Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

DOMA: Deep Smooth Trajectory Generation Learning for Real-Time UAV Motion Planning

Jin Yu[1*], **Haiyin Piao**[2*†], **Yaqing Hou**[3*], **Li Mo**[4*], **Xin Yang**[3*], **Deyun Zhou**[2*]

[1] SADRI Institute
[2] Northwestern Polytechnical University
[3] Dalian University of Technology
[4] Beijing Institute of Technology
haiyinpiao@mail.nwpu.edu.cn

Abstract

In this paper, we present a Deep Reinforcement Learning (DRL) based real-time smooth UAV motion planning method for solving catastrophic flight trajectory oscillation issues. By formalizing the original problem as a linear mixture of dual-objective optimization, a novel Deep smOoth Motion plAnning (DOMA) algorithm is proposed, which adopts an alternative layer-by-layer gradient descending optimization approach with the major gradient and the DOMA gradient applied separately. Afterwards, the mix weight coefficient between the two objectives is also optimized adaptively. Experimental result reveals that the proposed DOMA algorithm outperforms baseline DRL-based UAV motion planning algorithms in terms of both learning efficiency and flight motion smoothness. Furthermore, the UAV safety issue induced by trajectory oscillation is also addressed.

Introduction

Recently, the development of Deep Reinforcement Learning (DRL) methods have attracted increasing attention in solving highly maneuverable autonomous UAV motion planning problems. Compared with other traditional planning solvers, i.e., in (González-Sieira et al. 2020; Kaur, Chatterjee, and Likhachev 2021; González et al. 2015; Nägeli et al. 2017; Liu et al. 2017), DRL converts the labor extensive onboard planning workloads into the offline interactive sampling in environment. Moreover, Deep Neural Network (DNN) is introduced to model fitting pipeline, and continuously optimizes the planning accuracy (Tong et al. 2021; Gutierrez and Leonetti 2021; Wang et al. 2020; Faust et al. 2016). Once a DNN-based motion planning model is learned, the onboard planning is simply required to execute computational affordable DNN forward inference, which brings superior real-time planning performance.

Nevertheless, one of the emerging challenges is to address the flight trajectory oscillation problem during motion planning of UAVs. Specifically, the trajectory oscillation issue induced by excessively switching aircraft steering actions is also the main cause of pilot-induced oscillation in manned aircraft (Andrievsky et al. 2019). DRL methods naturally require high-frequency trial and error interaction during environment exploration. This exploration mechanism brings additional difficulties in improving the stabilization and smoothness which is focused on by this paper.

This paper presents a step in this direction. Specifically, our particular interest is placed on improving vanilla DRL to generate smooth flight trajectories during the motion planning skills learning and execution process. There are three novel contributions as follows:

- The auxiliary motion smoothness optimization objective is defined as a Multi-Step Smoothness Metric (MSSM). On this basis, we formalize the practical smooth UAV motion planning as a linear mixture of dual-objective optimization problem.
- A novel Deep smOoth Motion plAnning (DOMA) method is proposed. The overall smooth flight motion planning agent is thus obtained by adopting an alternative layer-by-layer optimization approach by processing the major gradient and the DOMA gradient separately with the mix weight coefficient adaptively updated.
- Experimental result reveals that the proposed DOMA algorithm performs better than baseline DRL-based UAV motion planning algorithms in terms of both learning efficiency and flight trajectory smoothness. Since the aircraft steering action excessively switching phenomenon is also properly suppressed, the overall flight safety is enhanced.

Preliminaries

A Markov Decision Process (MDP, denoted as $\mathcal{M}$) is adopted for the original UAV motion planning problem formulation defined by the tuple $\langle \boldsymbol{S}, \boldsymbol{A}, r, \mathcal{T}, \rho_0, \gamma, T \rangle$ (Sutton and Barto 2018). Where $\boldsymbol{S}$ is the set of states, $\boldsymbol{A}$ is the set of actions, $r : \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \mapsto \mathbb{R}$ is the bounded reward function, $\mathcal{T} : \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \mapsto [0, 1]$ is the transition probability distribution, where $\mathcal{T}(\boldsymbol{s}, \boldsymbol{a})$ is the deterministic transi-

[*] These authors contributed equally.
[†] Corresponding author.

tioning of $\mathcal{T}(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ from state $\boldsymbol{s}$ to $\boldsymbol{s}'$ assuming action $\boldsymbol{a}$ was taken, $\rho_0 : \boldsymbol{S} \mapsto [0,1]$ is the initial state distribution, γ is the discount factor that we assume $\gamma \in [0,1]$ and T is the episode horizon. The objective of a reinforcement learning agent is to find a policy π that maximize the expected cumulative discounted reward: $\mathcal{J}(\pi)_{MDP} = \max \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi}\left[\sum_{k=0}^{T}\gamma^k r\left(\boldsymbol{s}_{t+k}, \boldsymbol{a}_{t+k}\right)\right]$ where ρ_π is the state-action marginals of the trajectory distribution induced by policy π.

In this study, an open access aerodynamic model and thrust data from (Stevens, Lewis, and Johnson 2015; Nguyen 1979) is adopted in the vector form in Eq. (1)-(2).

$$\left(\frac{d}{dt}\right)_B \left(\boldsymbol{V}_K^{CG}\right)_B = \left(\boldsymbol{F}_{tot}^{CG}\right)_B - \left(\boldsymbol{\omega}_K^{IB}\right)_B \times \left(\boldsymbol{V}_K^{CG}\right)_B \quad (1)$$

$$\begin{aligned}\left(\frac{d}{dt}\right)_B \left(\boldsymbol{\omega}_K^{IB}\right)_B = \left(\boldsymbol{I}_{BB}^{CG}\right)^{-1} \big[\left(\boldsymbol{M}_{tot}^{CG}\right)_B \\ - \left(\boldsymbol{\omega}_K^{IB}\right)_B \times \boldsymbol{I}_{BB}^{CG}\left(\boldsymbol{\omega}_K^{IB}\right)_B\big]\end{aligned} \quad (2)$$

Where $\boldsymbol{V}_K^{CG}$ is the kinematic velocity of drone center of gravity, $\boldsymbol{F}_{tot}^{CG}$ is the total forces acting on the drone, $\boldsymbol{\omega}_K^{IB}$ is the kinematic angular velocity, $\boldsymbol{I}_{BB}^{CG}$ is the moment of inertia about drone center of gravity denoted in body axes, $\boldsymbol{M}_{tot}^{CG}$ is the total moments acting on the drone.

The UAV nonlinear mathematical model (Wang and Stengel 2004) in the state space form $F(\boldsymbol{s}, \boldsymbol{u})$ can be described as Eq. (3).

$$\frac{d}{dt}\boldsymbol{s} = F(\boldsymbol{s}, \boldsymbol{u}) = f(\boldsymbol{s}) + g(\boldsymbol{s})\boldsymbol{u} \quad (3)$$

Where $\boldsymbol{s} = [\,\alpha\ \beta\ V_T\ T\ n_n\ p\ q\ r\ \phi\ \theta\ \psi\ x\ y\ z\,]^T$ is the UAV state vector, α is the angle of attack, β is the side-slip angle, V_T is the true airspeed, T is the engine thrust, n_n is the normal overload, $[p, q, r]$ is the attitude angular rate, $[\,\phi,\ \theta,\ \psi\,]$ is the attitude in form of euler angles , $[x, y, z]$ is the global position of UAV, and $\boldsymbol{u} = [\,\delta_a\ \delta_e\ \delta_r\ \delta_T\,]^T$ is the airplane control input vector, which is consist of aileron deflection, elevator deflection, rudder deflection and throttle control command, f and g are nonlinear state and control distribution functions, respectively.

Nonlinear Dynamic Inversion (NDI) controller is used for stabilizing UAV configurations that would otherwise be aerodynamically unstable (Snell, Enns, and Garrard Jr 1992). Mapping the basic actuator commands into higher level control commands that would be easy for DRL agent to learn with. Denote $\tilde{\boldsymbol{s}}_c = [\,T_c\ n_{nc}\ p_c\,]^T \in \mathbb{R}^3$ as DRL actions $\boldsymbol{a}$, the NDI control law can be found from Eq. (4) under the assumption that $g(\tilde{\boldsymbol{s}})$ is invertible for all values of $\tilde{\boldsymbol{s}}$:

$$\boldsymbol{u} = g^{-1}(\tilde{\boldsymbol{s}})\left(\frac{d}{dt}\tilde{\boldsymbol{s}} - f(\tilde{\boldsymbol{s}})\right) = g^{-1}(\tilde{\boldsymbol{s}})\left(\tilde{\boldsymbol{\omega}}(\boldsymbol{a} - \tilde{\boldsymbol{s}}) - f(\tilde{\boldsymbol{s}})\right) \quad (4)$$

Where T_c, n_{nc}, and p_c are thrust, normal load factor, and rolling rate steering commands, respectively. $\tilde{\boldsymbol{\omega}}$ is bandwidth frequency set as high as they can be without exciting structural modes or being subject to the bandwidth limitations of the control actuators.

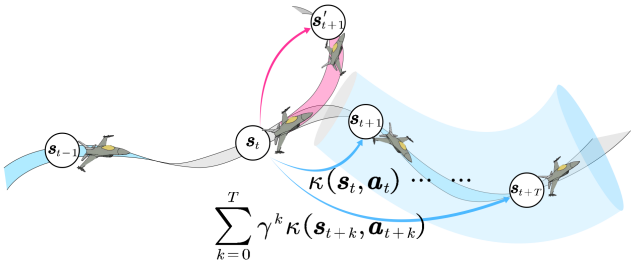

Figure 1: Multi-Step Smoothness Metric.

Differing from autonomous vehicles, the trajectory oscillation is mainly induced by sustained flight attitude variation (Xu et al. 2018; Wang, Chan, and de La Fortelle 2018; Zhu et al. 2020). The quaternion form of aircraft attitude angular rate $\left\|\left(\frac{d}{dt}\right)_E(\boldsymbol{q})_E\right\|$ is adopted as the key smoothness metric.We define One-Step Smoothness Metric (OSSM) $\kappa(\boldsymbol{s}, \boldsymbol{a})$ as below:

$$\begin{aligned}\kappa(\boldsymbol{s}, \boldsymbol{a}) &= \left\|\left(\frac{d}{dt}\right)_E(\boldsymbol{q})_E\right\| \\ &= \left\|\frac{1}{2}\boldsymbol{\omega}_K^{IB}(\boldsymbol{s}, \boldsymbol{a})\left(\boldsymbol{q}(\boldsymbol{s}, \boldsymbol{a})\right)_E\right\| \le \epsilon\end{aligned} \quad (5)$$

Where ϵ is the upper bound of OSSM, $\kappa(\boldsymbol{s}, \boldsymbol{a})$ indicates how smooth the one-step UAV attitude angles change. OSSM only represents the oscillation impact currently, which isn't enough to describe the smoothness of the overall episodic policy. The metric of multi-step flight trajectory smoothness is defined as a recursive form of MSSM.

As shown in Fig. 1, the pink flight trajectory from $\boldsymbol{s}_t$ to $\boldsymbol{s}'_{t+1}$ which represents a excessive attitude change rate oscillation within a given duration T. On the contrary, the cumulative attitude change rate of the blue flight trajectory from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+T}$ is always within the light blue oscillation feasible envelope, which achieves better smoothness. Notably, given an OSSM limit ϵ and time horizon T by applying summation formula of proportional sequence, assume that $\kappa(\boldsymbol{s}, \boldsymbol{a}) < \epsilon$, $0 < \gamma < 1, \forall T \ge 0$, then the supremum of MSSM which is denoted as $\mathcal{K}_t$ for the given policy π exists:

$$\sup_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi} \mathbb{E}\left[\sum_{k=0}^{T}\gamma^k\kappa\left(\boldsymbol{s}_{t+k}, \boldsymbol{a}_{t+k}\right)\right] := \frac{\epsilon(1-\gamma^T)}{1-\gamma} \quad (6)$$

DOMA

The overall proposed DOMA network architecture is shown in Fig. 2. In general, the whole neural network can be divided into two parts: the major network and the DOMA network, which carries motion planning and trajectory smoothing functionality respectively. In which major network is consist of actor network $\pi_\theta(\boldsymbol{s})$ and critic network $Q_\phi(\boldsymbol{s}, \boldsymbol{a})$, the learnable parameters are θ and ϕ accordingly. Similarly, the DOMA network $\varkappa_\psi(\boldsymbol{s}, \boldsymbol{a})$ holds ψ as its learnable parameter.

The corresponding smooth motion planning problem is formalized by a linear mixture of dual-objective optimiza-

tion problem, which can be formally defined as:

$$\begin{aligned}\min \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi}\hat{\lambda}^*\left[-\varkappa_\psi(\boldsymbol{s},\boldsymbol{a})+\mathcal{K}_t\right]\\ +\max \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi}Q_\phi(\boldsymbol{s},\boldsymbol{a})\end{aligned} \tag{7}$$

DOMA then solves the above problem by using an alternating gradient descending optimization as described in Eq. (8). To begin with, the first level MDP accumulative return optimizing gradient namely major gradient is calculated, the parameter update process is consistent with DDPG (Lillicrap et al. 2015). The policy parameter θ is then optimized via major gradient till convergence, while parameter ϕ of $Q_\phi(\boldsymbol{s},\boldsymbol{a})$ is then temporarily frozen. Successively, the second-level MSSM constraint exceeding minimization gradient namely DOMA gradient is calculated on this basis, policy parameter θ is further optimized, parameter ψ of $\varkappa_\psi(\boldsymbol{s},\boldsymbol{a})$ is also frozen till it's converged. Consequently, hyperparameter $\hat{\lambda}^*$ is optimized via a gradient descending process.

$$\begin{aligned}\hat{\lambda}^* = \arg\min_{\hat{\lambda}>0}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi}\hat{\lambda}\left[-\varkappa_\psi(\boldsymbol{s},\pi_\theta(\boldsymbol{s}))+\mathcal{K}_t\right]\\ \min_\theta \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi}\hat{\lambda}^*\left[-\varkappa_\psi(\boldsymbol{s},\pi_\theta(\boldsymbol{s}))+\mathcal{K}_t\right]\\ \max_\theta \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi}Q_\phi(\boldsymbol{s},\pi_\theta(\boldsymbol{s}))\end{aligned} \tag{8}$$

In order to calculate the proposed DOMA gradient at the t-th iteration, we update the $\varkappa_\psi(\boldsymbol{s},\boldsymbol{a})$ by minimizing the associated mean squared Bellman error of transitions $\left\{\left(\boldsymbol{s}^i,\boldsymbol{a}^i,\kappa^i,\boldsymbol{s}'^{,i},d^i\right)\right\}_{i\in B}$ sampled from an independent replay buffer which stores the OSSM metric namely DOMA Replay Buffer (See Fig. 2 (c) for detail), in which κ^i is the OSSM metric for the i-th time step. Let $\varkappa'_\psi(\boldsymbol{s},\boldsymbol{a})$ be the target network of DOMA, we have the training label z_t^i in form of:

$$z_t^i = \mathbb{E}_{\boldsymbol{s}'\sim p(\cdot|\boldsymbol{s},\boldsymbol{a})}\left[\kappa(\boldsymbol{s},\boldsymbol{a})+\gamma\varkappa'_\psi\left(\boldsymbol{s}',\pi_\theta\left(\boldsymbol{s}'\right)\right)\right] \tag{9}$$

Afterward, the MSSM fitting DNN ψ_{t+1} can be updated by the loss function listed below. Updating process are listed as described in Eq. (10).

$$\psi_{t+1} = \arg\min_\psi \sum_{i\in B}\left(z_t^i-\varkappa_\psi\left(\boldsymbol{s}_t^i,\boldsymbol{a}_t^i\right)\right)^2 \tag{10}$$

Furthermore, the DOMA gradient calculation for regularizing θ of UAV flight trajectory can be treated as second-level gradient descending optimization and the DOMA gradient is calculated with minimizing the MSSM constraint exceeding times function $\varkappa_\psi(\boldsymbol{s},\boldsymbol{a})$ by utilizing the chain rule listed below:

$$\theta_{t+1} = \theta_t + \underbrace{\hat{\lambda}_t^*\frac{\eta}{|B|}\sum_{i\in B}\nabla_{\boldsymbol{a}}\varkappa_{\psi_{t+1}}(\boldsymbol{s},\boldsymbol{a})\nabla_\theta\pi_{\theta_t}\left(\boldsymbol{s}_i\right)}_{DOMA\ Gradient} \tag{11}$$

We then need to adjust the approximate mix weight coefficient $\hat{\lambda}$. Instead of requiring the user to set the $\hat{\lambda}^*$ manually, we can automate this process by formulating a gradient descending reinforcement learning objective below:

$$\hat{\lambda}^* = \arg\min_{\hat{\lambda}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi^*}-\hat{\lambda}\left[\varkappa_\psi(\boldsymbol{s},\boldsymbol{a})-\mathcal{K}_t\right] \tag{12}$$

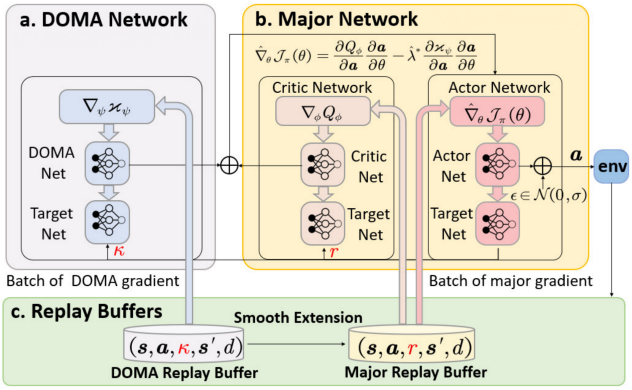

Figure 2: DOMA Neural Network Architecture.

Moreover, the approximate mix weight coefficient $\hat{\lambda}^*$ updating gradient can be calculated against parameter $\hat{\lambda}$ of the above reinforcement learning objective, and the new $\hat{\lambda}_{t+1}$ can be updated by Eq. (13).

$$\begin{aligned}\hat{\lambda}_{t+1} =& \hat{\lambda}_t - \frac{\partial\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho_\pi^*}-\hat{\lambda}\left[\varkappa_\psi(\boldsymbol{s},\boldsymbol{a})-\mathcal{K}_t\right]}{\partial\hat{\lambda}}\\ =& \hat{\lambda}_t - \frac{\eta_{\hat{\lambda}}}{|B|}\sum_{i\in B}\left(\mathcal{K}_t-\varkappa_\psi(\boldsymbol{s},\boldsymbol{a})\right)\end{aligned} \tag{13}$$

Experiments

For the sake of concentrating on evaluating the performance of DOMA, a specific sequential target zone approaching simulation is established. The scenario adopts the NED (North-Earth-Down) global coordinate system, with the space size of $15km\times15km\times10km$ respectively. A successful UAV motion planning activity should pass through all spherical target zones scattered among the environment. The simulation interval is 0.3 seconds and the horizon for each episode is 90 seconds. The reward settings are described as follows. For the dense reward, we have a) $r = -|D_T|/5000$, where D_T is the altitude difference between UAV and target; b) $r = -|A_T|/pi$, where A_T is the angle between the speed direction and the line of sight; c) $r = V_T/100$, where V_T is approach velocity. Moreover, for the option reward, we have a) $r = 200*n$, when getting the nth target; b) $r = -500$, when UAV falling to the ground; c) $r = -500$, when flight envelope limit is exceeded.

We first test the learning performance of the proposed DOMA method and compare it with baseline DRL-based motion planning methods, which include A2C, DDPG, and SAC.

It can be directly observed from Fig. 3 that the DOMA algorithm proposed has stronger learning ability and can master UAV motion planning skills with less training samples compared with other baseline algorithms. Generally speaking, the learning efficiency of SAC algorithm is significantly higher than that of DDPG algorithm (Haarnoja et al. 2018), but DOMA algorithm, as a variant of DDPG with an auxiliary smooth regularizer, shows stronger learning efficiency in experiments.

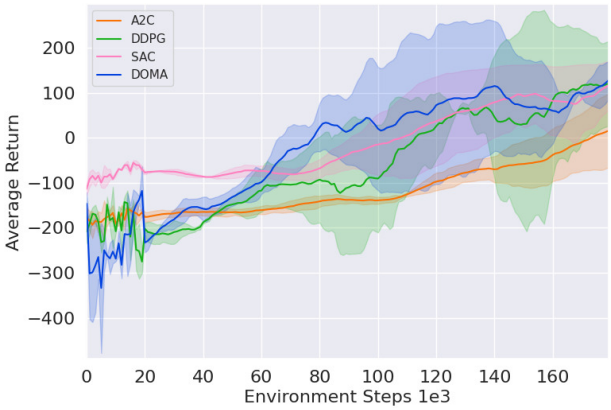

Figure 3: Learning curves comparison with baseline algorithms.

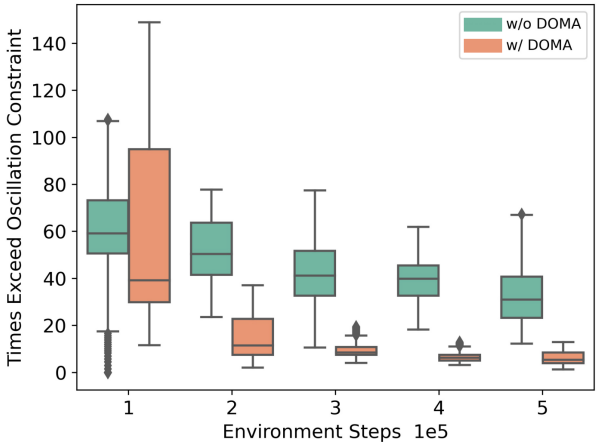

Figure 4: Times of MSSM constraint $\mathcal{K}_t$ violation.

Fig. 4 shows the times of MSSM constraint violation of the two comparison algorithms with the training process. As training processes, the times of violation for the original method decreases linearly, but still shows a certain extent of oscillation. When training ended, after 5×10^5 samples were collected, the average violation times of the original method is 32. In contrast, the average violation times in the comparative experimental group with DOMA is reduced to less than 20 times only after 2×10^5 samples collected. Furthermore, the overall violation times decays rapidly to about 10 times exponentially, and the corresponding variance is also significantly reduced.

Compared with DOMA, the baseline algorithm shows significant oscillation when flying over complex environments(See Fig. 5). In terms of $|\dot{\boldsymbol{q}}|$ time series, the peak value of baseline algorithm even reach 0.57 with more than 50 steps of MSSM constraint violations occur (See Fig. 6), while DOMA is suppressed within 0.2 with none constraint violation occur (See Fig. 7). The agent does not even adjust its attitude excessively and violently in the whole process,

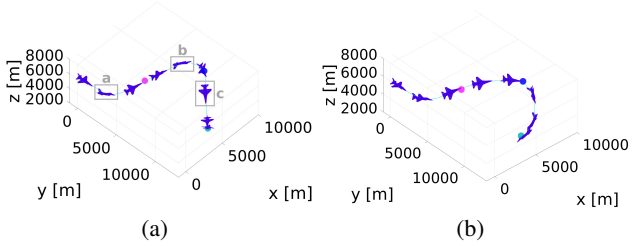

Figure 5: Trajectory of complex multi-target scenario. (a) Trajectoy w/o DOMA. (b) Trajectoy w/ DOMA.

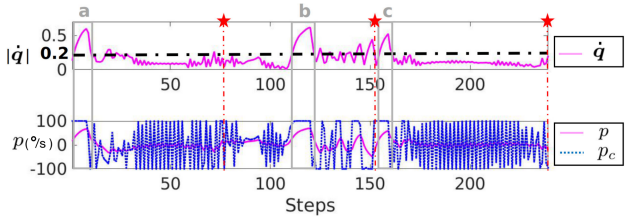

Figure 6: Ket state time series w/o DOMA.

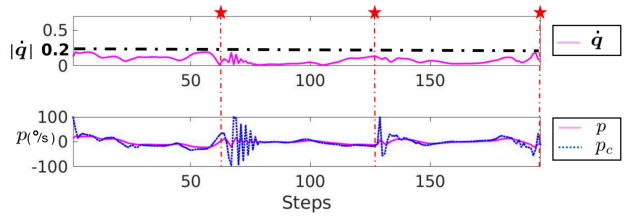

Figure 7: Ket state time series w/ DOMA.

which proves that DOMA achieves better trajectory stability even in complex multi-target crossing task. Therefore, we have reason to believe that DOMA will refrain from trajectory divergence induced by oscillation.

Conclusion

In this paper, we have described approaches for ensuring DRL-based UAV trajectory smoothness. A novel Deep smOoth Motion plAnning (DOMA) algorithm is proposed for addressing this issue. Firstly, the original problem is converted into a linear mixture of dual-objective optimization form. Secondly, the trajectory smoothness constraint is further modeled as an auxiliary objective beyond the chief optimization goal with an additional DOMA gradient calculated, which significantly suppresses the trajectory oscillation by utilizing an alternative layer-by-layer gradient descending optimization approach. To a certain extent, DOMA improves flight safety by inhibiting trajectory oscillation comprehensively. Joint optimization is another way to improve flight trajectory smoothness and related research will be carrying out in future investigations. Also, meaningful extensions of this work may attempt to enrich the application domain into self-driving cars, unmanned ships, and any other intelligent vehicles.

References

Andrievsky, B.; Arseniev, D. G.; Kuznetsov, N. V.; and Zaitceva, I. S. 2019. *Pilot-induced oscillations and their prevention*. Springer International Publishing.

Faust, A.; Chiang, H.-T.; Rackley, N.; and Tapia, L. 2016. Avoiding moving obstacles with stochastic hybrid dynamics using pearl: Preference appraisal reinforcement learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 484–490. IEEE.

González, D.; Pérez, J.; Milanés, V.; and Nashashibi, F. 2015. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4): 1135–1145.

González-Sieira, A.; Cores, D.; Mucientes, M.; and Bugarín, A. 2020. Autonomous navigation for UAVs managing motion and sensing uncertainty. *Robotics and Autonomous Systems*, 126: 103455.

Gutierrez, R. L.; and Leonetti, M. 2021. Meta Reinforcement Learning for Heuristic Planing. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 551–559.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Kaur, J.; Chatterjee, I.; and Likhachev, M. 2021. Speeding Up Search-Based Motion Planning using Expansion Delay Heuristics. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 528–532.

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

Liu, S.; Atanasov, N.; Mohta, K.; and Kumar, V. 2017. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2872–2879. IEEE.

Nägeli, T.; Meier, L.; Domahidi, A.; Alonso-Mora, J.; and Hilliges, O. 2017. Real-time planning for automated multi-view drone cinematography. In *ACM Transactions on Graphics*, volume 36, 1–10. ACM New York, NY, USA.

Nguyen, L. T. 1979. Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability. *NASA Technical Paper*, 12854.

Snell, S. A.; Enns, D. F.; and Garrard Jr, W. L. 1992. Nonlinear inversion flight control for a supermaneuverable aircraft. *Journal of guidance, control, and dynamics*, 15(4): 976–984.

Stevens, B. L.; Lewis, F. L.; and Johnson, E. N. 2015. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tong, G.; Jiang, N.; Biyue, L.; Xi, Z.; Ya, W.; and Wenbo, D. 2021. UAV navigation in high dynamic environments: A deep reinforcement learning approach. *Chinese Journal of Aeronautics*, 34(2): 479–489.

Wang, D.; Fan, T.; Han, T.; and Pan, J. 2020. A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing. *IEEE Robotics and Automation Letters*, 5(2): 3098–3105.

Wang, P.; Chan, C.-Y.; and de La Fortelle, A. 2018. A reinforcement learning based approach for automated lane change maneuvers. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, 1379–1384. IEEE.

Wang, Q.; and Stengel, R. F. 2004. Robust nonlinear flight control of a high-performance aircraft. *IEEE Transactions on Control Systems Technology*, 13(1): 15–26.

Xu, X.; Zuo, L.; Li, X.; Qian, L.; Ren, J.; and Sun, Z. 2018. A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(10): 3884–3897.

Zhu, M.; Wang, Y.; Pu, Z.; Hu, J.; Wang, X.; and Ke, R. 2020. Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. *Transportation Research Part C: Emerging Technologies*, 117: 102662.