Likewise, Table 3 and Table 4 in the appendix summarize their results.

Table 2: Average return, body jerk, and joints jerk during the last epoch in the *Humanoid* environment for the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference.

	Without reward shaping					**With reward shaping**				
Last ep.	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**
Return	**80**	**80**	58	0	26	0	**14**	0	0	0
Body	0.59	0.53	0.57	0.54	**0.43**	0.61	**0.29**	0.61	0.47	0.57
Joints	3.64	**3.25**	3.88	6.21	3.81	4.36	**2.50**	4.44	5.44	5.27

3.5 Comparison with reward shaping

Now we repeat the same comparison but adding the reward shaping approach described in Section 2.4 to the training loop. To compute the jerk penalty we take the instantaneous jerk measurements for both the body and joints and compute a single value $J_t = \|\boldsymbol{j}_t\| + \max_i |\Omega_{i,t}|$. We use a penalty weight $\beta = 50$. The results adding the jerk penalty are shown in Figure 4 and Figure 5 for the *Ant* and *Humanoid* environments, respectively. The return shown corresponds to the return without the jerk penalty, so it can be compared to the values shown in Figure 2 and Figure 3. Table 1 and Table 2 also show the last epoch values for this comparison.

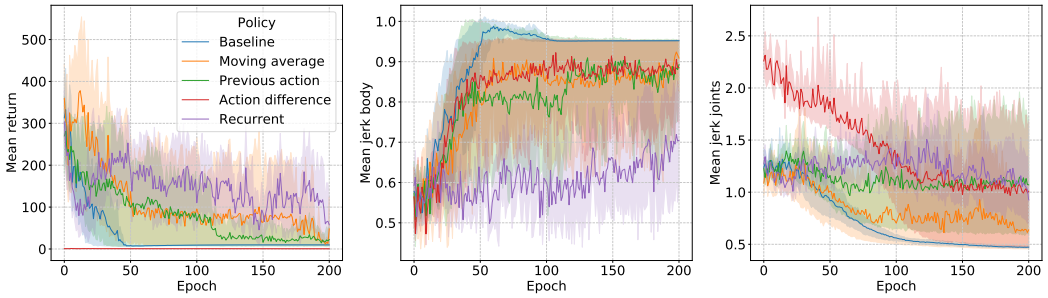

Figure 4: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Ant* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is used*. Average and 95% CI across five seeds is shown.

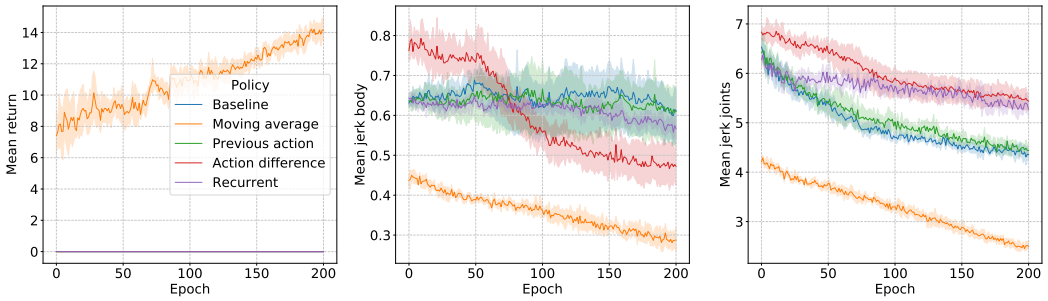

Figure 5: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Humanoid* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is used*. Average and 95% CI across five seeds is shown.

In this case we can see the methods implemented generally achieve jerk reduction – specifically joint jerk – compared to the baseline. However, looking at the original return we see that agents fail to complete the original task. This suggests the way in which reward shaping has been implemented is not appropriate for the overall goal – reduce jerk *and* move forward. In addition, different environments display different behaviors, raising the question of how generalizable these approaches are. Still, we can see interesting behaviors in the Moving average policy, as it achieves the most robust results when reward shaping is used. This is specifically seen for the *Humanoid* environment, and the *Hopper* and *Half Cheetah* environments, as seen in Figure 11 and Figure 13 in the appendix, respectively. The fact that joint jerk is prioritized might not be negative, as the joints is where the actuators are and are prone to deteriorate faster. To better appreciate how policies compare and trade to each other, Figure 6 and Figure 7 show the return against the jerk for each independent run carried out in the *Ant* and *Humanoid* environments, respectively. Figure 8 and Figure 9 in the appendix display the same analyses for the *Hopper* and *Half Cheetah* environments, respectively.

4 Conclusion and future work

This project has explored different strategies to reduce policy jerkiness and improve smooth control in model-free RL. The approaches considered have been using the previous action as input to the

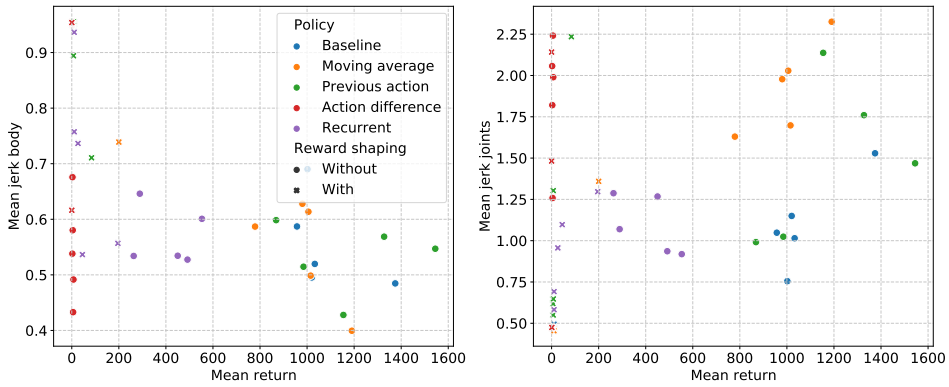

Figure 6: Mean return against mean body jerk and mean joints jerk for each of the five seeds during the last epoch in the *Ant* environment. It compares the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference. Connects with data shown in Table 1.

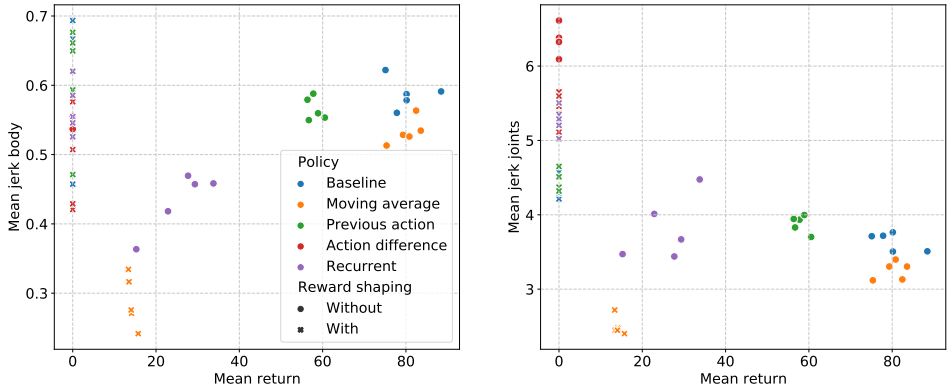

Figure 7: Mean return against mean body jerk and mean joints jerk for each of the five seeds during the last epoch in the *Humanoid* environment. It compares the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference. Connects with data shown in Table 2.

policy, carrying out a moving average of the action, and having the policy compute the difference between consecutive actions. In addition, a reward shaping strategy that directly penalizes jerk has been proposed. All of these methods have been compared against a baseline and recurrent policies, using four different locomotion environments.

The results have shown that reducing jerkiness while keeping returns stable is possible in some cases. However, the approach that works better and the degree to which it does so substantially depends on the specific environment, suggesting that no method is able to achieve generalization. Among the approaches considered, adding reward shaping to the policy has been shown to be the hardest to tune. When using reward shaping, the Moving average policy shows the most robust behavior.

This project has different future work directions: 1) tuning the reward shaping approach, possibly adding a dynamic weight that better adapts to the training process, 2) investigating other methods that generalize better across environments, 3) improving the loss function by considering the policy's entropy, and 4) extending the work to other types of task, like manipulation.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[3] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. mar 2020.

[4] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, apr 2021.

[5] Annie Xie, James Harrison, and Chelsea Finn. Deep Reinforcement Learning amidst Lifelong Non-Stationarity. jun 2020.

[6] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to Walk via Deep Reinforcement Learning. dec 2018.

[7] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

[8] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810. IEEE, may 2018.

[9] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic. *ACM Transactions on Graphics*, 37(4):1–14, aug 2018.

[10] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators. In *Conference on Robot Learning*, pages 916–926. PMLR, 2018.

[11] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-Conquer Reinforcement Learning. nov 2017.

[12] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network Randomization: A Simple Technique for Generalization in Deep Reinforcement Learning. oct 2019.

[13] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 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)*, pages 23–30. IEEE, sep 2017.

[14] Yuan Lin, John McPhee, and Nasser L. Azad. Anti-Jerk On-Ramp Merging Using Deep Reinforcement Learning. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 7–14. IEEE, oct 2020.

[15] Fotios Dimeas and Nikos Aspragathos. Reinforcement learning of variable admittance control for human-robot co-manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1011–1016. IEEE, sep 2015.

[16] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[18] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. jun 2014.

Appendix

Result tables

Table 3: Average return, body jerk, and joints jerk during the last epoch in the *Hopper* environment for the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference.

	Without reward shaping					With reward shaping				
Last ep.	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**
Return	**1678**	895	1557	1146	37	16	**20**	19	15	**20**
Body	1.50	1.80	1.62	2.14	**0.30**	0.25	**0.12**	0.24	0.18	0.41
Joints	5.70	4.98	5.70	5.52	**1.45**	0.51	**0.31**	0.63	0.43	1.19

Table 4: Average return, body jerk, and joints jerk during the last epoch in the *Half Cheetah* environment for the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference.

	Without reward shaping					With reward shaping				
Last ep.	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**	**Base.**	**Mov. avg.**	**Prev. act.**	**Act. diff.**	**Rec.**
Return	**1448**	599	1018	0	0	0	0	0	0	0
Body	0.64	0.79	0.68	0.93	**0.63**	0.81	**0.66**	0.81	0.84	0.72
Joints	5.30	5.08	**4.78**	5.35	5.27	4.15	**3.37**	4.50	4.82	3.77

Return vs. Jerk plots

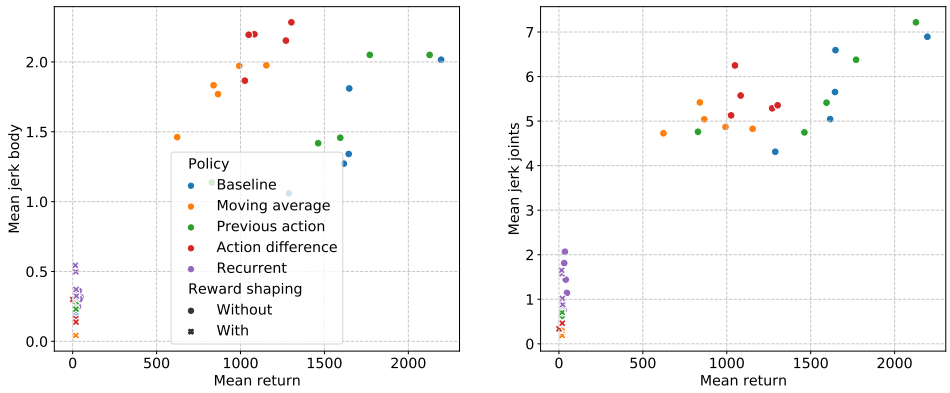

Figure 8: Mean return against mean body jerk and mean joints jerk for each of the five seeds during the last epoch in the *Hopper* environment. It compares the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference. Connects with data shown in Table 3.

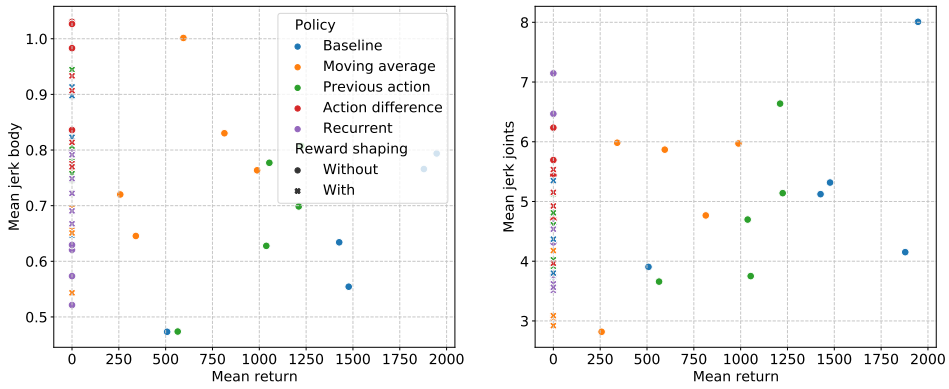

Figure 9: Mean return against mean body jerk and mean joints jerk for each of the five seeds during the last epoch in the *Half Cheetah* environment. It compares the baseline and recurrent policies and the three jerk mitigation strategies considered: Moving average, Previous action, and Action difference. Connects with data shown in Table 4.

Training plots

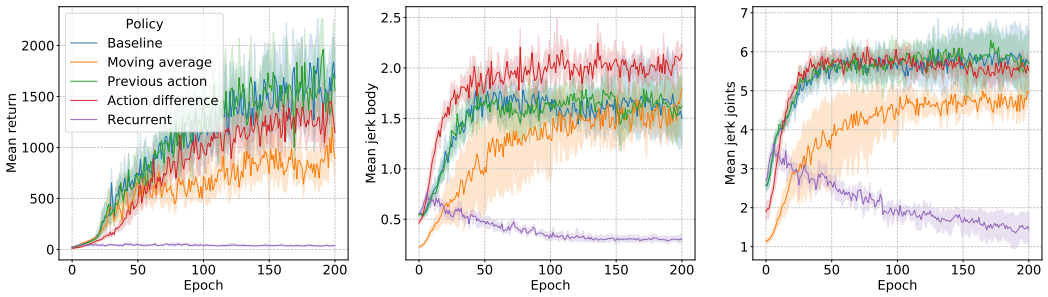

Figure 10: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Hopper* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is not used*. Average and 95% CI across five seeds is shown.

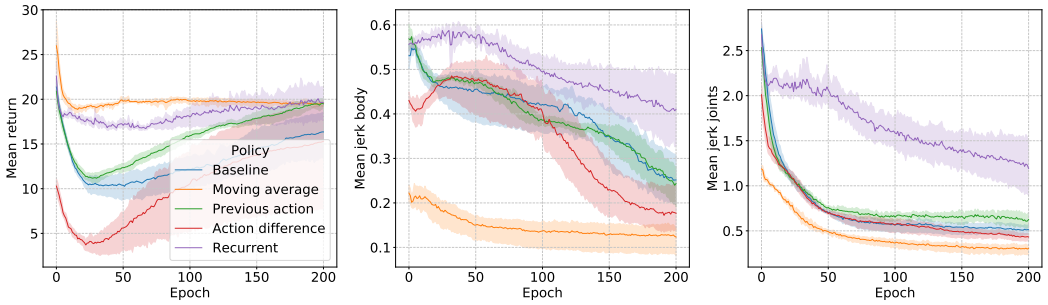

Figure 11: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Hopper* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is used*. Average and 95% CI across five seeds is shown.

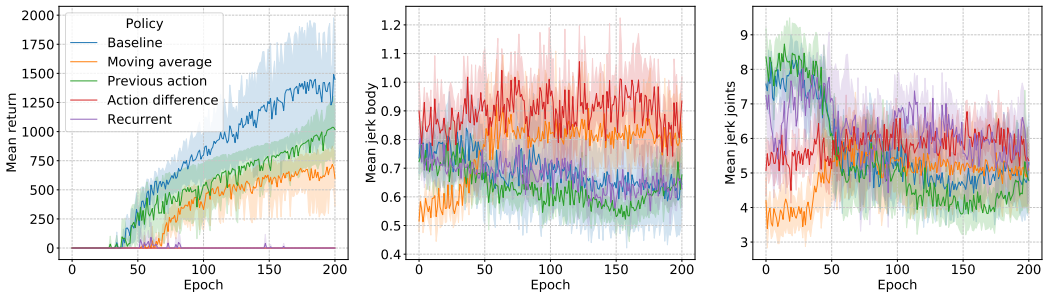

Figure 12: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Half Cheetah* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is not used*. Average and 95% CI across five seeds is shown.

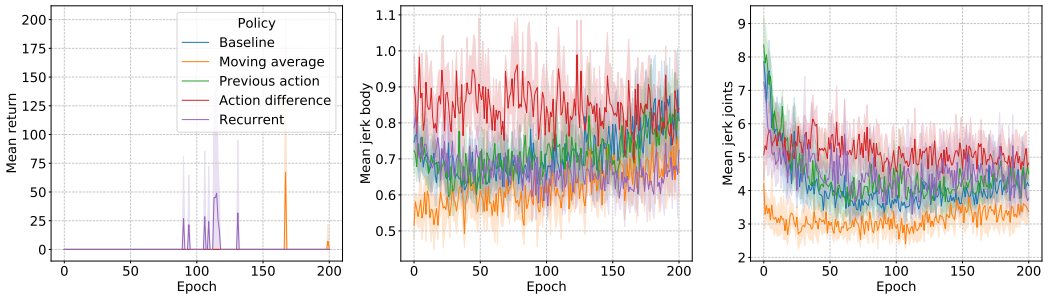

Figure 13: Mean return (left), body jerk (center), and joints jerk (right) per epoch in the *Half Cheetah* environment for the baseline policy and the three jerk mitigation strategies considered when *reward shaping is used*. Average and 95% CI across five seeds is shown.