Residual Policy Learning Facilitates Efficient Model-Free Autonomous Racing

Ruiqi Zhang, Jing Hou, Guang Chen, Zhijun Li, *Fellow, IEEE*, Jianxiao Chen, and Alois Knoll, *Senior Member, IEEE*

***Abstract*—Motion planning for autonomous racing is a challenging task due to the safety requirement while driving aggressively. Most previous solutions utilize the prior information or depend on complex dynamics modeling. Classical model-free reinforcement learning methods are based on random sampling, which severely increases the training consumption and undermines the exploration efficiency. In this letter, we propose an efficient residual policy learning method for high-speed autonomous racing named ResRace, which leverages only the real-time raw observation of LiDAR and IMU for low-latency obstacle avoiding and navigation. We first design a controller based on the modified artificial potential field (MAPF) to generate a policy for navigation. Besides, we utilize the deep reinforcement learning (DRL) algorithm to generate a residual policy as a supplement to obtain the optimal policy. Concurrently, the MAPF policy effectively guides the exploration and increases the update efficiency. This complementary property contributes to the fast convergence and few required resources of our method. We also provide extensive experiments to illustrate our method outperforms the leading algorithms and reaches the comparable level of professional human players on the five F1Tenth tracks.**

***Index Terms*—Autonomous vehicle navigation, motion and path planning, reinforcement learning.**

Manuscript received 5 March 2022; accepted 8 July 2022. Date of publication 20 July 2022; date of current version 14 September 2022. This letter was recommended for publication by Associate Editor H. Zhang and Editor A. Banerjee upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grants 61906138, 62133013, and U2013601, in part by the European Union's Horizon 2020 Framework Program for Research and Innovation through Human Brain Project SGA3 under Grant Agreement 945539, in part by the Shanghai Municipal Science and Technology Major Project of ZJ Lab and Shanghai Center for Brain Science and Brain-Inspired Technology under Grant 2018SHZDZX01, in part by Anhui Provincial Natural Science Foundation, in part by the Major Science and Technology Projects of Anhui Province under Grant 202103a05020004, and in part by Shanghai Rising Star Program under Grant 21QC1400900. *(Corresponding author: Guang Chen.)*

Ruiqi Zhang, Jing Hou, and Jianxiao Chen are with the School of Automotive Studies, Tongji University, Shanghai 201804, China (e-mail: 1854136@tongji.edu.cn; 1911061@tongji.edu.cn; jxchen@tongji.edu.cn).

Guang Chen is with the School of Automotive Studies, Tongji University, Shanghai 201804, China, and also with the Department of Informatics, Technical University of Munich, 80333 Munich, Germany (e-mail: guang@in.tum.de).

Alois Knoll is with the Department of Informatics, Technical University of Munich, 80333 Munich, Germany (e-mail: knoll@in.tum.de).

Zhijun Li is with the Wearable Robotics and Autonomous Systems Lab, University of Science and Technology of China, Hefei 230022, China (e-mail: zjli@ieee.org).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3192770, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3192770

I. Introduction

Autonomous racing is a promising issue and has obtained much attention. The objective of racing players is to complete the laps as fast as possible. The players are required to generate precise actions and aggressively drive at the dynamics limitation of vehicles. To solve this problem, classical approaches decouple autonomous racing into trajectory planning and controller optimization [1], [2]. These approaches are widely studied and show impressive results with optimization-based techniques and model predictive control. However, their performances are highly related to the selection of parameters, and the reference trajectory requires prior information like fine dynamics model, global maps and known routes. Meanwhile, they require expensive hardware for nonlinear optimization and prediction, which undermine the economy of application for autonomous racing.

To handle the complex nonlinear dynamics models, some researchers optimize the control strategy through real-world data sets and develop the learning-based frameworks [3], [4]. Through expert demonstrations and labeled data, neural networks can implicitly construct the mapping between raw observations and control strategy. Although these methods can effectively overcome the real-time and adaptability constraints of conventional approaches, they heavily rely on the graphics processing unit (GPU) and impose much higher hardware requirements [5]. Their performances are also limited by the quality and quantity of data sets so they can hardly outperform the human players [6].

According to real races, the vehicle dynamics may change during the competition (such as switching different tires). Meanwhile, professional players mainly make real-time decisions from vehicle states like speed, acceleration, and distance to the edge. This mechanism enables their policies and experiences to be plug-and-play when driving on a new track or with different dynamics. Therefore, we describe autonomous racing as a model-free local motion planning task without a global map. Besides, high-speed racing is a sensitive control application, where low latency and high precision are both required to catch the crucial braking and turning points [2]. Deep reinforcement learning (DRL) [7] is considered as a promising solution for this motion planning task [8]–[11]. Previous works demonstrate the outstanding performance of DRL for end-to-end autonomous driving and racing [12]–[15]. Classical DRL methods utilize the Gaussian probability distributions to cover workable action space. Nevertheless, in complex environments, updating from a randomly initialized policy to the optimal one is always time-consuming. For example, agents always crash into edges during turning and circle at the long straight. To jump out of these bad conditions, agents need more time-steps for exploration. Meanwhile, bad initialization sometimes causes premature convergence to local minima [16], [17]. Instead of learning from zero, it is much easier to learn the residual on the basis of guidance.

Hence, in this letter, we propose a model-free algorithm for autonomous racing with DRL and modified artificial potential field (MAPF) method named ResRace. It utilizes only the real-time observations of a 2D LiDAR and an inertial measurement unit (IMU) to achieve efficiency and low-latency control. We first leverage the raw LiDAR observation to select a local target. Then the MAPF-based controller provides a "guide-policy" through the point cloud and local target to guide the exploration and generate better experiences for training. Concurrently, the policy network is updated and provides a "residual-policy". Then the racing agent adopts the sum of them. In brief, our main contributions are three-fold:

1) We first define autonomous racing as a high-speed local motion planning task. With this prerequisite, we propose an efficient model-free algorithm ResRace algorithm, which is not dependent on prior information and has outstanding lap time grades and real-time performance.

2) Instead of learning from zero, we propose a novel method that works by optimizing the residual-policy with reinforcement learning algorithms on the basis of the guide-policy from a designed MAPF controller. With extensive experiments, we illustrate that our method can highly improve the learning efficiency of model-free DRL methods.

3) We compare our method with existing leading baselines. It outperforms two champion algorithms and reaches a comparable level to human players. We also provide validation results and a series of behavior examples to show our method can handle the different scenarios and system dynamics robustly and learn the racing techniques like human players.

II. Related Works

The autonomous driving task has been widely studied in the past decades [4], [18], [19]. Here, we divide previous studies by their methodology and they can be separated into three groups: classical hierarchical control approaches, supervised learning approaches, and reinforcement learning approaches.

Classical Control: Most previous researches describe autonomous driving as a hierarchical perception-planning-control task. In this paradigm, model predictive control (MPC) is widely used and demonstrates outstanding results in motion control [20], [21]. As an improved version, the MPC with Gaussian Process (GP)-based optimized dynamics models is a practical method for autonomous racing [22]. With the fine dynamics model and the optimized reference trajectories, these methods show excellent performances in both simulation and real-world. While the deficiency in flexibility and adaption prompts researchers to focus on the learning-based MPC [23]–[26]. By generating vehicle models and control strategies through the neural network, learning-based MPC can be utilized in diverse scenarios. Nevertheless, an inevitable dilemma is the trade-off between computation consumption and performance.

Deep and Imitation Learning: To skip the modeling, researchers develop the learning-based end-to-end methods to generate control policy directly from observation. The deep neural network possesses impressive ability in feature extraction and pattern recognition are impressive so that it can be deployed on autonomous vehicle [27], [28]. As a symbolic work, researchers successfully develop a convolutional neural network (CNN)-based controller to follow the road and lane [3]. Similarly, imitation learning leverages expert demonstration as a template and is proved to be feasible in off-road self-driving [4]. However, though they overcome the adaptability constraints, their performance depends on the scale and quality of data sets and they can hardly outperform the human [6].

Reinforcement Learning: Instead of establishing the data sets, classical reinforcement learning (RL) collects experiences and updates the policy by continuous intersection with dynamic environments. Prior works prove RL is an efficient solution for various complex tasks [12]–[15], [18]. Broadly, reinforcement learning is grouped into the model-based and model-free RL according to whether agent establishes the transition model. Many research proves that model-free RL is practicable in realistic racing games [12], [13], [15], but the stochastic exploration process could damage robots and hinders their applications in the real-world. Meanwhile, random initialization and sampling also cause an unstable training process and extensive time consumption. Contrarily, other researchers introduce the state-of-the-art (SOTA) model-based DRL algorithm Dreamer [29] to learn the system dynamics from interactions and construct latent imagination of future state [14]. However, this approach requires expert demonstration to pre-train and the imagination mechanism severely increases the computational and time consumption. Besides, the performance of model-based RL methods is significantly determined by the model accuracy [30]. In summary, though DRL achieves outstanding performance in both real and simulated scenarios, we still need a more robust and efficient solution for aggressive but safe autonomous racing.

III. Preliminaries

In this section, we give out the crucial definitions of the autonomous racing process and the detailed settings of the racing agent and simulation environment, which are the premise of our methodology in subsequent sections.

Problem Definition: We formalize autonomous racing as a partially observable Markov decision process (POMDP). The POMDP can be presented as a tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{O}, \mathcal{T}, \mathcal{R})$, where Ω and $\mathcal{R}$ are respectively the observation and reward. We will discuss our reward design and observation settings in the subsequent sections. $s \in \mathcal{S}$ is the possible state for racing agent and $u \in \mathcal{A}$ is the possible action. $\mathcal{O} : \mathcal{S} \times \Omega \to [0, 1]$ includes the system uncertainty and represent the probability of observation in a given state. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function under a specific state-action and also includes the uncertainty of system. Thus, an finite behavior trajectory can be presented as $\{s_0, u_0, r_0, s_1, u_1, r_1, \ldots, s_T, u_T, r_T\}$, where the subscript indicates the time-step within the maximum length T and r is the reward provided by defined $\mathcal{R}$.

Simulation Environment: In this letter, MAPF controller and DRL agent utilize the real-time observation of LiDAR and IMU. We leverage a PyBullet-based [32] F1Tenth environment for simulation [31]. We deploy the agents on the 5 tracks for evaluation with different difficulties as shown in Fig. 2. They are different in length, width and turn angles. For example, Montreal possesses the irregular track edges, changing width and hairpin turns. But Barcelona track is more narrow and has the long straights, V-shape turns and chicanes, which challenge the racing technique of agents. The diversity of tracks makes the racing task more challenging and ensures the reliability of our conclusions.

Agent Settings: In the F1Tenth simulator, the agent is a racing car with Ackerman steering and rigid body in unified robot description format (URDF) models. The racing agent is equipped with a 2D LiDAR with a maximum 10 meters range and 675 measurements evenly distributed over a 270° field of view. Besides, a 60 Hz IMU measures the motion of the agent, including the transverse and longitudinal velocity and acceleration in the horizontal plane. The actions of agent u are described as $u_a = \{T, \alpha\}$, where T presents the motor torque and α denotes the steering angle in $[-45^\circ, +45^\circ]$.

IV. Methodology

Based on the preliminaries, we illustrate the complementary property between the MAPF controller and DRL module, and the pipeline of ResRace in this section. Additionally, we explain the principles and methodology of the two modules in detail to emphasize their contributions.

A. Principle of ResRace

Learn Residuals above Guide-Policy: ResRace works by dividing the output policy $\pi_s(s|\theta) \rightarrow u_s$, which is learned from zero in previous DRL methods, into the residual-policy $\pi_a(s|\theta) \rightarrow u_a$ with trainable parameters θ and the guide-policy $\pi_m(s) \rightarrow u_m$. Thus, we can express this relationship as $\pi_s(s|\theta) = \pi_a(s|\theta) + \pi_m(s)$ or the action equation $u_s = u_a + u_m$. Importantly, the gradient satisfies $\bigtriangledown_\theta \pi_s(s|\theta) = \bigtriangledown_\theta \pi_a(s|\theta)$. In other words, the gradient of π_s does not depend on π_m so the optimal π_s is reachable through policy gradient method even π_m is not differentiable. By policy division, our method combines the complementary advantages of the DRL and MAPF and pushes the performance boundaries of what either can achieve independently. When the guide-policy is close to the optimal one, the residual-policy can be regarded as a corrective term. If it is defective, we describe it as a hint to guide the exploration [16]. In racing tasks, agents of classic DRL always obtain amount of low-reward experiences due to the collision, circling or wrong directions in initial episodes. We interpret contributions of guide-policy as providing a positive displacement towards the optimal one for the initial policy in the search space, so our policy skips these inefficient conditions and is much closer to the optimal.

Pipeline of ResRace: The structure of our algorithm is shown in the Fig. 1. For MAPF controller, it takes only the real-time point cloud as its input and outputs a guide action u_m. Each point in the point cloud is regarded as a tiny obstacle, and the target generator takes the point cloud and generates a local target. According to the position of local target and obstacles, $u_m \in [-1, +1]$ is calculated by the defined potential field and policy function. Concurrently, the policy network, a multi-layer perceptron (MLP) is randomly initialized and provides the u_a, which is set in $[-2, +2]$ to ensure it can cover the bad actions of MAPF controller. The racing agent takes the truncated sum of actions $u_s = clip\ (u_m + u_a,\ -1,\ +1)$ to drive. During training, the replay buffer stores the observation, reward and the residual u_a, then θ is updated with trajectories sampled from the replay buffer.

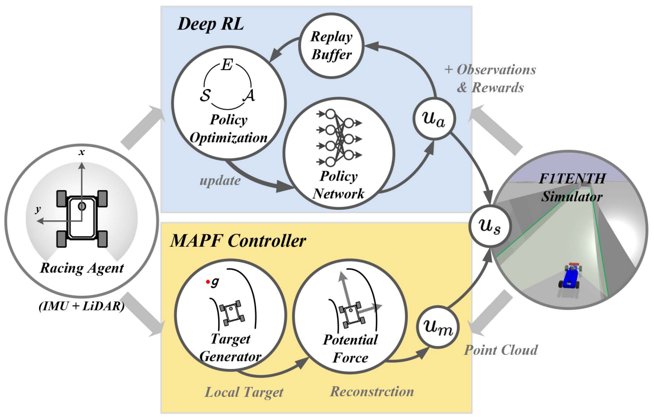


Fig. 1. The overview of ResRace. Our framework mainly consists of two parallel action generators. The MAPF controller provides a fundamental action u_m through the target generator and defined potential field to guide the exploration. The DRL agent provides a residual action u_a to optimize the global action $u_s = u_m + u_a$ and minimize the lap time.

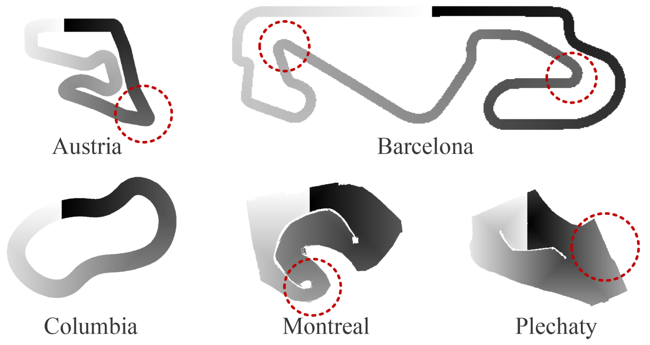
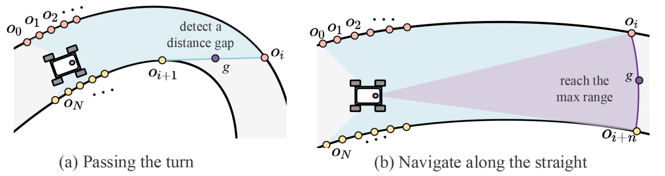


Fig. 2. The bird's-eye view of tracks in F1Tenth simulator [31]. With the increasing normalized distance in [0, 1], the color fades from black to white, which means agents are required to run clockwise. The red circles marks where previous model-free approaches failed to improve their performances, like the V-shape turns on Austria and Barcelona tracks, the chicane on Montreal circuit and the U-shape turn with irregular edges on Plechaty circuit.

B. MAPF Controller

Why MAPF: For local motion planning tasks, search-based approaches like A* and its variants are feasible and proved to generate the shortest path [33]. However, the optimal race line is generally between the minimum-distance curve and the minimum-curvature one [34] so it still needs further optimization. Meanwhile, these methods with discrete maps and action spaces require high resolution to generate precise actions for high-speed racing, which increases their time consumption [2]. Similarly, sample-based methods like rapidly-exploring random tree (RRT) and its variants [35]–[37] are also practical but paths generated by these methods are stochastic and not smooth, so they are also unsuitable for high-speed navigation. Importantly, the latency of the above methods is unstable in different scenarios. For example, the agent traveling at the wide segment in Montreal observes a larger area than that of it at the narrow one, so more time is required for searching and sampling. Contrarily, the actions generated by MAPF are much smoother, so it can handle high-speed vehicle control and it also contributes to suppressing the bang-bang control problem [38]. Meanwhile, the mapping from the forces to the commands is simple so a



(a) Passing the turn (b) Navigate along the straight

Fig. 3. The demonstration of local target selection when the racing agent drive (a) at the turn and (b) along the straight. The point cloud can be regarded as a set of tiny obstacles $\{o_1, o_2, o_3, \ldots, o_N\}$, where $N = 675$. The red and yellow points indicates the detected left and right edges.

dynamics model and controllers are not required. In addition, the time complexity of our MAPF depends on only the horizontal resolution of LiDAR so its real-time performance is stable. According to our experiments, our MAPF controller is faster than the DRL module. Thus, when they are paralleled, MAPF can hardly increase the global latency of ResRace.

Target Generator: The MAPF controller first utilizes raw observation of LiDAR to generate a local target g. Denote the detected points on the edges as $\{o_1, o_2, o_3, \ldots, o_N\}$, where $N = 675$ is the total measurements of LiDAR. As an intuitive demonstration in Fig. 3(a), when the agent drives at the turn, a large distance gap can be detected between o_i and o_{i+1} and we think the edge is switched. Generally, the center point of segment $o_i o_{i+1}$ is on the track and we selected it as the local target of the MAPF controller in this case. Another case is that the agent drives along the long straight as shown in Fig. 3(b). On the long straight, there is an observation arc $\{o_i, \ldots, o_{i+n}\}$ reaching the limitation range of LiDAR, which means the next turn is out of detection range. Thus, we select the center point $o_{\lfloor i+\frac{n}{2} \rfloor}$ of the arc as the local target.

$$\mathbf{U}_{att} = -\frac{1}{2}\mathbf{d}_g^2, \;\; \mathbf{U}_{rep}^j = \frac{1}{2}\mathbf{d}_o^j \left(1/\left|\mathbf{d}_o^j\right| + 1/d_e\right)^2 \tag{1}$$

Potential Field Definition: The potential field function is defined as (1). $\mathbf{U}_{att}$ is the attractive potential of the local target g. $\mathbf{d}_g$ is the relative position of the local target. Similarly, $\mathbf{U}_{rep}^j$ presents the repulsive potential of the j-th point in length-N observation and $\mathbf{d}_o^j$ presents its location. d_e is the effective distance of repulsive field and is set as 2 meters. Based on the above field functions, the attractive and repulsive force can be calculated as (2) and (3), respectively. After that, we transform the potential force into the normalized action in $[-1, +1]$ through a nonlinear $tanh$ function as (4). The general constant η is a general scaling factor of the attractive force.

$$\mathbf{F}_{att}^g = -\bigtriangledown \mathbf{U}_{att}^g \tag{2}$$

$$\mathbf{F}_{rep}^j = -\bigtriangledown \mathbf{U}_{rep}^j, \; s.t. \; d_e \leq \left|\mathbf{d}_o^j\right| \tag{3}$$

$$u_m = tanh \left(\eta \, \mathbf{F}_{att}^g + \sum_{j=1}^{N} \mathbf{F}_{rep}^j * \mathcal{X}_{xy}\right) \tag{4}$$

Modification in APF: The stagnation at dead points is a common issue for APF [39]. To solve this problem, we set a probability $\gamma = 0.1$ for the agent to explore forward with a tiny velocity when the stagnation occurs. Because $\mathbf{F}_{rep}$ is the sum of $N = 675$ terms with a large value, we need to scale it otherwise the discrete actions will be output by the hyperbolic tangent function. Meanwhile, the controller is sensitive to the lateral offset and a smaller scale is required. Thus, we set a general empirical factor $\mathcal{X}_{xy} = [0.05, 0.02]$ to scale repulsive forces on $\mathbf{x}$ and $\mathbf{y}$ directions unevenly. Our experimental results show this modification significantly contributes to ResRace performance.

C. Reinforcement Learning Module

Residual Policy Optimization: The policy optimization method is replaceable in the DRL module and we utilize two classic online model-free policy optimization algorithms TRPO [40] and PPO [41], which consist of policy and value networks. The policy network $\pi_a(s|\theta)$ with trainable parameters θ provides the residual-action u_a. Meanwhile, the value network $V(s|\theta_v)$ is parameterized by θ_v and estimates the value of state. The critic network is trained with the mean square error $\mathcal{L}_v(v_{ture}, v_e)$ between the estimation value v_e and the true one v_{ture}. The probability ratio is defined as $p_\theta = \pi_a^{new}(s, u_a)/\pi_a^{old}(s, u_a)$ to describe the similarity of the old policy π_a^{old} and the updated one π_a^{new}.

$$\hat{A}_t = -V(s_t) + r_t + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{5}$$

$$\mathcal{L}_{trpo} = \hat{\mathbb{E}}\left[p_\theta \hat{A} - \beta \mathrm{KL}\left(\pi_a^{old}, \pi_a^{new}\right)\right] \tag{6}$$

$$\mathcal{L}_c(\theta) = \hat{\mathbb{E}}\left[\min(p_\theta \hat{A}, \; clip\left(p_\theta, 1-\epsilon, 1+\epsilon\right)\hat{A})\right] \tag{7}$$

$$\mathcal{L}_{ppo} = \hat{\mathbb{E}}\left[\mathcal{L}_c(\theta) + \mathcal{H}(\pi_a(\cdot|s))\right] \tag{8}$$

Our method updates the networks with a stochastic gradient ascent algorithm through computing an estimator of policy gradient and advantage [42]. Here, the advantage function is defined as $\hat{A}$ in (5). Given a length-T trajectory as mentioned in Section III, t specifies the time-step in $[0, T]$. The expectation $\hat{\mathbb{E}}$ presents the empirical average over a finite batch of samples. In TRPO [40], the KL divergence is utilized to constrain the policy update and its surrogate objective is maximizing (6). In PPO [41], the policy update is constrained by the clipped objective as (7) with the ratio $\epsilon = 0.2$. We normalize $\hat{A}$ with batch statistics in practice, which effectively serves as an adaptive learning rate heuristic that bounds the gradient variance [43]. Besides, we set the policy entropy $\mathcal{H}(\pi_a(\cdot|s))$ as suggested in prior works [42] to ensure sufficient exploration, and the surrogate objective of PPO is maximizing the Equation (8). We leverage a temporal replay buffer to collect these finite trajectories, and the networks are updated with Adam optimizer [44]. According to the results, ResRace with PPO performs better than it with the TRPO in most cases.

$$\mathcal{R} = \mathcal{R}_{fin} + \mathcal{R}_a(u) - \mathcal{P}_a(u) - \mathcal{P}_s \tag{9}$$

Reward Design: The racing task is minimizing the lap time, which is equivalent to maximizing the lap progress in the given time-steps. For DRL method, its objective is to generate a policy to maximize the episode return. Hence, the reward function should present the intention of minimizing the lap time. Here, we define the reward function in ResRace as (9). The discrete reward $\mathcal{R}_{fin} = +100$ is issued when passing the finish line to encourage the agent to complete more laps in one episode. However, this sparse reward signal is difficult to attribute to specific actions, we thus add three continuous rewards. The continuous reward $\mathcal{R}_a(u)$ linearly depends on the agent velocity to encourage aggressive driving. The penalty $\mathcal{P}_a(u)$ linearly depends on the action difference of two adjacent time-steps and the absolute value of actions. It significantly restrains undesired bang-bang control of DRL methods [38], [45]. In racing, unnecessary and excessive actions cause a longer racing path and speed reduction. Meanwhile, the safety penalty $\mathcal{P}_s = 1$ restrains the collision with the track edge. Our reward definition can obviously accelerate the convergence and improve final performance, which will be demonstrated in the ablation experiment section.

V. Experimental Setup

In this section, we introduce the existing leading baselines and experimental settings. Additionally, we transfer our method to unseen tracks to demonstrate its generation ability. Besides, we illustrate its dynamics robustness and low parameter sensitivity. All experiments are conducted with INTEL Gold 5218R CPU and NVIDIA GeForce RTX 2070S GPU.

Baselines: We choose the winners of F1Tenth race in 2019 and 2020 as our competitive baselines. Other representative methods including MPC-based and model-free DRL approaches are also included.

1) RacingDreamer [14] wins the champion of F1Tenth 2020 and utilizes the SOTA model-based algorithm Dreamer [29] in racing task with raw LiDAR observation or occupancy reconstruction. We record them as Dreamer (LO) and Dreamer (OR) independently and train them with the process reward as suggested in the original letter.

2) Follow-the-Gap (FTG) [46] is a classic tracking method and wins F1Tenth 2019 champion. FTG is used as the expert demonstration to initialize the policy in RacingDreamer, so we consider it as a significant baseline.

3) Local learning MPC (LMPC) [23] utilizes closed-loop data from last lap to update the controller for the next with operator splitting solver for quadratic programs (OSQP) [47]. To initialize LMPC, we follow the path with 1 m/s in the initial 2 laps as the suggestion of authors.

4) Pure model-free DRL methods are included. We train five leading baselines including TRPO [40], PPO [41], DDPG [48], TD3 [49] and SAC [50]. In previous work [14] and our results, their performances are not well so we record the best one as an independent baseline MFRL (best).

5) Five professional formula student players are invited to control the agent by a game console. They can practice on arbitrary tracks, and observe the scenarios from the bird's-eye view (BE) or the following view (FV).

Training Settings: When an episode begins, the racing agent is placed randomly on the finish line and each episode has a maximum length of 5,000 time-steps. We train our ResRace for only 2 M time-steps to show its advantage on convergence speed, and other DRL baselines are trained for 5 M time-steps. For fair comparison, all model-free baselines including ResRace utilize the same MLP with two layers of 256 neurons and are trained for five trials with independent random seeds. Dreamer baselines are trained with the same settings as its original letter. For evaluation, all algorithms are tested for 100 K steps with 5,000 time-steps per episode. LMPC updates its controller every two laps and generally converges in 40 laps, so we test it after 50 laps for five trials and other settings are as the default value [23]. Human players are evaluated on the five tracks for 50,000 time-steps with two views and the average performance of their five best episodes is calculated as the baseline Human (BE) and Human (FV).

Performance Metrics: We first visualize the training curves in 2 M steps as Fig. 4 and analyze their learning abilities and other properties. Besides, we divide the lap progress by the episode length to calculate the surrogate lap time for all baselines according to their validation performance. When the agent is unable to move like crashing into the edges, or navigates towards the wrong direction, we think it is dead and a new episode will start. By this way, the efficiency of learning-based methods is improved. Importantly, when a method can not complete a full lap (except the longest track Barcelona) after training, we consider it fails on this track and its lap time is recorded as *Dnf.* (do not finish). Moreover, we evaluate their real-time performance on the model size and maximum frame rates in the simulator with the same scenario and hardware for 100 K test time-steps.

Robustness Validation: To evaluate the robustness of ResRace, we trial our models in different scenarios without retraining. We first increase the potential force $\mathbf{F}$ from $0.7\mathbf{F}$ to $1.3\mathbf{F}$ to validate its sensitivity to MAPF intensity. Meanwhile, we increase the original friction factor $\mu = 0.80$ from 0.55 to 0.95 to validate our models under different dynamics. Besides, we conduct the cross-validation by testing the trained agent on unseen tracks for $100\,K$ time-steps. We use the ratio of test results and results of the model trained on the test track to evaluate the generalization ability.

Ablation Studies: We test the performance of MAPF controller and pure PPO independently to illustrate their complementary property. Furthermore, to emphasize the contribution of modification in APF, we remove the anti-stagnation probability γ and replace the balancing array $\mathcal{X}_{xy}$ with $[0.05, 0.05]$ to restore its sensitivity to lateral offset. Besides, we replace our reward design with the original setting in the simulator as $\mathcal{R} = c\Delta p - \mathcal{P}_s$, where $c = 100$ is a constant scalar and Δp denotes the lap progress in one time-step.

VI. Experimental Results

In this section, we evaluate the baselines and visualize their experimental results. We also further discuss the causes of obtained results according to the simulator demonstration. In addition, we provide the comparison and behavior analysis of human players and ResRace during Barcelona racing process.

Training Process: In Fig. 4, because of the guide-policy, the lap progress in initial episodes of ResRace is much better than other methods. Model-free baselines and Dreamer explore with random actions so they collide with the track edges in most cases. Compared with MFRL (best), the MAPF-based policy effectively reduces the difficulty of policy optimization and highly contributes to performance improvement. Besides, though Dreamer reaches comparable performances to our method in 2 M steps on three tracks, its prediction process and extensive modeling lead to much higher algorithm complexity and more time consumption within the same training steps. Meanwhile, on the Montreal and Plechaty circuits, where the track width is uncertain and the edge shapes are irregular, conventional model-free methods and Dreamer can hardly learn the practicable policy.

Performance Comparison: In Table I, the performance of ResRace is better than other methods in most cases though

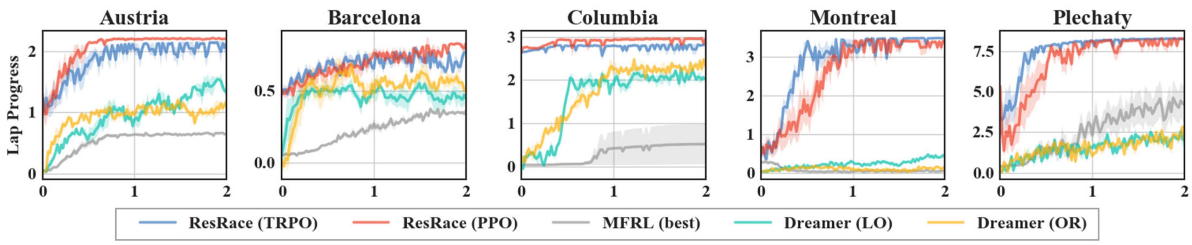


Fig. 4. The training process curves of reinforcement learning-based baselines in 2 million time-steps. Each episode consists of 5,000 time-steps. ResRace and other baselines are all trained for five trials with independent random seeds. Classic DRL methods learn from zero and early converge to the sub-optimal policy. Our methods perform better than other DRL baselines in both the initial and the last epochs due to learning on the basis of the guide-policy.

TABLE I
THE LAP TIME COMPARISON ON F1TENTH TRACKS

Methods	Lap Time (s)↓				
	Aus.	Bar.	Col.	Mon.	Ple.
FTG [46]	41.92	123.34	34.03	*Dnf.*	11.98
LMPC [23]	39.31	105.91	28.37	26.48	10.96
MFRL (best)	*Dnf.*	151.52	35.82	*Dnf.*	13.12
Dreamer (LO) [14]	37.56	102.21	**27.60**	*Dnf.*	16.34
Dreamer (OR) [14]	37.68	100.23	28.12	*Dnf.*	16.42
Human (BE)	37.57	97.89	29.24	25.54	10.52
Human (FV)	**36.77**	**95.82**	28.05	**23.02**	10.35
ResRace (TRPO)	**36.71**	99.21	28.15	**23.74**	**10.06**
ResRace (PPO)	36.87	**95.79**	**27.59**	23.88	**9.84**

Dnf. = Do not finish

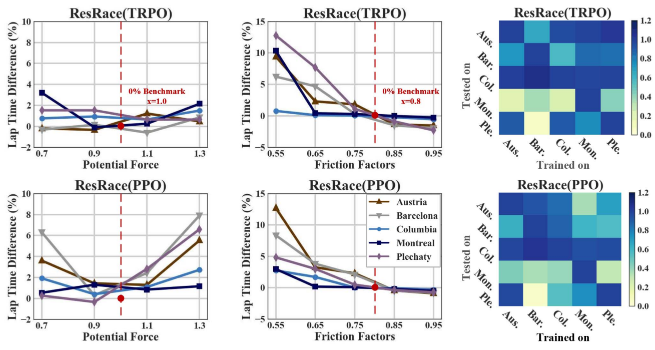


Fig. 5. The results of robustness and cross-validation. (a) and (b) show the performance of ResRace with different potential forces and dynamics. (c) illustrates the results of ResRace tested on unseen tracks. Each block represents the progress ratio of the tested model and the trained model. The darker the block, the better the generalization performance of the tested model. The red points mark the 0% benchmarks, which indicate the default settings with 1.0 potential force magnification and 0.8 tire friction factor.

they are trained for more steps. Especially, other approaches can hardly learn the practicable policy to pass the hairpin turn on the Montreal. Meanwhile, though Dreamer takes FTG as expert demonstration and outperforms the model-free methods, it easily converges to the sub-optimal policy and is still hard to defeat human players. LMPC obtains higher scores than that of FTG. We find it performs obviously better on the wide tracks than other tracks and almost catch up with the human players. But passing the narrow and sharp turns rapidly is still difficult for LMPC, and its race lines can be further optimized. For human players, the following view provides more intuitive observation so it is easier to control the distance from the track edges. As a result, the performance of Human (FV) is better than that of Human (BE). Besides, in most cases, ResRace can reach the comparable performance of human players with the following view. Although ResRace takes more $+0.62\ s$ than humans on the Montreal, the human players may crash into the edges during turning. Our results and demonstration indicate that ResRace adopts an aggressive policy and controls the vehicle at the dynamics limitation like professional players.

Efficiency: Although we harmonize the network with the same MLPs, the numbers of required models are various for different methods. Experimental results show our method requires two networks for value estimation and policy generation with only 0.48 M parameters, while other MFRL methods like SAC and TD3 require more target networks and therefore possess more parameters. Model-based Dreamer (LO) and Dreamer (OR) require more parameters for additional system dynamics modeling. Meanwhile, due to the latent imagination mechanism, Dreamer's sampling rates are respectively 38.62 Hz and 30.51 Hz with LO and OR, which requires much more computational resources to achieve the same real-time performance. Similarly, the prediction process of LMPC increases its latency, so it reaches only 45.92 Hz and is better than model-based DRL but much slower than our method. Meanwhile, by connecting the modules in series, ResRace reaches 211.27 Hz and 210.43 Hz sampling rates, which are only 9.7% and 10.3% lower than pure TRPO and PPO, respectively. These results illustrate that the MAPF controller only slightly affects the real-time performance of our framework.

Change Potential Force: Fig. 5(a) demonstrates that ResRace is not sensitive to the potential field intensity and indicates the parameters can be easily adjusted in the range of $\pm 30\%$. We notice that the performance of ResRace (PPO) with $1.3\mathbf{F}$ is reduced by about 8% on Barcelona. According to the simulation, when potential field intensity is too weak or strong, the racing agent yaws and navigates along the wave line due to the tiny or excessive transverse force. These undesired problems directly cause the collision at turns especially on the narrow track. Meanwhile, ResRace (TRPO) demonstrates lower parameter sensitivity than ResRace (PPO).

Change Tire Friction: Fig. 5(b) shows our method can adapt to the changing tire friction. With the increased friction, the racing agent can reduce its lap time with higher acceleration and turning speed. While the friction is reduced, though the performance shrinks slightly, it still finishes the laps stably. For instance, the turns are smoother in Barcelona than those in others so the agent requires less friction to pass them. On the contrary, when driving at the sharp turns of Austria and Barcelona, the racing agent must pass them with a longer path or lower velocity to reduce

TABLE II
THE RESULTS OF ABLATION EXPERIMENTS

Methods	**Lap Progress ↑**				
	Aus.	**Bar.**	**Col.**	**Mon.**	**Ple.**
MAPF Controller	1.31	0.43	2.64	0.64	6.25
Pure PPO	0.69	0.52	0.11	0.75	6.13
w/o Modification	2.15	0.81	2.72	**3.61**	7.59
w/o Reward Design	2.14	0.71	2.65	3.24	8.08
ResRace (PPO)	**2.26**	**0.87**	**3.01**	3.49	**8.47**

the required friction, which is consistent with experiences of human drivers.

Cross-Validation: As shown in Fig. 5(c), trained models maintain their performance when tested on the track with similar features. For example, the model trained on Austria can handle the sharp turns in Plechaty. On the contrary, the irregular and coarse edges of Montreal challenge the generalization of models trained on the other track, so that the test results on Montreal are always unsatisfactory. Meanwhile, the model trained on complex tracks demonstrates outstanding reliability. Specially, on Columbia, the model trained on Barcelona outperforms the Columbia baseline model and reaches 27.55 s per lap, which beats the professional player by 0.5 s and proves that the model trained on the challenging tracks process the outstanding generalization ability.

Ablation Studies: We validate contributions of crucial components in ResRace (PPO) as shown in Table II and ResRace (TRPO) has similar performances. The MAPF controller is workable in most cases and its maximum progress matches the initial performance of ResRace in Fig. 4. Meanwhile, TRPO and PPO can hardly work independently on these tracks. These results demonstrate that the initial exploration of ResRace is dominated by MAPF policy, and the subsequent improvement is attributed to the DRL module. For the modification in APF, experimental results (w/o Modification) illustrate that it contributes to the performance of our framework, especially on Columbia and Plechaty. It improves the performance by $+10.1\%$ on the tracks except for Montreal. When the track is wide, the modification decreases the lateral potential force excessively and causes a conservative steering. Thus, the performance of ResRace without modification outperforms the complete one on Montreal slightly. Besides, we replace our reward design with the original process reward in the simulator. According to the results (w/o Reward Design), our reward design can improve the average performance by 7.61%. In the simulation, the action penalty can effectively restrain the redundant steering and deceleration. Concurrently, with the discrete reward signal $\mathcal{R}_{fin}$, the agent learns to exit the last turn rapidly and sprint to the finish line. These results prove that our reward design can describe the task finely and effectively reduce the lap time. While after removing the reward design, the performance of ResRace is still better than most of the baselines.

Behavior Analysis: The Barcelona-Catalunya circuit possesses sixteen turns (denoted as T01 to T16) and is the official winter test track of Formula One for its outstanding design of turns. As shown in Fig. 6, we select three representative and challenging parts of this track, including the continuous T01-T02, the V-shape T05 and the chicane T14-T15. For continuous turns, ResRace agent adopts a larger entering angle and earlier turning point than those of human players at the T01, which reduces its

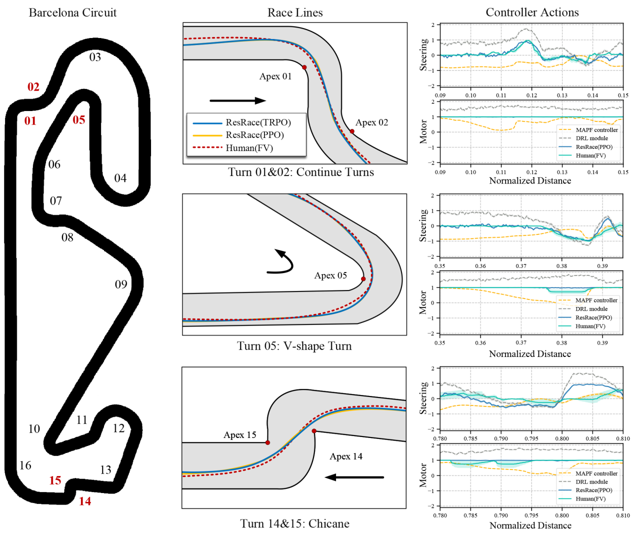


Fig. 6. The behavior demonstration on the Barcelona circuit. The red numbers indicate the difficult turns on this track, and the race lines of ResRace agents and human players are demonstrated in the second column. The steering/motor-normalized distance curves of ResRace (PPO) and its modules are shown in the third column. By comparing them with the action curve of human players, we can analyze the differences of their driving preferences.

velocity. But it passes the next turn with a smaller angle so the exiting speeding is higher at T02. For the V-shape turn, instead of approaching the apex, our agent adopts a later turning point and catches the late apex of T05. Meanwhile, human players release the throttle to decelerate at their turning points, while our agent keeps the maximum power, which can reduce the time consumption during turning when the traction is large enough. At the chicane T14-T15, ResRace chooses an earlier turning point and grasps two apexes, and then exits T15 with the full throttle and turns sharply to catch the apex of T16. However, human players prefer to decelerate and pass the chicane with a smaller curvature. We also visualize the action materials in the third column of Fig. 6. The throttle actions of MAPF controller are cautious and the residual actions from DRL module are aggressive after training. In the initial epochs, the output of ResRace is mainly determined by the MAPF controller, and the agent drives along the center line. However, the trained agent prefers to navigate along the track edges to minimize its lap time. In other words, with the increasing training steps, the guidance ability of MAPF begins to decline and the residual-policy determines the output, especially for the steering action.

VII. CONCLUSION

In this letter, we propose a residual policy learning method ResRace for model-free local motion planning. Instead of the inefficient learning-from-zero in classic DRL methods, we design an efficient MAPF to provide guidance for the agent. The experimental results show that our model-free framework can outperform a series of leading approaches and reach comparable level of human players. Meanwhile, we evaluate the robustness and compare the actions of our method and human players through extensive validations and demonstrations. Besides, ResRace works robustly with different policy optimization methods, APF parameters and vehicle dynamics. We also notice the limitation of our method. Although the MAPF controller can improve the exploration efficiency, the guide-policy is soft and may be covered by the residual-policy during initial exploration. Meanwhile, the performance of ResRace is significantly related to that of the fundamental policy. Although our residual learning framework only needs a workable policy to guide the exploration, the guide-policy still requires fine-tuning and further revision. Additionally, our method still needs extensive validation in the real world. In future work, we will further improve the performance of our method and explore its application in more scenarios.

References

[1] G. Williams et al., "Aggressive driving with model predictive path integral control," in *Proc. Int. Conf. Robot. Autom.*, 2016, pp. 1433–1440.

[2] D. Kalaria et al., "Local NMPC on global optimised path for autonomous racing," 2021, *arXiv:2109.07105*.

[3] M. Bojarski et al., "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.

[4] Y. Pan et al., "Agile autonomous driving using end-to-end deep imitation learning," in *Proc. Robot. Sci. Syst.*, 2018, doi: 10.15607/RSS.2018.XIV.056.

[5] Y. E. Wang, G. Y. Wei, and D. M. Brooks, "Benchmarking TPU, GPU, and CPU platforms for deep learning," 2019, *arXiv:1907.10701*.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[7] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] J. Hwangbo et al., "Control of a quadrotor with reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017.

[9] W. Koch et al., "Reinforcement learning for UAV attitude control," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 2, pp. 1–21, 2019.

[10] C. You et al., "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robot. Auton. Syst.*, vol. 114, pp. 1–18, 2019.

[11] C. Yang et al., "Multi-expert learning of adaptive legged locomotion," *Sci. Robot.*, vol. 5, no. 49, 2020, Art. no. eabb2174.

[12] E. Perot et al., "End-to-end driving in a realistic racing game with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop*, 2017, pp. 3/4.

[13] M. Jaritz et al., "End-to-end race driving with deep reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 2018, pp. 2070–2075.

[14] A. Brunnbauer et al., "Latent imagination facilitates zero-shot transfer in autonomous racing," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 7513–7520.

[15] F. Fuchs et al., "Super-human performance in gran turismo sport using deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4257–4264, Jul. 2021.

[16] T. Silver, K. Allen, J. Tenenbau, and L. Kaelblin, "Residual policy learning," 2018, *arXiv:1812.06298*.

[17] T. Johannink et al., "Residual reinforcement learning for robot control," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 6023–6029.

[18] S. Levine et al., "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[19] G. Chen et al., "Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 34–49, Jul. 2020.

[20] J. Kabzan et al., "Learning-based model predictive control for autonomous racing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019.

[21] M. F. Elmorshedy et al., "Recent achievements in model predictive control techniques for industrial motor: A comprehensive state-of-the-art," *IEEE Access*, vol. 9, pp. 58170–58191, 2021.

[22] A. Jain et al., "BayesRace: Learning to race autonomously using prior experience," in *Proc. Conf. Robot Learn.*, 2020, pp. 1918–1929.

[23] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: A predictive control approach," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2713–2719, Nov. 2020.

[24] G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 2017, pp. 1714–1721.

[25] G. Williams et al., "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, 2018.

[26] G. Bellegarda and K. Byl, "An online training method for augmenting MPC with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020, pp. 5453–5459.

[27] A. Mehta et al., "Learning end-to-end autonomous driving using guided auxiliary supervision," in *Proc. Indian Conf. Comput. Vis., Graph. Image Process.*, 2018, pp. 1–8.

[28] T. Weiss and M. Behl, "Deepracing: Parameterized trajectories for autonomous racing," 2020, *arXiv:2005.05178*.

[29] D. Hafner et al., "Dream to control: Learning behaviors by latent imagination," in *Proc. Int. Conf. Learn. Representations*, 2019.

[30] A. Plaat, W. A. Kosters, and M. Preus, "Model-based deep reinforcement learning for high-dimensional problems, a survey," 2020, *arXiv:2008.05598*.

[31] A. Brunnbauer and L. Berducci, "Racecar gym," Github Repository, 2020. [Online]. Available: https://github.com/axelbr/racecar_gym

[32] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning," *Github Repository*, 2016. [Online]. Available: https://github.com/bulletphysics/bullet3

[33] P. E. Hart et al., "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[34] J. Betz et al., "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.

[35] S. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," *Comput. Sci. Dept. Oct.*, vol. 98, no. 11, 1998.

[36] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[37] M.-C. Kim and J.-B. Song, "Informed RRT* towards optimality by reducing size of hyperellipsoid," in *Proc. IEEE Int. Conf. Adv. Intell. Mech.*, 2015, pp. 244–248.

[38] T. Seyde et al., "Is bang-bang control all you need? Solving continuous control with bernoulli policies," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2021, pp. 27209–27221.

[39] J. Sun et al., "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[40] J. Schulman et al., "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[42] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[43] G. Tucker et al., "The mirage of action-dependent baselines in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5015–5024.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*," 2015.

[45] J. P. LaSalle, "The 'bang-bang' principle," *IFAC Proc.*, vol. 1, no. 1, pp. 503–507, 1960.

[46] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "Follow the gap method," *Robot. Auton. Syst.*, vol. 60, no. 9, pp. 1123–1134, 2012.

[47] B. Stellato et al., "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.

[48] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[49] S. Fujimoto et al., "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1582–1591.

[50] T. Haarnoja et al., "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Machin. Learn.*, 2018, pp. 1861–1870.