Multi-Constraint Deep Reinforcement Learning for Smooth Action Control

Guangyuan Zou[1,2], **Ying He**[1,2,*], **F. Richard Yu**[1,2], **Longquan Chen**[1,2], **Weike Pan**[1] and **Zhong Ming**[1]

[1]College of Computer Science and Software Engineering, Shenzhen University, P.R. China
[2]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China
1900271046@email.szu.edu.cn, {heying, yufei}@szu.edu.cn, chenlongquan2019@email.szu.edu.cn, {panweike, mingz}@szu.edu.cn

Abstract

Deep reinforcement learning (DRL) has been studied in a variety of challenging decision-making tasks, e.g., autonomous driving. However, DRL typically suffers from the action shaking problem, which means that agents can select actions with big difference even though states only slightly differ. One of the crucial reasons for this issue is the inappropriate design of the reward in DRL. In this paper, to address this issue, we propose a novel way to incorporate the smoothness of actions in the reward. Specifically, we introduce sub-rewards and add multiple constraints related to these sub-rewards. In addition, we propose a multi-constraint proximal policy optimization (MCPPO) method to solve the multi-constraint DRL problem. Extensive simulation results show that the proposed MCPPO method has better action smoothness compared with the traditional proportional-integral-differential (PID) and mainstream DRL algorithms. The video is available at https://youtu.be/F2jpaSm7YOg.

1 Introduction

Deep reinforcement learning (DRL) has been widely recognized as a promising way to learn optimal policies in a wide range of practical decision-making and controlling domains, such as video games [Mnih *et al.*, 2015] and robotic manipulation [Siekmann *et al.*, 2021; Lillicrap *et al.*, 2016]. One of the most attractive characteristics of DRL is the ability to learn optimal policies in a model-free manner, even in complex environments with high-dimensional state and action space.

However, DRL typically suffers from the action shaking problem, particularly in control problems with continuous actions, which means that agents can select actions with big difference even though states only slightly differ. Although the agent can achieve good task-specific rewards in simulations, action shaking may strongly affect the user experience in many practical interactive applications. Some works [Neunert *et al.*, 2019; Chen *et al.*, 2021] propose to output an "act-or-repeat" binary result to decide if the action at the previous step should be repeated to reduce the action shaking by keeping the last action. However, since the "act-or-repeat" decision is examined at every step depending on the current environment state, this results in the loss of action diversity. Song's work [Song *et al.*, 2020] adds a regularization term to the objective function of actor-critic methods to constrain the difference between neighboring actions to increase action smoothness in autonomous driving tasks.

More crucially, action shaking has been one of the main reasons limiting the deployment of DRL agents to real-world domains such as autonomous driving scenarios, where safety is the very first requirement [Chen *et al.*, 2021]. The motion control system of autonomous driving requires high smoothness of action, which is a crucial part of the autonomous driving system. It consists of lateral control and longitudinal control, which are coupled together. Lateral control has a longitudinal braking effect, and longitudinal control affects lateral acceleration when turning, etc. At present, lateral and longitudinal cooperative control mainly uses traditional control algorithms such as PID control [Bellman, 2015], although PID controllers have limited scenario generalization and rely on hyperparameter tuning in different traffic scenarios. DRL has the ability to respond to complex scenarios compared with the traditional control algorithms. If the problem of action shaking in DRL can be solved, it will be of great value for real-world control tasks in autonomous driving.

Although some excellent works have studied the action shaking issue, most existing works [Neunert *et al.*, 2019; Chen *et al.*, 2021] focus on designing the *action space* of DRL to solve it. Consequently, the *reward* in DRL is largely ignored. However, the reward is a key factor that guides DRL agents to search the optimal policy. In addition, action smoothness metrics are often mutually exclusive with task rewards, so it is difficult to achieve the action smoothness objective and other objectives simultaneously.

To fill this gap, we propose a novel framework, multi-constraint deep reinforcement learning, to solve the action shaking problem. The main contributions of this paper are as follows:

- We propose a novel way to incorporate the smoothness of actions in the reward of DRL. Specifically, we introduce sub-rewards and add multiple constraints related to these sub-rewards in DRL to improve the smoothness of

*Corresponding author

continuous motions.

- We propose a multi-constraint proximal policy optimization (MCPPO) method to solve the multi-constraint DRL problem.
- Extensive simulation results in a wide range of autonomous driving tasks show that our proposed approach can achieve better action smoothness compared with the traditional proportional-integral-differential (PID) and mainstream DRL algorithms.

2 Background and Related Work

In this section, we introduce constrained Markov decision process, followed by constrained reinforcement learning.

2.1 Constrained Markov Decision Process

Constrained Markov decision process (CMDP) [Altman, 1999] extends MDP [Sutton and Barto, 1998] to incorporate constraints into DRL. We define CMDP as the tuple $< \mathcal{S}, \mathcal{A}, R, \mathcal{C}, \boldsymbol{d}, \mathcal{P}, \rho_0, \boldsymbol{\gamma}, H >$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{C} = \{C^i\}_{i=1}^m$ is a set of cost functions $C^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$, $\boldsymbol{d} : \mathbb{R}^{|\mathcal{C}|}$ is the vector of constraint thresholds corresponding to every cost function C^i, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\in[0,1]}$ is the state transition probability distribution, $\rho_0 : \mathcal{S} \to \mathbb{R}_{\in[0,1]}$ is the initial state distribution, $\boldsymbol{\gamma}$ is the discount factor vector for reward and costs that we assume $\boldsymbol{\gamma} \in \mathbb{R}^{|\mathcal{C}|+1}_{\in[0,1)}$ and H is the episode horizon. The agent interacts with the CMDP at discrete timesteps by performing its policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\in[0,1]}$, generating a trajectory of states and actions, $\tau = (s_0, a_0, s_1, a_1, \dots s_H, a_H)$, where $s_0 \sim \rho(s)$, $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. The objective of a constrained reinforcement learning agent is to find a policy that maximizes the expected cumulative reward, denoted by

$$J^R(\pi) = \mathbb{E}_{\tau\sim\rho_\pi}[\sum_{t=0}^{H}(\boldsymbol{\gamma}^0)^t R(s_t, a_t, s_{t+1})]. \tag{1}$$

Moreover, there is a constraint for every cost function C^i,

$$J^{C^i}(\pi) = \mathbb{E}_{\tau\sim\rho_\pi}[\sum_{t=0}^{H}(\boldsymbol{\gamma}^{i+1})^t C^i(s_t, a_t, s_{t+1})] \le d^i, \tag{2}$$

where ρ_π is the state-action marginals of the trajectory distribution induced by policy π, and d^i is the desired threshold used to constrain the cumulative discount cost. Hence, the optimal policy in CMDP is

$$\begin{aligned} &\pi^\star = \operatorname*{argmax}_\pi J^R(\pi) \\ &s.t. \quad J^{\mathcal{C}}(\pi) \le \boldsymbol{d}. \end{aligned} \tag{3}$$

2.2 Constrained Reinforcement Learning

Constrained RL solves the CMDP problem by the Lagrange multiplier method [Chow *et al.*, 2017; Tessler *et al.*, 2019]. In the Lagrange multiplier method, CMDP is converted into an equivalent unconstrained problem. In addition to the objective, a penalty term is added for infeasibility, thus making infeasible solutions sub-optimal. Given CMDP in Eq.(3), the unconstrained problem is

$$\min\max \mathcal{L}(\pi^\star, \boldsymbol{\lambda}^\star) = \min_{\boldsymbol{\lambda}\in\mathbb{R}_+^{|\mathcal{C}|}} \max_\pi [J^R(\pi) - \boldsymbol{\lambda}(J^{\mathcal{C}}(\pi) - \boldsymbol{d})], \tag{4}$$

where $\boldsymbol{\lambda} \in \mathbb{R}_+^{|\mathcal{C}|}$ is the Lagrange multiplier vector. Eq.(4) is usually solved by the primal-dual algorithms [Boyd and Vandenberghe, 2014; Chow *et al.*, 2017; Tessler *et al.*, 2019; Achiam *et al.*, 2017]. The primal problem is denoted by

$$P^\star : \pi^\star = \operatorname*{argmax}_\pi \mathcal{L}(\pi, \boldsymbol{\lambda}^\star). \tag{5}$$

And the dual problem is denoted by

$$D^\star : \boldsymbol{\lambda}^\star = \operatorname*{argmin}_{\boldsymbol{\lambda}\in\mathbb{R}_+^{|\mathcal{C}|}} \mathcal{L}(\pi^\star, \boldsymbol{\lambda}). \tag{6}$$

Some works of CRL focus on how to optimize the dual variables ($\boldsymbol{\lambda}$) to satisfy cost constraint during policy learning, such as constrained policy optimization (CPO) [Achiam *et al.*, 2017], the PID Lagrangian method [Stooke *et al.*, 2020], projection-based constrained policy optimization (PCPO) [Yang *et al.*, 2020], etc. And these constraints are usually related to safety. The main differences in our work are that our method is committed to solve the multi-constraint RL problems. This can be explained in two specific ways. Firstly, we use cost constraints to define the problem objectives to search policy close to our expectations without requiring constraint satisfaction in the training process. Secondly, our experiments validate the effectiveness of our approach to solving multi-constraint reinforcement learning problems in the autonomous driving control tasks, while most existing works are based on experiments with single-constraint setting of cost or sum of costs.

Paternain's work [Paternain *et al.*, 2019] proves that the problems in Eq.(4) have zero duality gap $\Delta = D^\star - P^\star = 0$ under Slater's conditions. We note that the satisfiability of Slater's conditions is highly dependent on our constraint settings. Convergence proofs have relied upon updating the multiplier more slowly than the policy parameters [Tessler *et al.*, 2019], implying many constraint-violating policy iterations may occur before the penalty comes into full effect. And Tessler's work [Tessler *et al.*, 2019] introduces the fixed Lagrangian multipliers to guide the policy, which is useful for multi-constraint RL problems. However, constraint satisfaction is required during training in the above work. By contrast, we do Lagrange multipliers adaption during training.

3 Problem Formulation

In this section, we introduce CMDP formulation of an autonomous driving smoothness control problem, including state space, action space, reward function and constraints. Autonomous driving tasks occur in environments that require artificial rewards (no nature reward $r_t^e = 0$). Figure 1 reveals the insertion points of the reward and constraint design during the interaction of an agent with its environment.

3.1 State Space and Action Space

The autonomous driving control task is to control the ego vehicle to drive smoothly under different road conditions. The

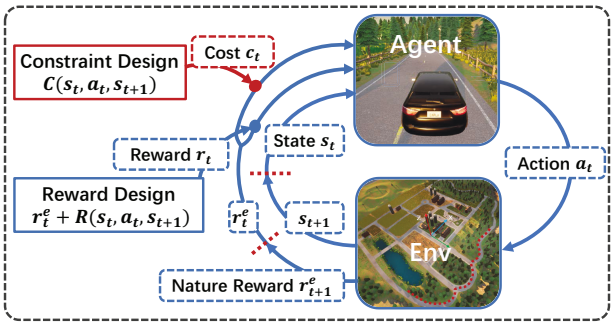


Figure 1: An agent-environment interaction architecture that introduces reward design and constraint design. In most environments (e.g., autonomous driving), there are no nature rewards (extrinsic rewards [Badia *et al.*, 2020], rewards from environment). Rewards and constraints are artificially designed based on human objectives.

state space includes the ego vehicle state and some waypoints state. Please refer to Section A.1 of Appendix[1] for details. The continuous action space includes the longitudinal control about throttle or brake and lateral control about steering. Please refer to Appendix A.2 for details.

3.2 Reward

Reward design is critical, and it directly affects the direction of policy optimization. A trained DRL agent reaches return convergence, but sometimes the policy does not perform as we expect, usually due to a reward design that does not accurately reflect our objective.

For the problem of autonomous driving control, we provide the following reward design

$$R = R^t + R^s + R^a + R^c, \tag{7}$$

where R^t is the task reward, R^s is the safety reward, R^a is the reward of action magnitude and R_c is the comfort reward. R^t and R^s are defined separately for task and safety objectives, which are necessary rewards for autonomous driving, and are described in Appendix A.3. But both of them have no explicit incentive to have action smoothly, even reward hacking [Amodei *et al.*, 2016] for sacrificing smoothness for higher reward. The action smoothness should be considered from at least two aspects: action magnitude and comfort. It is not enough to consider only one. Although there is a correlation between action magnitude and comfort, there are cases where the ego vehicle is subjected to small forces but large action magnitude.

The reward of action magnitude has both longitudinal control (a^0) and lateral control (a^1) parts

$$R_t^a = \alpha_4 R^{a^0} + \alpha_5 R^{a^1}. \tag{8}$$

At time step t, the reward of the dth dimension action component magnitude $R_t^{a^d}$ is

$$R_t^{a^d} = \sum_{i=t-|H_a|}^{t} [\text{clip}(\text{sign}(a_{i-1}^d \cdot a_i^d), -1, 0)(a_{i-1}^d - a_i^d)^2], \tag{9}$$

[1]https://github.com/GyChou/mcppoElegantRLforCarla.

where H_a is a list of historical actions of a certain length. The effect in Eq.(9) is to penalize the action shaking. The comfort reward R^c is related to acceleration and jerk, both of which have longitudinal control and lateral control parts

$$R_t^c = \alpha_6 R^{acc^0} + \alpha_7 R^{acc^1} + \alpha_8 R^{j^0} + \alpha_9 R^{j^1}. \tag{10}$$

At time step t, the acceleration reward for the dth dimension action is

$$R_t^{acc^d} = -\frac{\Delta t}{g} \sum_{i=t-\text{int}(\frac{1}{\Delta t})}^{t} |acc_i^d|^\beta, \tag{11}$$

and the jerk reward for the dth dimension action is

$$R_t^{j^d} = -\frac{\Delta t}{g} \sum_{i=t-\text{int}(\frac{1}{\Delta t})}^{t} |acc_i^d - acc_{i-1}^d|^\beta, \tag{12}$$

where Δt is the decision interval, g is the gravitational acceleration ($9.8m/s^2$), acc_i^d is the acceleration of the dth action dimension at time step i and $\beta = 2$ is a hyperparameter. Jerk indicates the degree of acceleration change. The role of acceleration and jerk rewards are to penalize the degree of force on the ego vehicle. The less the vehicle is subjected to lateral and longitudinal forces during driving, the better its comfort.

3.3 Constraints

Rather than iteratively adjusting the reward weights, we choose to add constraints to replace the reward weighting adjustment. The reason is that some sub-rewards are related to constrained objectives. Given an example about keeping lane reward in our problem, we want the ego vehicle to drive within the lane, not strictly in the centerline of the lane. More generally, for example, limiting collisions of an agent, limiting the times of an agent can do certain actions, etc. Generally, it is easy to find the constraints, but it is difficult to adjust the weights of the rewards about the constraints. So, it may be a more reasonable way to leave the weight adjustment to the algorithm by constructing constraints directly for the problem. Constrained RL can adaptively adjust the value between various costs and reward thus avoiding the work of adjusting weights.

In our problem, we establish three constraints, including action magnitude, comfort and safety. Each constraint is

$$J^{C^i}(\pi) = \mathbb{E}_{\tau \sim \rho_\pi}[\frac{1}{H} \sum_{t=0}^{H} C_t^i] \leq d^i, \tag{13}$$

which denotes that the average of every cost in $\mathcal{C}$ is constrained by threshold $\boldsymbol{d} = [0.05, 0.02, 0.12]$. We use Eq.(13) to replace Eq.(2) because it is difficult to determine the threshold of the cumulative discounted cost in our problem considering the impact of the episode horizon H. The smoothness of lateral action is important for autonomous driving, and its shaking is amplified with increasing longitudinal speed. So, the first cost $C_t^1 = \frac{1}{|H_a|} \sum_{i=t-|H_a|}^{t} (a_{i-1}^1 - a_i^1)^2$ is about lateral action magnitude, the second cost $C_t^2 = -R_t^{acc^1}$ is about lateral acceleration. In addition, we design the third cost $C_t^3 = -R_t^{LaD}$ to prevent vehicles from off the lane.

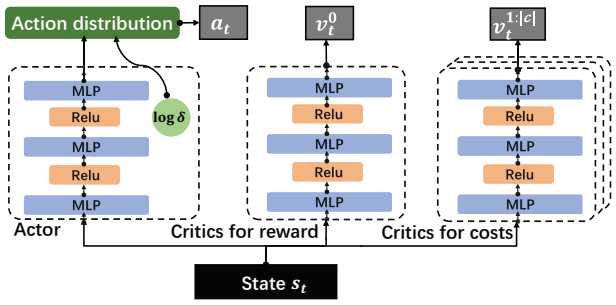


Figure 2: The network architecture of our MCPPO.

4 Multi-constraint Proximal Policy Optimization

In this section, we describe multi-constraint proximal policy optimization (MCPPO) to solve the multi-constraint RL problem in Section 3. Compared with standard PPO, MCPPO has three changes: 1) MCPPO adds several critic networks for estimating cost values, 2) MCPPO optimizes policy optimization with reward and costs advantage, 3) and MCPPO optimizes the Lagrange multipliers $\boldsymbol{\lambda}$ to trade-off reward and costs values. Figure 2 shows the network policies of MCPPO, which is trained by the prime-dual method, including finding optimal policy parameters to solve the primal problem and finding optimal Lagrange multipliers to solve the dual problem.

4.1 Solve the Primal Problem

For balancing improvements between different directions based on rewards and multi-cost, we propose a re-scaled objective for solving the primal problem in Eq.(5):

$$\theta^{\pi}_{new} = \underset{\theta^{\pi}}{\operatorname{argmax}}\, \mathbb{E}_{\tau\sim\rho_{\pi}} \frac{wJ^{R}(\theta^{\pi}) - \sum_{i=1}^{|\mathcal{C}|} \boldsymbol{\lambda}^{\boldsymbol{i}} J^{C^i}(\theta^{\pi})}{w + \operatorname{sum}(\boldsymbol{\lambda})}, \tag{14}$$

where $w = \max(1, 2\cdot \operatorname{sum}(\boldsymbol{\lambda}))$ is a design with reward weight adaption to reduce large change in parameters θ^{π} when $\boldsymbol{\lambda}$ is large. We use the PPO [Schulman *et al.*, 2017] clipped surrogate objective to maximize the advantage of reward and minimize the advantage of costs. We prefer samples with a small reward advantage

$$J^{R}(\theta^{\pi}) = \min(\operatorname{clip}(r_t(\theta^{\pi}), 1-\epsilon, 1+\epsilon)\widehat{\boldsymbol{A}}^{0}_{t}, r_t(\theta^{\pi})\widehat{\boldsymbol{A}}^{0}_{t}), \tag{15}$$

which contributes to the stable improvement of policy reward. And we choose samples with larger costs advantage to stably reduce the policy costs, denoted by

$$J^{C_i}(\theta^{\pi}) = \max(\operatorname{clip}(r_t(\theta^{\pi}), 1-\epsilon, 1+\epsilon)\widehat{\boldsymbol{A}}^{i}_{t}, r_t(\theta^{\pi})\widehat{\boldsymbol{A}}^{i}_{t}). \tag{16}$$

$r_t(\theta^{\pi}) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the important sampling ratio. $\widehat{\boldsymbol{A}}$ is the vector of advantage estimation about the reward and costs, $\widehat{\boldsymbol{A}}^{0}_{t}$ is about the reward and $\widehat{\boldsymbol{A}}^{1:|\mathcal{C}|}_{t}$ is about the costs. GAE [Schulman *et al.*, 2016] is used to calculate the advantage vector $\widehat{\boldsymbol{A}}$ by

$$\widehat{\boldsymbol{A}}_t = \sum_{l=t}^{H} (\boldsymbol{\gamma\lambda}^{\boldsymbol{GAE}})^{l-t}[(M_l) + \boldsymbol{\gamma V^{\pi}}(s_{t+1}) - \boldsymbol{V^{\pi}}(s_t)], \tag{17}$$

where $M_l = [R_t, C^1_t, C^2_t, \ldots, C^{|\mathcal{C}|}_t]$ is a vector concatenated by reward and costs, and $\boldsymbol{V^{\pi}}(s_t)$ is the value function about reward and costs. Loss function of reward and costs value prediction is calculated by $SmoothL1Loss$ [Girshick, 2015]:

$$L^{\boldsymbol{V}} = \sum_{i=0}^{|\mathcal{C}|} (\mathbb{E}_{\tau\sim\rho_{\pi}}[\text{SmoothL1}(\boldsymbol{V^{\pi i}}(s_t), \sum_{l=t}^{H} (\boldsymbol{\gamma}^{i})^{l-t} M^{i}_{t})]). \tag{18}$$

Eq.(16) is also suitable for the performance of average cost Eq.(13). It is because the objective in primal problems has been simplified to minimizing the performance about costs, and the optimization direction of minimizing the cumulative discounted cost is the same as that of the average cost. However, the gap between the altered primal problem and the original dual problem needs to be proved.

4.2 Solve the Dual Problem

In solving the dual problem, Lagrangian relaxation techniques are used to process inequality constraints, such as Eq.(2), which is difficult for adjusting reward weights. The objective of the dual problem is

$$\boldsymbol{\lambda}^{\star} = \underset{\boldsymbol{\lambda}\in\mathbb{R}^{|\mathcal{C}|}_{+}}{\operatorname{argmin}}\, \boldsymbol{\lambda}(J^{\mathcal{C}} - \boldsymbol{d})_{+}. \tag{19}$$

We can tolerate actions that have a small cost but not enough to violate the constraint, and it is difficult for reward design that inevitably is strapped in weighting between sub-rewards. About solving the dual problems in Eq.(19), Stooke's work [Stooke *et al.*, 2020] shows that PID control update can stably improve gradient descent on Lagrange multipliers $\boldsymbol{\lambda}$ thus avoiding the negative impact of unstable training of $\boldsymbol{\lambda}$ on policy optimization. PID control updates $\boldsymbol{\lambda}$ as follows

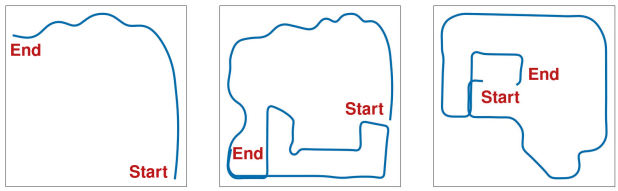
$$\begin{aligned}\boldsymbol{\lambda_{k+1}} \leftarrow &(\boldsymbol{K_p}(J^{\mathcal{C}} - \boldsymbol{d}) + (\boldsymbol{\lambda_k} + \boldsymbol{K_i}(J^{\mathcal{C}} - \boldsymbol{d}))_{+} + \\ &+ \boldsymbol{K_d}(J^{\mathcal{C}} - J^{\mathcal{C}}_{p})_{+})_{+},\end{aligned} \tag{20}$$

where $J^{\mathcal{C}}$ is the performance of average costs for the current iteration, $J^{\mathcal{C}}_{p}$ is that for the last iteration, $\boldsymbol{K_p}$ are coefficients of the proportional term, $\boldsymbol{K_i}$ are coefficients of the integral term, $\boldsymbol{K_d}$ are coefficients of the derivative term. And the above three variables have the same size as the number of constraints $|\mathcal{C}|$. Integral control is equivalent to gradient descent update, which remains necessary for eliminating steady-state violations at convergence. Proportional control hastens the response to constraint violations and dampens oscillations. Derivative control allows action to be taken in the event of a constraint violation, which prevents cost overshoot and limits the rate of cost increase in the feasible region. The derivative term is projected as $(\cdot)_{+}$ so that it acts against increases in cost but does not impede decreases.

Finally, please refer to Appendix B for the details of the MCPPO pseudo-code.

5 Simulation Results and Discussions

In this section, we first provide the environments, followed by the simulation results. Then, we discuss the robustness of the decision interval.



(a) Town07_V1 (b) Town07_V2 (c) Town03_V1

Figure 3: The three driving environments we designed, where (a) and (b) are in a village scenario, and (c) is in an urban scenario.

5.1 Setups

Environments. We design three autonomous driving control scenarios for empirical study shown in Figure 3. Town07_V1 is a mountain road with continuous curves on an uneven tarmac surface. Town07_V2 is based on the Town07_V1 expansion of the country roads with sharp turns, roundabouts, turns, etc., more challenging. Town03_V1 is an urban road that contains intersections, roundabouts, forks, turns, lane changes, etc. All environmental details are provided in Appendix A.

Benchmark Approaches. We compare with benchmark algorithms, PID control and mainstream DRL algorithms (mainstream DRLs) such as D3QN [Wang *et al.*, 2016], PPO [Engstrom *et al.*, 2020], SAC [Haarnoja *et al.*, 2018]. PID control is provided by CARLA [Dosovitskiy *et al.*, 2017]. mainstream DRLs are our implementation[2] based on ElegantRL [Liu *et al.*, 2021]. All DRLs' hyperparameters are provided in Appendix C.1.

Training and Evaluation Setting. We select Top5 policies in return performance during each DRL training, with each performing $M_{ep} = 8$ evaluation rollouts with some other seed every 4000 interaction steps. We consider the return and costs above thresholds to select the Top5 policies of MCPPO,

$$G_{CRL} = \sum_{t=0}^{H} R_t - \sum_{i=1}^{|\mathcal{C}|} \sum_{t=0}^{H} (C_t^i - d^i)_+. \tag{21}$$

The decision interval Δt is 0.2s during training, but Δt is 0.05s and H expanded 4 times during evaluation, which is for comparison with PID controllers.

Evaluation Metrics. In our evaluation, we focus the performance on real metrics of DRL policies in autonomous driving tasks, in addition to return performance. This is because the challenge of designing action-smoothing rewards results in rewards that fail to evaluate action smoothness. The evaluation of action smoothness includes evaluation of action magnitude M_s and comfort metric M_c. A smaller M_s and a larger M_c indicate better action smoothness.

- $M_s = \frac{1}{M_{ep}} \sum_{i=1}^{M_{ep}} \int_0^T \Delta steer$. $\Delta steer \in (0, 0.6]$ reflects the magnitude in the steering angle of the ego vehicle front wheels.

[2]https://github.com/GyChou/mcppoElegantRLforCarla.

Env	Algo	Action Smoothness		*Return*
		M_s	M_c	
Town07_ maintain- Road_V1	PID	0.0060 ± 0.0000	0.6542 ± 0.0000	597.55 ± 0.00
	D3QN	0.0867 ± 0.0368	0.6574 ± 0.0160	377.05 ± 167.47
	PPO	0.0115 ± 0.0007	0.7110 ± 0.0013	680.29 ± 1.01
	SAC	0.0072 ± 0.0004	0.6868 ± 0.0051	675.34 ± 2.28
	MCPPO	**0.0036 ± 0.0007**	**0.7530 ± 0.0054**	667.27 ± 1.76
Town07_ maintain- Road_V2	PID	0.0062 ± 0.0000	0.6550 ± 0.0000	3095.55 ± 0.00
	D3QN	-	-	-178.23 ± 125.12
	PPO	0.0150 ± 0.0007	0.6944 ± 0.0063	3478.01 ± 11.12
	SAC	0.0122 ± 0.0008	0.6746 ± 0.0018	3392.62 ± 7.84
	MCPPO	**0.0058 ± 0.0004**	**0.7539 ± 0.0038**	3394.03 ± 104.85
Town03_ urban- Road_V1	PID	0.0032 ± 0.0000	0.7670 ± 0.0000	3534.39 ± 0.00
	D3QN	0.0319 ± 0.0030	0.7107 ± 0.0121	3388.71 ± 69.92
	PPO	0.0109 ± 0.0005	0.7717 ± 0.0053	3489.13 ± 93.47
	SAC	0.0117 ± 0.0022	0.7552 ± 0.0198	3406.74 ± 65.59
	MCPPO	**0.0029 ± 0.0004**	**0.8251 ± 0.0407**	3151.83 ± 196.46

- The return of D3QN, too low in Town07_V2, indicates that the policies don't control the ego vehicle properly, so it's meaningless to evaluate the action smoothness about D3QN.

Table 1: Action smoothness and average return performance of PID, mainstream DRLs and our MCPPO in our three autonomous driving environments.

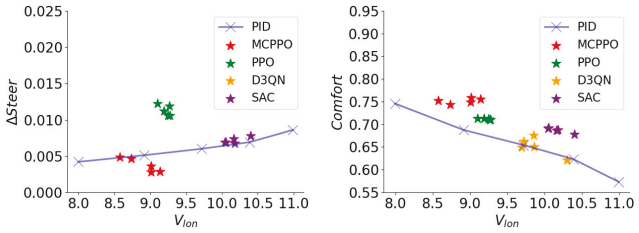
- $M_c = \frac{1}{M_{ep}} \sum_{i=1}^{M_{ep}} \int_0^T \frac{1}{1+acc_{la}+acc_{lo}+j_{la}+j_{lo}}$. $M_c \in (0, 1]$ overall reflects the lateral and longitudinal forces and force changes of the ego vehicle.

5.2 Results

Training Mainstream DRLs. For the above three environments (Town07_V1, Town07_V2 and Town03_V1), we train driving policies by D3QN, PPO, SAC and MCPPO. And Appendix C.2 provided the return performance of each DRL training process. Appendix C.3 provided the Lagrangian multipliers trends and cost curves of the MCPPO training process. In addition, the experiment in Appendix C.3 shows that reward weight adaption in Eq.(14) is valid.

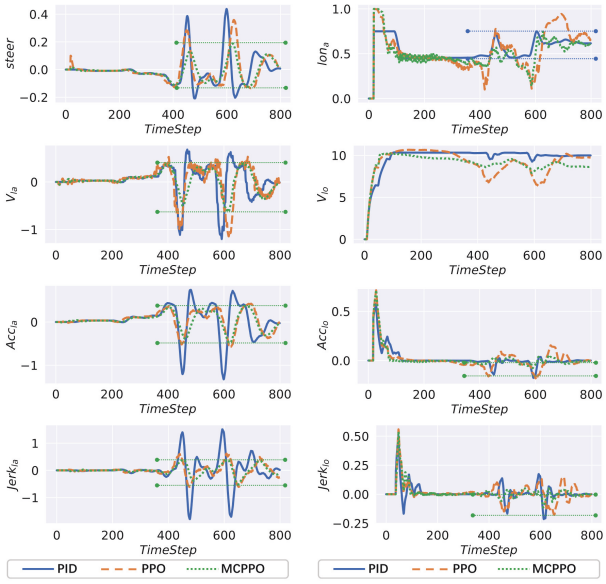
Comparing MCPPO with PID and Mainstream DRL algorithms. Top5 policies of mainstream DRLs are selected by return performance, and those of MCPPO are selected by Eq.(21). The target speed of the PID controller is set equal to the desired speed of DRL algorithms. Table 1 summarizes the performance of the Top5 policies of each DRL in three environments. We have three observations. (a) In terms of action magnitude, MCPPO is overall better than PID and mainstream DRLs. In details, MCPPO improves by 18.61% compared with PID on average of three environments, and improves by 57.71% compared with SAC, which has the best performance in mainstream DRLs. Mainstream DRL algorithms are overall poor than PID. (b) In terms of comfort, MCPPO is overall better than PID and mainstream DRLs, and mainstream DRLs are overall better than PID. Compared to PPO, having the best performance in mainstream DRLs, MCPPO improves 7.13% on average. (c) Under the condition that the return performance is close, MCPPO is significantly better than PID and mainstream DRLs in terms of action smoothness.

Figure 4 compares the action smoothness of DRL's policies with PID controllers at $[8, 11] m/s$ in environment Town07_V1. Only MCPPO policies have better action magnitude and comfort than PID controllers, while mainstream DRLs have better comfort but smaller action magnitude than



(a) Action magnitude (b) Comfort

Figure 4: Action smoothness comparison of our MCPPO and mainstream DRLs with PID controllers at different target speeds in Town07_V1. In (a), the performances of D3QN policies are $\Delta Steer > 0.025$.



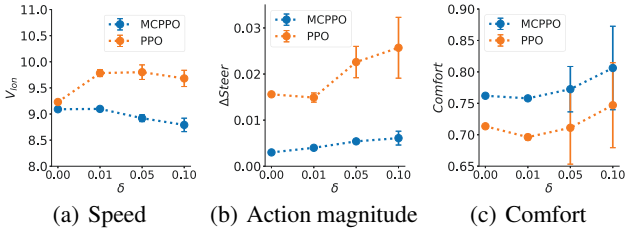
(a) Lateral metrics (b) Longitudinal metrics

Figure 5: Detailed performance in Town07_V1. Each horizontal line interval is the smallest amplitude in PID PPO and MCPPO.

PID. It contributes to demonstrate the result (c) obtained from Table1. It shows that our multi-constraint design is effective and that our proposed MCPPO can effectively solve the multi-constraint RL problem.

5.3 Comprehensibility

Performance on lateral metrics. Figure 5(a) shows that PID, PPO and MCPPO have similar curves in the four lateral metrics (i.e., action magnitude, velocity, acceleration and jerk), and the phase difference is due to the difference in longitudinal velocity. The constraints in Eq.(13) have a limited effect on amplitudes on lateral metrics. MCPPO has the smallest amplitudes on the four lateral metrics, and PPO has smaller amplitudes than PID in acceleration and jerk, but is closer to PID in speed. This further illustrates the importance of our constraint design about lateral metrics, which effectively enhances the smoothness of the lateral action.



(a) Speed (b) Action magnitude (c) Comfort

Figure 6: PPO and our MCPPO perform on longitudinal velocity, lateral action magnitude and comfort, which are evaluated in 10 times of the Town07_01 task under different disturbance intensity σ.

Performance on longitudinal metrics. Figure 5(b) shows that the curves of PID, PPO and MCPPO are in the four longitudinal metrics. There is no constraint of longitudinal metrics. MCPPO are better than PPO overall, of which the longitudinal speed changes are smoother, due to the lateral and longitudinal coupling. PID is the most smooth one in terms of speed and action magnitude, but has a little more volatility in acceleration and jerk. Note that MCPPO and PPO are faster to start compared with PID. This further indicates that our constraints for lateral control have some positive effects on longitudinal control in terms of action smoothness.

5.4 Robustness of Decision Interval

In this section, we discuss the advantages of DRLs in terms of the decision interval Δt. PID controllers are sensitive to Δt, and a slight disturbance in Δt can cause PID crash. But we find DRLs have excellent robustness in the decision interval Δt. We set the decision interval $\Delta t = 0.05 + noise$ and a disturbance $noise \sim \text{clip}(\mathcal{N}(0.05, \sigma), 0.05, 0.2)$ where σ is disturbance intensity, for evaluating the performance of PPO and MCPPO against different intensity of Δt disturbance. Figure 6 shows that as the disturbance increases, policies of PPO and MCPPO drive normally, with only a slight decrease in performance in terms of speed and action smoothness, which is much more robust compared with the collapse of PID.

6 Conclusions and Future Work

In this paper, we have proposed a novel way to incorporate the smoothness of actions in the reward of DRL. Specifically, we introduced sub-rewards and added multiple constraints related to these sub-rewards in DRL to improve the smoothness of continuous motions. For solving the multi-constraint RL problems, we presented multi-constraint proximal policy optimization (MCPPO). Extensive simulation results in a wide range of autonomous driving tasks showed that our proposed MCPPO achieves substantial action smoothness improvement with little sacrifice in task performance, which is of great importance for real-world scenarios. We also found that DRLs are very robust against decision interval disturbance. Future work is in progress to apply and develop our approach to practical applications such as real-world self-driving cars, and to investigate the extension for off-policy algorithms.

Acknowledgments

We thank the support of National Natural Science Foundation of China Nos. 62002238 and 61836005, and the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) under Grant No. GML-KF-22-26.

References

[Achiam *et al.*, 2017] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *ICML*, 2017.

[Altman, 1999] Eitan Altman. *Constrained Markov decision processes*. CRC Press, 1999.

[Amodei *et al.*, 2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[Badia *et al.*, 2020] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *ICLR*, 2020.

[Bellman, 2015] Richard E. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press, 2015.

[Boyd and Vandenberghe, 2014] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2014.

[Chen *et al.*, 2021] Chen Chen, Hongyao Tang, Jianye Hao, Wulong Liu, and Zhaopeng Meng. Addressing action oscillations through learning policy inertia. In *AAAI*, 2021.

[Chow *et al.*, 2017] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *In J. Mach. Learn. Res.*, 18:167:1–167:51, 2017.

[Dosovitskiy *et al.*, 2017] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017.

[Engstrom *et al.*, 2020] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on PPO and TRPO. *CoRR*, 2020.

[Girshick, 2015] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

[Lillicrap *et al.*, 2016] 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. In *ICLR*, 2016.

[Liu *et al.*, 2021] Xiao-Yang Liu, Zechu Li, Zhuoran Yang, Jiahao Zheng, Zhaoran Wang, Anwar Walid, Jian Guo, and Michael I. Jordan. Elegantrl-podracer: Scalable and elastic library for cloud-native deep reinforcement learning. *CoRR*, abs/2112.05923, 2021.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Neunert *et al.*, 2019] Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Jost Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin A. Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *CoRL*, 2019.

[Paternain *et al.*, 2019] Santiago Paternain, Luiz F. O. Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *NeurIPS*, 2019.

[Schulman *et al.*, 2016] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Siekmann *et al.*, 2021] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan W. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. In *RSS*, 2021.

[Song *et al.*, 2020] Wenjie Song, Shixian Liu, Yujun Li, Yi Yang, and Changle Xiang. Smooth actor-critic algorithm for end-to-end autonomous driving. In *ACC*, 2020.

[Stooke *et al.*, 2020] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID Lagrangian methods. In *ICML*, 2020.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

[Tessler *et al.*, 2019] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *ICLR*, 2019.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.

[Yang *et al.*, 2020] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Projection-based constrained policy optimization. In *ICLR*, 2020.