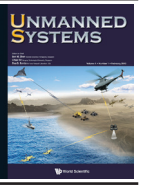
Unmanned Systems, Vol. 14, No. 2 (2026) 391–400
© World Scientific Publishing Company
DOI: 10.1142/S230138502650007X

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us



Time-Optimal Flight in Cluttered Environments via Safe Reinforcement Learning

Wei Xiao [*], Zhaohan Feng [*], Ziyu Zhou [*], Jian Sun [*], Gang Wang [*,‡], Jie Chen [*,†]

[*]*School of Automation, Beijing Institute of Technology, Beijing 100081, China*

[†]*Department of Control Science and Engineering, Tongji University, Shanghai 201804, China*

This paper addresses the problem of guiding a quadrotor through a predefined sequence of waypoints in cluttered environments, aiming to minimize the flight time while avoiding collisions. Previous approaches either suffer from prolonged computational time caused by solving complex non-convex optimization problems or are limited by the inherent smoothness of polynomial trajectory representations, thereby restricting the flexibility of movement. In this work, we present a safe reinforcement learning approach for autonomous drone racing with time-optimal flight in cluttered environments. The reinforcement learning policy, trained using safety and terminal rewards specifically designed to enforce near time-optimal and collision-free flight, outperforms current state-of-the-art algorithms. Additionally, experimental results demonstrate the efficacy of the proposed approach in achieving both minimum flight time and obstacle avoidance objectives in complex environments, with a commendable 66.7% success rate in unseen, challenging settings.

Keywords: Reinforcement learning; time-optimal flight; obstacle avoidance.

US

1. Introduction

In recent years, drones have gained increasing popularity across various domains due to their exceptional performance, sparking a research fervor in autonomous drone racing. For instance, the AlphaPilot Challenge [1, 2], the autonomous drone racing at IROS and NeurIPS [3, 4]. Autonomous drone racing is an excellent platform for challenging perception, trajectory planning and control technologies. The rules stipulate that the drones must systematically navigate through a sequence of gates, ensuring collision-free traversal while preserving the aggressive capabilities of the drones throughout the course.

In the context of racing in cluttered environments, planning a collision-free trajectory with minimum-time traversal across a series of waypoints poses a formidable challenge. Achieving the objective of minimum time necessitates pushing the quadrotor to its physical extremes in terms of speed and acceleration. Furthermore, the existence of obstacles in three-dimensional space introduces highly non-convex optimization problems, making conventional methodologies impractical. The traditional framework often segregates planning and control, thereby increasing the risk of crashes due to inconsistencies in models. This poses a substantial challenge to the agility of planners and the resilience of controllers.

Previous works either do not utilize the complete dynamics of the quadrotor [5], resulting in suboptimal and even dynamically infeasible solutions, or represent trajectories using polynomials or splines, which limit the positivity of control inputs [6], reducing the aggressiveness

Received 18 October 2024; Accepted 14 January 2025; Published 22 March 2025. This paper was recommended for publication in its revised form by editorial board member, Jinqiang Cui.
Email Address: ‡*gangwang@bit.edu.cn*
‡Corresponding author.

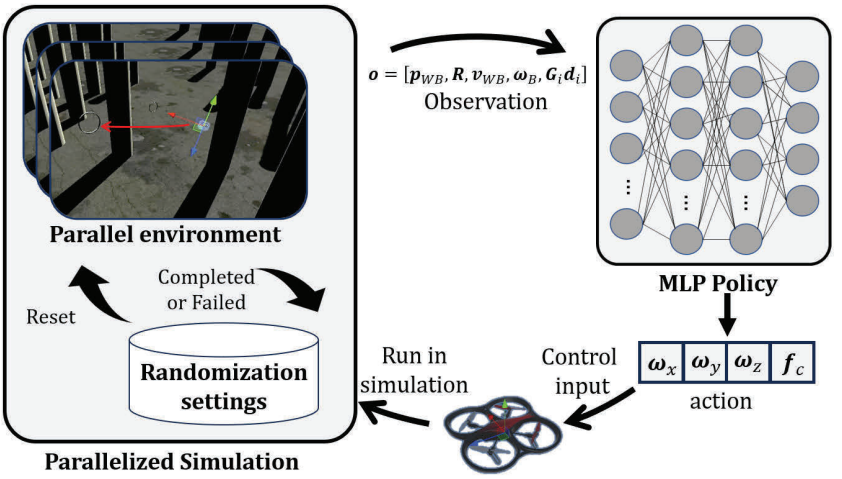


Fig. 1. The proposed SRL framework.

of the trajectories. While optimization-based techniques might provide optimal solutions [7], they typically demand significant computational resources and time, making them less suitable for agile flight, particularly in environments rife with obstacles where computational demands are further compounded by non-convexity. Additionally, search-based methods [8] encounter model mismatch problems due to the separation of planning and control.

Given the successful applications of deep reinforcement learning (DRL), we develop a DRL-based approach for autonomous drone racing in cluttered environments. DRL stands out as a promising alternative to traditional controller designs, harnessing the trial-and-error process to automatically optimize control strategies. Its capacity to accommodate nonlinear dynamics and non-convex optimization objectives renders it well-suited for our application. Previous studies in autonomous drone racing have shown that neural network controllers trained with reinforcement learning (RL) outperform optimal control methods [9].

This work contributes a safe reinforcement learning (SRL) based method for racing quadrotors in cluttered environments, as shown in Fig. 1. In pursuit of minimizing flight time and ensuring obstacle avoidance in cluttered environments, a safety reward is proposed to encourage the quadrotor to maintain a safe distance from obstacles. Further, we introduce a terminal reward, activated upon task completion, to save flight time even further. The proposed safety and terminal rewards are designed to facilitate performance that either matches or surpasses the capabilities of the state-of-the-art method presented in [9] for achieving minimum-time flight within cluttered environments. Extensive experimental analyses are undertaken, encompassing comparisons of flight time performance, evaluation of generalization capabilities to navigate in unseen environments, and ablation studies aimed at validating individual items of the proposed design. Our experimental findings demonstrate that

- Our method significantly improves the policy's obstacle avoidance capability in cluttered environments by introducing a safety reward enforcing obstacle avoidance.
- By assigning terminal rewards (e.g. the objective of minimizing the total time) upon task completion, the negative impact by the safety reward can be mitigated. Experimental validation illustrates that this approach can achieve near-optimal agile and collision-free flights in highly complex environments.
- The SRL policy, trained with randomized environments of varying complexity levels, demonstrates robust generalization performance, achieving a commendable success rate of 66.7% even in unseen, highly complex environments with dense obstacle configurations.

2. Related Work

In traditional navigation results, methods for obstacle avoidance often decouple trajectory planning and control. Given the minimum-time and collision-free trajectory, accurate trajectory tracking by the controller is instrumental for the quadrotor to navigate through the environment safely. These methods can be classified into polynomial and spline trajectory, search-based methods, and optimization-based methods. Their effectiveness depends on the quality of the planned trajectory and the resilience of the controller. Polynomial [10] and B-spline [11] representations are widely used due to their ability to leverage the differential

flatness property [12], simplifying the planning process by reducing the dimensionality of the problem. However, while these methods excel at producing smooth trajectories, they often fail to achieve optimal performance for minimum-time flight [6].

Search-based methods translate trajectory planning into graph search problems, with the goal of optimizing discretized time intervals. However, these methods [5, 8] face the curse of dimensionality, especially in high-dimensional spaces. Furthermore, they rely on point-mass models for trajectory planning, which can compromise trajectory quality. These methods have primarily been applied to path planning between two points, which contrasts with our multi-waypoint obstacle avoidance problem.

Optimization-based methods treat the time-optimal trajectory as a constrained optimization problem and solved it through nonlinear programming. These methods [7, 13, 14] provide the optimal state and input control sequence at each step while satisfying dynamic constraints. However, this approach often faces challenges in time allocation. The work by [15] introduces a complementary progress constraints (CPC) approach to address this issue, while also considering true actuator saturation. Additionally, the work proposes a novel approach to computing time-optimal trajectories [16] by fixing nodes with waypoint constraints and adopting separate sampling intervals for trajectories between waypoints. This approach significantly reduces planning time compared to CPC but is not suitable for online scenarios with long trajectories. Moreover, it does not consider environments with obstacles.

In recent years, learning-based methods have garnered extensive attention and achieved significant advancements in agile flight. These methods [17, 18] offer promising solutions to various challenges by training neural network policies to directly map high-dimensional observations to control commands. For instance, researchers have utilized imitation learning to train neural network policies for agile flight [19, 20], achieving high-speed trajectory tracking using learned policies, and integrating topological path planning methods with DRL for obstacle avoidance. Previous work [9] has highlighted the superior performance of neural network controllers trained with RL over optimal control methods. This superiority stems from learning-based methods directly optimizing policies at the task level. Nonetheless, the existing work primarily focuses on obstacle-free environments and does not address scenarios with obstacles. The work [21] combines DRL with topological path planning methods, leveraging the advantages of deep learning to overcome model mismatch issues. In contrast, our approach does not require explicit trajectory representation but directly outputs control commands from observation states end-to-end.

3. Methodology

3.1. *Quadrotor dynamics*

We model the quadrotor, actuated by four motors, as a 6 degree-of-freedom rigid body of mass m with a (diagonal) inertial matrix $\boldsymbol{J}$. We refer to the quadrotor dynamics model presented in [22] given as follows:

$$\begin{aligned}
\dot{\boldsymbol{p}}_{WB} &= \boldsymbol{v}_{WB},\\
\dot{\boldsymbol{v}}_{WB} &= \boldsymbol{q}_{WB} \odot \boldsymbol{c} - \boldsymbol{g} - \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v},\\
\dot{\boldsymbol{q}}_{WB} &= \frac{1}{2}\boldsymbol{\Lambda}(\omega_B)\cdot\dot{\boldsymbol{q}}_{WB},\\
\dot{\omega}_B &= \boldsymbol{J}^{-1}(\boldsymbol{\eta} - \omega_B \times \boldsymbol{J}\omega_B),
\end{aligned} \tag{1}$$

where $\odot$ denotes the quaternion multiplication, $\boldsymbol{p}_{WB}$ and $\boldsymbol{v}_{WB}$ represent the position and linear velocity of the quadrotor in the reference frame WB. The quaternion $\boldsymbol{q}_{WB}$ represents the attitude of the quadrotor, ω_B is the body rate in the body frame B, $\boldsymbol{R}$ is the rotation matrix, and $\boldsymbol{D} = \mathrm{diag}(d_x, d_y, d_z)$ is a constant diagonal matrix collecting the rotor drag coefficients, which is a linear effect in velocity. Moreover, $\boldsymbol{\Lambda}(\omega_B)$ is a skew-symmetric matrix, $\boldsymbol{c} = [0, 0, c]$ is the mass-normalized thrust vector and $\boldsymbol{g} = [0, 0, -g_z]^T$ is the gravity vector with $g_z = 9.81\,\mathrm{m/s^2}$. The conversion of the four rotor thrusts $[f_1, f_2, f_3, f_4]$ to the mass-normalized thrust c and the body torques $\boldsymbol{\eta}$ is given by

$$\boldsymbol{\eta} = \begin{bmatrix} \frac{l}{\sqrt{2}}(f_1 - f_2 - f_3 + f_4)\\ \frac{l}{\sqrt{2}}(-f_1 - f_2 + f_3 + f_4)\\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix},$$
$$c = \frac{1}{m}(f_1 + f_2 + f_3 + f_4), \tag{2}$$

where l is the arm length of the quadrotor, m is its mass, and κ is the torque constant. The thrust dynamics of each individual rotor is approximated as a first-order system $\dot{f}_i = (f_{d,i} - f_i)/k_m$, where $f_{d,i}$ is the desired thrust of rotor i and k_m is the time-delay constant. We employ a fourth-order Runge–Kutta method for integrating the dynamic equations.

3.2. *Reinforcement learning for minimum-time flight without collisions*

We formulate the autonomous drone racing problem in obstacle-rich environments within the RL framework. In this context, we model the drone racing task as a Markov decision process (MDP) defined by the 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma)$ [23], where $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state transition function which defines

the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, the reward function r determines that the quadrotor will receive an immediate reward $r(s,a)$ upon executing action a in state s. The quadrotor starts from a state $s_0 \in \mathcal{S}$ drawn from the initial state distribution ρ_0. At each step k, an action $a(k)$ is randomly sampled from a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ parameterized by a deep neural network (DNN) with weight parameters collected in θ.

Our goal is to find the optimal policy π_θ^*, namely, the weight parameters θ of the DNN policy π_θ, by maximizing the expected accumulative discounted reward as follows:

$$\pi_\theta^* = \arg\max_{\pi_\theta} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(k)\right]. \tag{3}$$

3.2.1. *Reward function*

Training RL policies directly with the objectives of minimizing total flight time and penalizing collisions will appear intuitive. However, total flight time and collision penalties represent sparse rewards, which make the learning difficult and often leads to suboptimal solutions. Furthermore, as the environment complexity escalates, training becomes increasingly challenging. A feasible strategy to address this challenge involves employing a proxy reward that closely approximates the true performance objective while offering feedback to the quadrotor at each step.

Rewards computed based on the path progress serve as an alternative to the minimum-time flight reward. The gate progress reward, as utilized in [9], provides a straightforward and efficient optimization objective to replace minimum-time flight while circumventing sparse rewards. Our approach extends the gate progress reward and introduces additional ones to enhance the efficacy of minimum-time flight in cluttered environments. As an ingredient of our reward, the gate progress reward $r_p(k)$ facilitates the quadrotor to find the shortest time path, given by

$$r_p(k) = \lambda_1(||g(k) - p(k-1)|| - ||g(k) - p(k)||) - \lambda_2||\omega(k)||, \tag{4}$$

where $g(k)$ and $p(k)$ represent the target point (e.g. the target gate center) and the position of the quadrotor at step k, respectively. The term $||g(k) - p(k-1)|| - ||g(k) - p(k)||$ guides the quadrotor toward the center of the current target gate; once it reaches the gate, its next gate in racing is set as the target point. Moreover, $\lambda_2||\omega(k)||$ denotes a penalty on the body rate, where λ_2 is a small coefficient.

Additionally, we introduce a safety reward to prevent collisions in cluttered environments, by progressively keeping the quadrotor away from obstacles within an observation range using exponential functions, defined as follows:

$$r_s(k) = \lambda_3 \sum_{i}^{N} e^{-||d_i(k)||}, \tag{5}$$

where λ_3 is a coefficient that adjusts the aggressiveness of the trajectory, N is designated as the nearest N obstacle within the quadrotor's observation range, and $||d_i(k)||$ denotes the distance between the quadrotor and the ith obstacle.

Therefore, the overall reward at each step k is given by

$$r(k) = r_p(k) + r_s(k) + \begin{cases} r_c & \text{if collision,} \\ \lambda_4 T & \text{if completed,} \\ 0 & \text{else,} \end{cases} \tag{6}$$

where r_c and $\lambda_4 T$ are the terminal rewards depending on whether collision occurs or the racing task is successfully completed. In our experiments, a substantial negative reward of $r_c = -30$ was applied in the event of a collision. Task completion is deemed unsuccessful if the quadrotor either collides or fails to reach the designated target point. The variable T signifies the total time required by the quadrotor to complete the task. Additionally, the reward $\lambda_4 T$ is introduced to penalize trajectories with excessively prolonged completion times.

3.2.2. *Policy architecture*

The policy network consists of two layers, which take observations $\boldsymbol{o}$ as input and directly map them to the quadrotor's action $\boldsymbol{a}$. The observation space of the DNN policy comprises three components: the state of the quadrotor, the positions of the next few target points, and the distances between the quadrotor and N obstacles. Following [24], we utilize rotation matrices to represent the attitude of the quadrotor, thereby avoiding ambiguities and discontinuities in gesture expression. The quadrotor's observation is defined as $\boldsymbol{o}(k) = [\boldsymbol{p}_{WB}(k), \boldsymbol{R}(k), \boldsymbol{v}_{WB}(k), \omega_B(k), \boldsymbol{G}_j(k), ||d_i(k)||]$, where $\boldsymbol{p}_{WB}(k), \boldsymbol{R}(k), \boldsymbol{v}_{WB}(k)$, and $\omega_B(k)$ are the quadrotor's position, rotation matrix, linear velocity, and body rate, and $\boldsymbol{G}_j(k)$ collects the positions of the next j target points with the current target point included (we set $j = 2$ in our experiment). The hyper-parameter j can take any value equal to or less than the total number of target points. Additionally, $||d_i(k)||$, where $i \in [1, \ldots, N]$, represents the Euclidean distance between the quadrotor and the ith obstacle.

The action is defined as $\boldsymbol{a}(k) = [f^{\text{total}}(k), \omega(k)]$, which consists of the desired collective thrust $f^{\text{total}}(k)$ and the body rate $\omega(k)$. Subsequently, a low-level controller is utilized to execute the action commands. Notably, this approach has exhibited robust sim-to-real transferability in learning-based control policies [25].

3.3. *Policy training*

Improving the training efficiency and generalization of DRL method remains a formidable challenge in robotics research. Previous studies have suggested solutions, including parallel training, segmented training [21], and domain randomization [19]. To bolster the generalization performance, we introduce a set of modifications to the training environment, as elaborated below.

3.3.1. *Parallel environment simulation*

To train the policy in simulation, we utilize the open-source Flightmare simulator [22], which can simulate several hundred quadrotors with a flexible physics engine. Specifically, 100 environments are designed for training RL policies. Each environment allows for different initialization settings and variations, significantly enhancing data diversity and saving time for data collection.

3.3.2. *Representation of collision*

Creating environments in the physics engine based on Flightmare involves establishing interaction rules between the environment and the drone, such as collision detection. In our training, obstacles, such as spherical, cylindrical, and cubical obstacles, are simplified as single or multiple spherical obstacles with a radius of r. A safety distance $d_{\text{safe}} = r + l + \epsilon_{\text{safe}}$ is defined to assess potential collisions, with ϵ_{safe} denoting a safety threshold. If the distance between the quadrotor and any obstacle falls below the safety distance, i.e. $d_{i,k} < d_{\text{safe}}$, or if the quadrotor surpasses the world boundaries, it is signaled as a collision, prompting a reset of the quadrotor's position.

3.3.3. *Domain randomization*

Domain randomization has emerged as an effective strategy for mitigating environmental and model uncertainties. To enhance the generalization performance and robustness of the RL planner, we incorporate domain randomization into our methodology. Throughout the training phase, the initial states of the quadrotor, the positions of waypoints, and the complexity of the environment (i.e. clutter) are randomly generated. The quadrotor parameters, such as linear drag coefficient and thrust mapping coefficient, are randomly sampled around calibrated values at each reset to ensure the policy's robustness against unknown and potentially stochastic aerodynamic effects.

4. Experiments

In this section, a lot of tests were conducted to compare the proposed SRL policy with state-of-the-art methods and to evaluate its robustness and generalization capabilities in uncertain and unseen experimental scenarios. Additionally, a series of ablation studies were performed to validate the design choices of the proposed method.

4.1. *Experiment setup*

The proximal policy optimization (PPO) algorithm [26] stands out for its robust performance and easy tuning process, making it a favorable choice among RL researchers. In our study, we leveraged the PPO algorithm to train the SRL policy that generates the quadrotor's action. PPO relied on Stable-Baseline3 [27], a trusted tool for RL implementations. Table 1 lists the PPO hyperparameters.

Several cluttered environments were designed based on Flightmare, allowing for the concurrent simulation of hundreds of drones and environments. This approach significantly bolstered the efficiency of data collection.

The scenarios, Split-S with obstacles and Forest with varying density levels, were designed to evaluate the performance of our approach. In both scenarios, the mission requires the quadrotor to traverse a sequence of designated waypoints from the starting point to the terminal while avoiding the obstacles. The distribution of waypoints on the Split-S track was generated similar to [9], while the positions of waypoints in the Forest environment were given in Table 2. In contrast to the Split-S track, the Forest environment presented a longer track with more obstacles. Two experiments were conducted to respectively evaluate the performance of flight time and the generalization capabilities of the policy using these environments, along with several ablation studies to validate our design. At the beginning of each training, the state of the quadrotor or the elements in the environment were randomly initialized. And

Table 1. PPO hyperparameters.

Parameter	Value
learning rate	0.0004
policy network	MLP[512, 512] + ReLu
value network	MLP[512, 512] + ReLu
GAE-λ	0.95
discount factor	0.95
entropy coeffieient	0.005
batch size	25000
clip range	0.2

Table 2. Positions of waypoints.

Waypoint number	Position
1	[−20.0, 0.0, 1.12]
2	[−15.0, −5.0, 1.12]
3	[−10.0, −10.0, 1.12]
4	[−5.0, −5.0, 1.12]
5	[0.0, −10.0, 3.53]
6	[5.0, −5.0, 1.12]
7	[10.0, 0.0, 3.53]
8	[15.0, 5.0, 3.53]
9	[20.0, 0.0, 1.12]

all tests collected 1000 trajectories for evaluation. Two metrics, average time and crash ratio, were used to evaluate the policy's performance. For Split-S, average time represents the mean lap time, while for Forest, it represented the mean time required to reach the endpoint. Unless noted otherwise, all experiments were conducted using a quadrotor configuration with diagonal inertia matrix $\boldsymbol{J} = [0.0025, 0.0021, 0.0043]$, rotor-drag coefficients $d_x = 0.26$, $d_y = 0.28$, $d_z = 0.42$, arm length $l = 0.15$ and torque constant $\kappa = 0.022$. Additionally, the parameters in the reward function were set as follows: $\lambda_1 = 1, \lambda_2 = 0.01, \lambda_3 = -0.05$, and $\lambda_4 = -1$.

4.2. *Minimum-time flight*

To benchmark the performance of our method in capability of flight time, a comparative analysis between our approach and the recently proposed method [9] was conducted. Since the RL method in [9] is not open-sourced, we reproduce the algorithm based on the information provided in [9] and achieve consistent results with those reported. In this experiment, we only randomized the initial state of the drone and one of waypoints at the beginning of each flight, including its position, linear velocity, and orientation. Specifically, the starting point of the quadrotor $[p_{WB,x}, p_{WB,y}, p_{WB,z}]$ was determined, and it was varied within the range of -1 to 1 m. Similarly, the linear velocity and orientation were also randomly initialized between $[-1, 1]$. It is worth noting that for the Forest environment, we did not randomize the obstacle density during training, and the environments used for testing were the same as those during training.

The results, as shown in Table 3, indicate promising performance for our method. Additionally, Fig. 2 illustrates

Table 3. Comparison of baseline and our RL method.

Method	Environment	Time (s)	Crash ratio (%)
RL in method in [9]	Split-S with obstacles	7.11	0
	Forest	7.34	90.20
Ours	Split-S with obstacles	**6.49**	0
	Forest	**7.31**	0

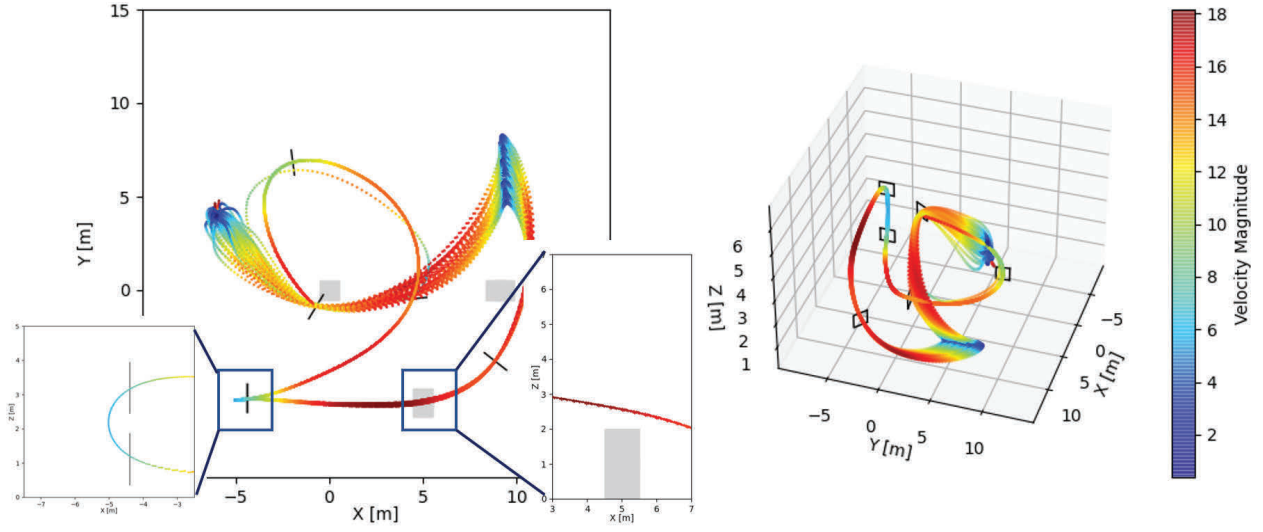


Fig. 2. The trajectories generated by our method in Split-S with obstacles are depicted in the figure. The light gray area represents obstacles. During the flight, the quadrotor needs to pass through 7 waypoints, with one of them being randomly initialized in position for each flight.

the 100% success rate achieved for both obstacle avoidance and the completion of tasks involving multiple waypoints in Split-S with obstacles. Our method demonstrates performance that either exceeds or closely approaches that of state-of-the-art methods in both environments. The Forest environment presents greater challenges due to its increased number of obstacles and longer track length. Despite these complexities, our method achieves performance closely aligned with state-of-the-art benchmark, while ensuring collision-free navigation throughout the entire trajectory. The improvement in performance can be attributed to the terminal reward $\lambda_4 T$ incorporated into our reward function, as demonstrated in Sec. 4.4.

4.3. *Performance in unseen environments*

To evaluate the robustness and generalization of the proposed SRL policy, in addition to randomly initializing the quadrotor's state as described in 4.2, we also changed the complexity of the environment. Specifically, 100 environments with varying complexity, representing different densities of obstacles, were created to train the policy. During testing, we randomly generated three unseen environments and categorized them into three levels based on the density of obstacles: Level 1 (obstacle spacing approximately $5m$), Level 2 (obstacle spacing approximately $3-5m$), and Level 3 (obstacle spacing approximately $1-3m$).

Table 4. Evaluation results in unseen environments.

Forest environment	Average time (s)	Crash ratio (%)
Level 1	7.31	0
Level 2	7.33	0
Level 3	7.46	33.33

The outcomes are depicted in Table 4. For each level, we executed tests on $1,000$ trajectories. Figure 3 showcases the performance of our method in Level 1. The trained neural network achieves a success rate of 66.7% in the most challenging environment, which is not seen during training.

4.4. *Ablation studies*

We conducted ablation experiments to validate our method, focusing primarily on the design of our reward function in both Forest and Split-S with obstacles. In these experiments, there were no changes made to the elements in the environment during training and testing, and the environments used for testing were the same as those during training. Only the initial state of the drone was randomized during both training and testing phases.

4.4.1. *Safety reward*

A safety reward, mentioned in Sec. 3.2, was designed to encourage the drone to avoid obstacles. A smaller value of $\lambda_3 < 0$ corresponded to more assertive trajectories. This experiment aimed to validate the impact of the safety reward on the policy. The result is shown in Table 5. In the Forest environment (obstacle spacing approximately $1-3m$), "Average Time" signifies the completion time, whereas in the Split-S environment, it represents the lap time. The Forest environment presents a higher degree of complexity compared to Split-S. Consequently, in the

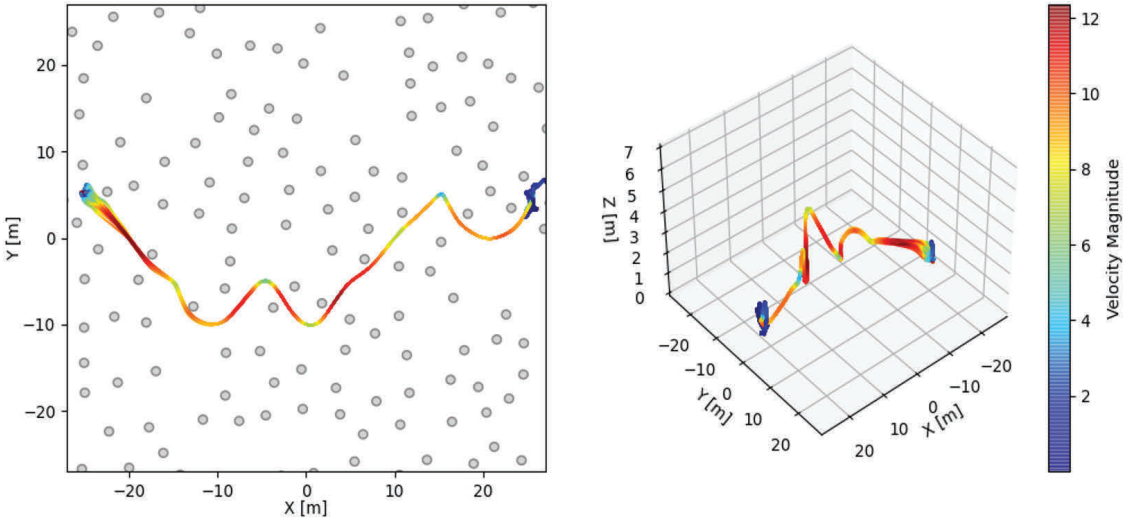


Fig. 3. The trajectories generated by our approach in a level 1 forest environment navigate through nine waypoints while effectively avoiding obstacles. The light gray area in the figure represents the obstacles.

Table 5. Ablation studies testing the efficacy of the safety reward.

Environment	Safety reward	Crash ratio (%)
Forest	✓	0
	×	60.95%
Split-S with obstacles	✓	0
	×	0

Table 6. Ablation studies for testing the efficacy of the terminal reward.

Environment	$\lambda_4 T$	Average time (s)
Forest	✓	7.31
	×	8.57
Split-S with obstacles	✓	6.49
	×	7.12

absence of the safety reward, the trained policy carries a risk of collisions. This underscores the beneficial influence of the safety reward introduced by our approach on obstacle avoidance in intricate environments.

4.4.2. *Terminal reward*

In the early stages of training, the quadrotor triggered the terminal reward r_c in (6) due to flying out of the world boundaries or colliding with obstacles because of its randomly initialized policy. We conducted experiments to verify this and found that, without r_c, the policy struggles to converge. This reward was assigned a substantial negative value to prevent the quadrotor from revisiting similar actions. In this experiment, our focus was on studying the impact of the reward $\lambda_4 T$ on the SRL policy. The results of the experiment are presented in Table 6. When eliminating the term $\lambda_4 T$, the policy exhibits subpar performance in terms of task completion time. The experimental results demonstrate that our design has a positive impact on minimum-time flight.

5. Conclusions

In this paper, we presented a novel learning-based methodology aimed at training an SRL policy to guide a quadrotor through cluttered environments, ensuring minimal flight time and collision-free navigation. To this end, novel safety and terminal rewards as well as training procedures are designed. Our experimental results showcase the robustness and generalization capabilities of the proposed approach. Additionally, ablation analyses underscore the positive contributions of each component within our framework. These findings indicate that SRL shows significant potential for high-speed flight and obstacle avoidance.

Acknowledgement

The work was supported in part by the National Natural Science Foundation of China under Grants U23B2059, 62173034, 61925303, and 62088101.

ORCID

Wei Xiao https://orcid.org/0009-0003-2623-2555
Zhaohan Feng https://orcid.org/0009-0008-9055-9276
Ziyu Zhou https://orcid.org/0000-0001-9485-7336
Jian Sun https://orcid.org/0000-0001-9898-3129
Gang Wang https://orcid.org/0000-0002-7266-2412
Jie Chen https://orcid.org/0000-0003-2449-9793

References

[1] W. Guerra, E. Tal, V. Murali, G. Ryou and S. Karaman, FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* (IEEE, 2019), pp. 6941–6948.
[2] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar and D. Scaramuzza, Alphapilot: Autonomous drone racing, *Auton. Robot.* **46**(1) (2022) 307–320.
[3] H. Moon *et al.*, Challenges and implemented technologies used in autonomous drone racing, *Intell. Serv. Robot.* **12** (2019) 137–148.
[4] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager and A. Kapoor, AirSim drone racing lab, in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, Vol. 123 (PMLR, 2020), pp. 177–191.
[5] A. Martínez Novo, L. Lu and P. Campoy, Fast rrt* 3d-sliced planner for autonomous exploration using mavs, *Unmanned Syst.* **10**(02) (2022) 175–186.
[6] M. W. Mueller, M. Hehn and R. D'Andrea, A computationally efficient motion primitive for quadrocopter trajectory generation, *IEEE Trans. Robot.* **31**(6) (2015) 1294–1310.
[7] H. Pham and Q.-C. Pham, A new approach to time-optimal path parameterization based on reachability analysis, *IEEE Trans. Robot.* **34**(3) (2018) 645–659.
[8] S. J. Fusic and R. Sitharthan, Improved rrt* algorithm-based path planning for unmanned aerial vehicle in a 3d metropolitan environment, *Unmanned Syst.* **12**(05) (2024) 859–875.
[9] Y. Song, A. Romero, M. Müller, V. Koltun and D. Scaramuzza, Reaching the limit in autonomous racing: Optimal control versus reinforcement learning, *Sci. Robot.* **8**(82) (2023) eadg1462.
[10] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu and F. Gao, Fast-racing: An open-source strong baseline for SE(3) planning in autonomous drone racing, *IEEE Robot. Autom. Lett.* **6**(4) (2021) 8631–8638.
[11] B. Penin, P. R. Giordano and F. Chaumette, Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions, *IEEE Robot. Autom. Lett.* **3**(4) (2018) 3725–3732.

[12] J.-C. Ryu and S. K. Agrawal, Differential flatness-based robust control of mobile robots in the presence of slip, *Int. J. Robot. Res.* **30**(4) (2011) 463–475.
[13] Z. Feng, J. Chen, W. Xiao, J. Sun, B. Xin and G. Wang, Learning hybrid policies for mpc with application to drone flight in unknown dynamic environments, *Unmanned Syst.* **12**(02) (2024) 429–441.
[14] C. Qin, M. S. Michet, J. Chen and H. H.-T. Liu, Time-optimal gate-traversing planner for autonomous drone racing, preprint (2023), arXiv:2309.06837.
[15] P. Foehn, A. Romero and D. Scaramuzza, Time-optimal planning for quadrotor waypoint flight, *Sci. Robot.* **6**(56) (2021) eabh1221.
[16] Z. Zhou, G. Wang, J. Sun, J. Wang and J. Chen, Efficient and robust time-optimal trajectory planning and control for agile quadrotor flight, *IEEE Robot. Autom. Lett.* **8**(12) (2023) 7913–7920.
[17] A. A. Cabrera-Ponce, L. O. Rojas-Perez, J. A. Carrasco-Ochoa, J. F. Martinez-Trinidad and J. Martinez-Carranza, Gate detection for micro aerial vehicles using a single shot detector, *IEEE Latin Am. Trans.* **17** (12) (2019) 2045–2052.
[18] F. Sadeghi and S. Levine, CAD^{2}RL: Real single-image flight without a single real image, in *Proc. Robot. Sci. Syst.*, Cambridge, Massachusetts, USA, July 12–16, 2017.
[19] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun and D. Scaramuzza, Learning high-speed flight in the wild, *Sci.Robot.* **6**(59) (2021) eabg5810.
[20] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun and D. Scaramuzza, Deep drone acrobatics, in *Proc. Robot. Sci. Syst.*, Corvalis, Oregon, USA, July 12–16, 2020.
[21] R. Penicka, Y. Song, E. Kaufmann and D. Scaramuzza, Learning minimum-time flight in cluttered environments, *IEEE Robot. Autom. Lett.* **7**(3) (2022) 7209–7216.
[22] Y. Song, S. Naji, E. Kaufmann, A. Loquercio and D. Scaramuzza, Flightmare: A flexible quadrotor simulator, *Proc. Conf. Robot Learn., PMLR*, London, UK, Nov. 8–11, 2021, pp. 1147–1157.
[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 2018).
[24] J. Hwangbo, I. Sa, R. Siegwart and M. Hutter, Control of a quadrotor with reinforcement learning, *IEEE Robot. Autom. Lett.* **2**(4) (2017) 2096–2103.
[25] E. Kaufmann, L. Bauersfeld and D. Scaramuzza, A benchmark comparison of learned control policies for agile quadrotor flight, in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, (IEEE, 2022), pp. 10504–10510.
[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, Proximal policy optimization algorithms, arXiv:1707.06347 (2017).
[27] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann, Stable-baselines3: Reliable reinforcement learning implementations, *Journal of Machine Learning Research* **22**(268) (2021) 1–8.

Wei Xiao received his B.Eng. degree from the Chongqing University of Technology, Chongqing, China, in 2019, and his M.Eng. degree from the Shenzhen University, Shenzhen, Guangdong, China, in 2022. He is currently pursuing his doctoral degree in the School of Automation at the Beijing Institute of Technology. His research works focus on learning-to-control, reinforcement learning, and their applications to drone flight.

Zhaohan Feng obtained his B.Eng. degree from the Department of Mechanical Engineering, Shantou University, Shantou, China, in 2018, the M.Eng. degree from the Beijing University of Posts and Telecommunications, in 2021. Currently, he is working toward his Ph.D. degree in the School of Automation at the Beijing Institute of Technology. His research interests include optimal control, unmanned aerial vehicles and reinforcement learning.

ZiYu Zhou received his B.Eng. degree from the School of Automation at the Beijing Institute of Technology, Beijing, China, in 2020, and the M.Eng. degree from the School of Automation at the Beijing Institute of Technology, in 2023. His research interests include optimal control, unmanned aerial vehicles and their applications to drone flight.

Jian Sun received his B.Sc. degree from the Department of Automation and Electric Engineering, Jilin Institute of Technology, Changchun, China, in 2001, the M.Sc. degree from the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences (CAS), Changchun, China, in 2004, and the Ph.D. degree from the Institute of Automation, CAS, Beijing, China, in 2007. He was a Research Fellow with the Faculty of Advanced Technology, University of Glamorgan, Pontypridd, U.K., from 2008 to 2009. He was a Post-Doctoral Research Fellow with the Beijing Institute of Technology, Beijing, from 2007 to 2010. In 2010, he joined the School of Automation, Beijing Institute of Technology, where he has been a Professor since 2013. His current research interests include networked control systems, time-delay systems, and security of cyber-physical systems.

Dr. Sun is an Editorial Board Member of the IEEE Transactions on Systems, Man and Cybernetics: Systems, the Journal of Systems Science & Complexity, Science China: Information Sciences, and Acta Automatica Sinica.

Gang Wang received B.Eng. degree in Automatic Control in 2011, and a Ph.D. degree in Control Science and Engineering in 2018, both from the Beijing Institute of Technology, Beijing, China. He also received a Ph.D. degree in Electrical and Computer Engineering from the University of Minnesota, Minneapolis, USA, in 2018, where he stayed as a postdoctoral researcher until July 2020. Since August 2020, he has been a professor with the School of Automation at the Beijing Institute of Technology. His research interests focus on the areas of signal processing, control, and reinforcement learning with applications to autonomous systems. He was the recipient of the Best Paper Award from the Frontiers of Information Technology & Electronic Engineering in 2021, the Excellent Doctoral Dissertation Award from the Chinese Association of Automation in 2019, the Best Student Paper Award from the 2017 European Signal Processing Conference, and the Best Conference Paper at the 2019 IEEE Power & Energy Society General Meeting. He is currently on the editorial board of Signal Processing, IEEE Transactions on Signal and Information Processing over Networks, and IEEE Open Journal of Control Systems.

Jie Chen received his B.Sc., M.Sc., and the Ph.D. degrees in Control Theory and Control Engineering from the Beijing Institute of Technology, Beijing, China, in 1986, 1996, and 2001, respectively. He was the President of Tongji University, Shanghai, China, during 2018-2023. He is a Professor with the Control Science and Engineering, Beijing Institute of Technology and Tongji University, where he serves as the Director of the National Key Laboratory of Autonomous Intelligent Unmanned Systems (KAIUS).

His research interests include complex systems, multiagent systems, multiobjective optimization and decision, and constrained nonlinear control. Prof. Chen is currently the Editor-in-Chief of Unmanned Systems and the Journal of Systems Science and Complexity. He has served on the editorial boards of several journals, including the IEEE Transactions on Cybernetics, International Journal of Robust and Nonlinear Control, and Science China Information Sciences. He is a Fellow of IEEE, IFAC, and a member of the Chinese Academy of Engineering.