Robust UAV Hovering Control under Aggressive Conditions via Bidirectional Thrust and Deep Reinforcement Learning

Tianyou Liu[1], Zhihong Liu[1*], Xiaoxin Li[1], Xiangke Wang[1]
[1]College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China

***Abstract*—Most quadrotor control methods are designed under the assumption that motors produce only positive thrust. However, these methods struggle to maintain stability and ensure the safety of the quadrotor under aggressive conditions. To overcome this limitation, we propose a robust UAV hovering control method under aggressive conditions via bidirectional thrust and deep reinforcement learning, which directly outputs motor-thrust commands. More specifically, we build a bidirectional motor–propeller model that includes an explicit dead-zone representation during thrust reversal. To compensate for this dead-zone and improve robustness to model uncertainties, we include action history in the observation to handle the delay and apply domain randomization over dynamics parameters and initial states during training. Experimental results show that our method recovers hover from aggressive conditions with an inverted attitude and a large downward velocity. In comparison, a conventional only positive-thrust control and an idealized bidirectional-thrust control baseline both crash. Ablation studies show that dead-zone modeling, action history, and domain randomization are necessary for robust performance.**

***Index Terms*—quadrotor; bidirectional thrust; deep reinforcement learning; neural network controller; low-level control**

I. Introduction

Unmanned aerial vehicles (UAVs), particularly quadrotors, have been widely deployed across domains due to their simple operation and decent maneuverability [1]–[3]. Most quadrotor platforms are typically designed such that each of the four propellers spins in a prescribed direction (two clockwise and two counterclockwise), generating thrust only in one direction. We refer to such scheme as only positive-thrust control (OPTC). OPTC simplifies controller design but restricts the vehicle's action space, preventing the platform from fully exploiting its potential under aggressive conditions.

With advances in hardware, modern quadrotor platforms allow the vehicle to generate both positive and negative thrust by reversing the motors direction, which makes bidirectional-thrust control (BTC) feasible. BTC enlarges the reachable attitude and trajectory space, enabling robust control under aggressive conditions such as fully inverted flight, large attitude deviations, and high descent rates. However, achieving bidirectional-thrust control robustly and efficiently remains a central challenge due to the quadrotor's inherent underactuation and the prolonged nonlinear dead-zone delay during motor reversal.

Numerous studies have been proposed in the field of BTC. A typical pipeline is to first derive a dynamics model of the vehicle and then apply a standard controller, such as cascaded Proportional–Integral–Derivative (PID) control to realize BTC. Michini *et al.* [4] are the first to demonstrate BTC using unidirectional motors with variable-pitch propellers. They employ cascaded PID control to achieve inverted flight and half-flip hover, thereby establishing the feasibility of bidirectional thrust. Building on this line, Jothiraj *et al.* [5] report inverted maneuvering of a BTC quadrotor, and later propose a refined control-allocation scheme to address motor saturation under bidirectional operation [6]. Maier *et al.* [7] implement BTC using reversible motors and fixed, symmetric propellers and analyze the characteristics of symmetric blades. Nevertheless, these cascaded PID-based approaches are strongly dependent on manual gain tuning and system identification and struggle to adapt to unmodeled dynamics and external disturbances.

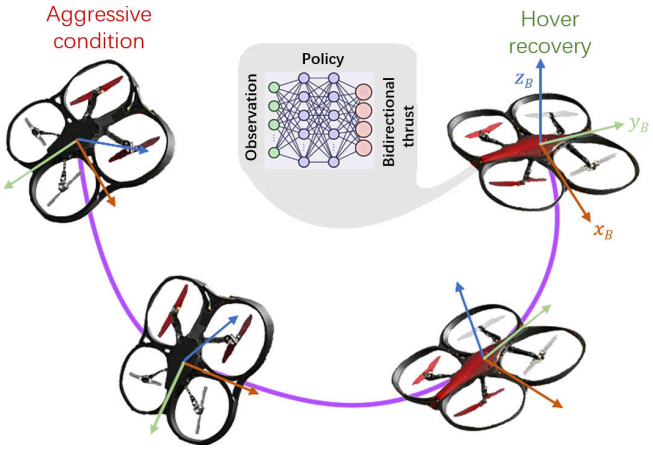


Fig. 1. Overall aggressive hover-recovery task and the RBTC control concept. The quadrotor starts from an aggressive initial state and recovers to the desired hover state, where the policy maps the current observation to per-motor bidirectional thrust commands.

Beyond PID, another line of work adopts model-based optimization methods such as MPC to improve BTC robustness via receding-horizon optimization. Jothiraj *et al.* [8] provide a detailed analysis of the motor response in both directions and introduces the dead-zone delay into MPC by modeling it using a standstill speed threshold and constraints on the thrust rate of change. Wehbeh *et al.* [9] approximate the dead-zone delay as a linearized constraint within MPC. These model-based approaches can achieve excellent performance under accurate models. However, they rely heavily on precise dynamics and

parameter identification. Uncertainties and disturbances in the environment can lead to model mismatch. In addition, the heavy computational burden of solving an optimization problem at each control step severely limits system responsiveness, making it difficult to accomplish highly challenging flight scenarios.

In recent years, learning-based control methods [10], [11] have been increasingly applied to quadrotor control. They enable end-to-end policy optimization without explicit modeling of complex dynamics. Hwangbo *et al.* [12] first propose a deep reinforcement learning (DRL)–based low-level controller that maps states directly to the four motor thrust commands. The controller achieves fast and robust hovering without requiring an explicit dynamics model. Molchanov *et al.* [13] employ domain randomization to train a single neural network for three different quadrotor platforms. Their approach does not require system identification or pre-tuned cascaded controllers, which demonstrates the robustness of DRL-based neural controllers. The University of Zurich group further explore DRL for autonomous drone racing. They achieve performance that surpass professional human pilots [14]. They also show that RL can outperform optimal control under richer task objectives [15]. However, these approaches are restricted to OPTC platforms. They do not leverage the expanded action space introduced by BTC, which provides clear advantages under aggressive conditions such as inverted attitudes or large attitude deviations.

To date, our prior work [16] provides the only work that combines RL with BTC. We integrate RL into a BTC framework and, under ideal dynamics, achieve rapid hover recovery from aggressive conditions, demonstrating that BTC can yield smoother and faster control compared with OPTC baselines. However, this work is evaluated in an idealized simulation setting and does not explicitly account for the nonlinear dead-zone delay during motor reversal. As a result, further investigation is needed to assess its performance and robustness under more realistic conditions and potential model mismatch.

In this paper, we propose Robust Bidirectional-Thrust Control (RBTC), a robust UAV hovering control method that leverages bidirectional thrust and deep reinforcement learning to ensure stability under aggressive conditions. Our goal is to achieve robust hover recovery under aggressive conditions shown in Fig. 1, where strong external disturbances or abrupt changes in operating conditions drive the vehicle far away from the nominal hover equilibrium, leading to large and rapid transients in attitude and velocity. Therefore, we let the policy directly output per-motor bidirectional thrust commands, which are obtained by mapping the observation to the action under a bidirectional motor–propeller model with an explicit dead-zone during thrust reversal. To compensate for the dead-zone delay, we augment the RL observation with action history so that the policy can capture the delay process. In addition, we apply large-scale domain randomization over the initial states, dynamics parameters, and dead-zone parameters to enhance robustness under unseen conditions. Finally, we validate RBTC through extensive comparative experiments and simulations, together with systematic ablation studies. In summary, the main contributions of this paper are summarized as follows:

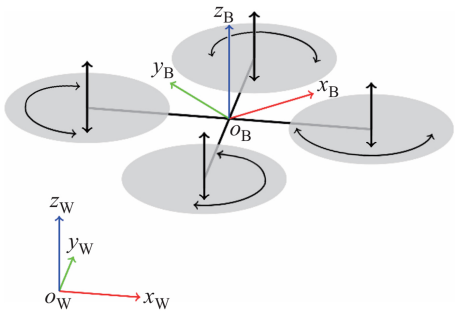


Fig. 2. Quadrotor structure and coordinate frames

1) We propose RBTC, a robust UAV hovering control method that leverages bidirectional thrust and deep reinforcement learning to enable robust hover under aggressive conditions.
2) We propose a series of robustness-enhancing strategies by modeling the thrust-reversal dead-zone delay, augmenting observations with action history, and applying domain randomization during training.
3) We validate RBTC through comparative simulations using a high-fidelity physics simulator against OPTC [12] and an idealized BTC [16] baseline, and perform ablation studies to quantify the necessity and contribution of each proposed component.

II. METHODOLOGY

In this section, we present the method in three parts. First, we introduce the dynamic model of a quadrotor with bidirectional thrust, including the geometry and coordinate frames, the 6-DoF rigid-body dynamics, and a bidirectional motor–propeller model with an explicit dead-zone representation during thrust reversal. Second, the DRL-based neural network controller is described by formulating the problem as a Markov decision process and specifying the observation space, action space, and reward function. Finally, we detail the training procedure in a simulator named Flightmare [18] with a physics engine.

A. Dynamic Model of a Quadrotor with Bidirectional Thrust

1) Geometry and Frames: The quadrotor considered here has an X-shaped frame (Fig. 2) and is modeled as a 6-DoF rigid body actuated by four motors. We use the world frame $\mathcal{W}$ and body frame $\mathcal{B}$.

2) Dynamics: Based on Newton's second law and the Newton–Euler equations, the equations of motion of the UAV are given by:

$$\dot{p}_{WB} = v_{WB}, \tag{1}$$

$$\dot{q}_{WB} = \tfrac{1}{2}\,\mathbf{\Lambda}(\boldsymbol{\omega}_B)\,q_{WB}, \tag{2}$$

$$\dot{v}_{WB} = q_{WB} \odot c + \mathbf{g}, \tag{3}$$

$$\dot{\boldsymbol{\omega}}_B = \mathbf{J}^{-1}(\boldsymbol{\eta} - \boldsymbol{\omega}_B \times \mathbf{J}\,\boldsymbol{\omega}_B)\,. \tag{4}$$

Here, $p_{WB} = [p_x, p_y, p_z]^\top$ and $v_{WB} = [v_x, v_y, v_z]^\top$ denote the UAV position and linear velocity in the world frame. To avoid the gimbal singularities associated with Euler angles, the attitude is represented by the unit quaternion $q_{WB} = [q_w, q_x, q_y, q_z]^\top$, and $\boldsymbol{\omega}_B = [\omega_x, \omega_y, \omega_z]^\top$ is the angular velocity expressed in the body frame. The translational motion is driven by the gravitational acceleration $\mathbf{g} = [0, 0, -g]^\top$ and the body-frame thrust acceleration $\mathbf{c} = [0, 0, c]^\top$, while the rotational motion is driven by the control torque vector $\boldsymbol{\eta}$. The inertia matrix is $\mathbf{J} = \mathrm{diag}(I_x, I_y, I_z)$. The matrix $\boldsymbol{\Lambda}(\boldsymbol{\omega}_B)$ is skew-symmetric and expands as:

$$\boldsymbol{\Lambda}(\boldsymbol{\omega}_B) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}. \tag{5}$$

The gravitational constant is set to $g = 9.81\,\mathrm{m\,s^{-2}}$. The body-frame thrust acceleration magnitude is:

$$c = \frac{f_1 + f_2 + f_3 + f_4}{M}, \tag{6}$$

where $f_i (i = 1, 2, 3, 4)$ denotes the thrust generated by rotor i. According to the quadrotor geometry, the control torque vector is:

$$\boldsymbol{\eta} = \begin{bmatrix} \frac{L}{\sqrt{2}} (f_1 - f_2 - f_3 + f_4) \\ \frac{L}{\sqrt{2}} (-f_1 - f_2 + f_3 + f_4) \\ K_\tau (f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \tag{7}$$

where L is the arm length, K_τ is the rotor torque–thrust coefficient, M is the total mass.

3) Bidirectional Motor–Propeller Model: To enable bidirectional thrust control of the quadrotor, based on symmetric propellers and a bidirectional brushless motor, we propose a bidirectional-thrust motor–propeller model. According to blade element–momentum theory, the thrust generated by rotor i is:

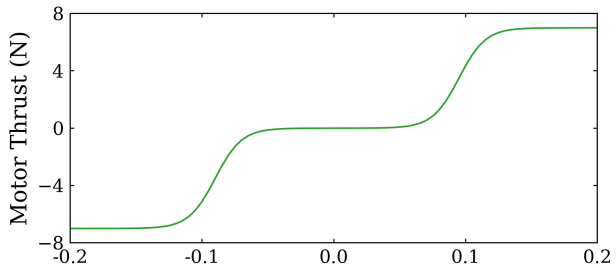
$$f_i = K_f \cdot \mathrm{sgn}(\Omega_i) \cdot \Omega_i^2, \tag{8}$$

where K_f is the propeller thrust coefficient, $\mathrm{sgn}(\Omega_i)$ denotes the motor rotation direction, and Ω_i is the angular speed of rotor i. For a single motor, the rotor speed Ω is modeled as a first-order system:

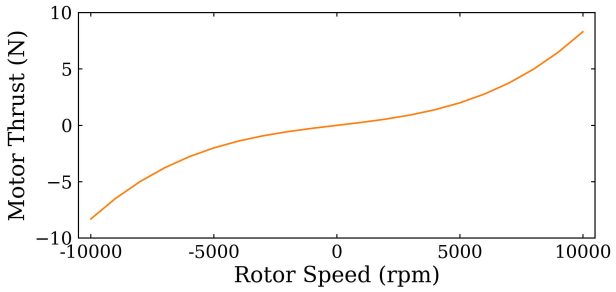
$$\dot{\Omega} = \frac{\Omega_{des} - \Omega}{K_\alpha}, \tag{9}$$

where K_α is the motor time constant and Ω_{des} is desired rotor speed.

4) Dead-Zone Modeling During Reversal: When the motor reverses its rotation direction, it undergoes a short near-zero-thrust interval that we call the dead-zone. Accurate modeling of this delay is crucial for realistic simulation and reliable learning. In this work, we adopt the motor model proposed in [8]. The model is identified by fitting curves to experimental data collected from a T-Motor Air 2213/920KV equipped with an APC-3D 9×4.5 symmetric propeller, as shown in Fig. 3.



(a) Thrust–time response curve



(b) Thrust–motor response curve

Fig. 3. Thrust response characteristics of the bidirectional motor–propeller system.

The model assumes that the dynamics of each motor act independently. It also assumes symmetric thrust capability in the positive and negative directions. Empirical response curves indicate that a full reversal from $-7\,\mathrm{N}$ to $+7\,\mathrm{N}$ takes about $0.4\,\mathrm{s}$ in total. Around reversal, the thrust remains near zero for roughly $0.1\,\mathrm{s}$, which reduces the effective control authority. Accordingly, we model the thrust–speed mapping with a logistic sigmoid function so that the thrust saturates once the motor reaches its rated maximum speed. The relationship between thrust and rotor speed is given by:

$$T_C(\omega) = \begin{cases} T_{\max} & \omega \geq \omega_r \\ \dfrac{T_{\max}}{1 + \exp\left[-\dfrac{12}{\omega_r}\left(\omega - \dfrac{\omega_r}{2}\right)\right]} & 0 < \omega < \omega_r \\ 0 & \omega = 0 \\ -\dfrac{T_{\max}}{1 + \exp\left[\dfrac{12}{\omega_r}\left(\omega + \dfrac{\omega_r}{2}\right)\right]} & -\omega_r < \omega < 0 \\ -T_{\max} & \omega \leq -\omega_r \end{cases}, \tag{10}$$

where $T_{\max}$ is the maximum thrust, ω_r is the positive saturation speed, and ω is the current motor speed.

B. Design of Neural Network Controller with DRL

To take full advantage of the quadrotor capabilities, we formulate control as an infinite-horizon Markov decision process (MDP). The MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. The function

p is the transition function. The function r is the reward. The discount factor is $\gamma \in (0,1)$. The goal is to find a policy π^* that maximizes the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right]. \tag{11}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots)$ denotes the trajectory generated by the policy π. The variable a_t denotes the action at time step t. Next, we introduce the observation space, the action space, and the reward function.

1) Observation Space: The UAV state s_t is defined as a 12-dimensional vector:

$$s_t = [p_x, p_y, p_z, v_x, v_y, v_z, \varphi_z, \theta_y, \rho_x, \omega_x, \omega_y, \omega_z]^{\mathrm{T}}, \tag{12}$$

where $[p_x, p_y, p_z]$ denotes the UAV position in the world coordinate frame, $[v_x, v_y, v_z]$ represents the UAV velocity in the world frame, $[\varphi_z, \theta_y, \rho_x]$ denotes the Euler angles of the UAV attitude, and $[\omega_x, \omega_y, \omega_z]$ represents the angular velocity of the UAV body frame. The final desired state of the UAV for hover stabilization is denoted by s_{des}. The difference between the current state s_t and the desired state s_{des}, denoted as Δs_t, is included as part of the observation space. It is noted that the motor switching delay is approximately 100 ms, which is significantly larger than the time interval of the low-level controller operating at several hundred hertz. To mitigate this effect, the historical control actions $\mathbf{H}$, i.e., the thrust history of the motors, are also incorporated into the observation space. $\mathbf{H}$ is a vector of dimension $N_H \times 4$, where N_H represents the length of the action history. Consequently, the observation vector of the UAV is defined as:

$$\mathbf{O}_a = [\Delta s_t, \mathbf{H}], \tag{13}$$

which serves as the input to the neural-network-based controller.

2) Action Space: To fully exploit the capability of the UAV, we let the policy directly map states to the four desired continuous motor thrusts, i.e., $\pi : s_t \mapsto \mathbf{a}_t = [f_1^{\mathrm{des}}, f_2^{\mathrm{des}}, f_3^{\mathrm{des}}, f_4^{\mathrm{des}}]^{\mathrm{T}}$. This particular action space preserves agility. By contrast, higher-level action outputs such as x–y–z axis velocities or collective thrust and body rates (CTBR) often sacrifice agility and responsiveness.

Using the inverse of the bidirectional motor–propeller model in (8), the desired rotor speeds are:

$$\Omega_i^{\mathrm{des}} = \mathrm{sgn}\big(f_i^{\mathrm{des}}\big)\sqrt{\frac{f_i^{\mathrm{des}}}{K_f}} \qquad i = 1, \ldots, 4. \tag{14}$$

The next state s_{t+1} is then obtained by numerically integrating the multirotor dynamics over one control step.

3) Reward Function: To drive the UAV to quickly stabilize at the hover target, we design the reward function as follows:

$$\begin{aligned} r_t = {} & \alpha_p \left\|\Delta p_t\right\|_2 + \alpha_v \left\|\Delta v_t\right\|_2 + \alpha_o \left\|\Delta o_t\right\|_2 \\ & + \alpha_\omega \left\|\Delta \omega_t\right\|_2 + \alpha_a + \alpha_c. \end{aligned} \tag{15}$$

The components of the reward function are, in order, the penalties on the position error, linear-velocity error, attitude error, and angular-velocity error which are the errors between desired and actual values, followed by the alive bonus and the crash penalty. The dominant terms are the position penalty $\alpha_p\|\Delta p_t\|_2$ and the attitude penalty $\alpha_o\|\Delta o_t\|_2$, which drive the UAV to rapidly reorient and move toward the desired state and therefore largely determine the gradient direction during DRL optimization; in our implementation we set $\alpha_p = 0.1$ and $\alpha_o = 0.1$. The linear-velocity and angular-velocity penalties, $\alpha_v\|\Delta v_t\|_2$ and $\alpha_\omega\|\Delta\omega_t\|_2$, are designed to reduce aggressive actuation and oscillations and thus improve the control behavior, and we set $\alpha_v = 0.01$ and $\alpha_\omega = 0.01$. The coefficient α_a corresponds to the alive bonus, which enhances robustness throughout the episode, and we set $\alpha_a = 0.1$ when the UAV remains alive. The coefficient α_c corresponds to the crash penalty, which is introduced to help the policy quickly discover feasible control at the beginning of training while avoiding crash, and we set $\alpha_c = -10$ when the UAV crashes.

C. Training

1) Policy Network: The policy network is a multilayer perceptron (MLP) with two fully connected layers of 64 neurons and $\tanh$ activations. This architecture can capture nonlinear features while mitigating overfitting risk, and is suitable for onboard deployment where computational resources are limited. The controller architecture and the PPO training loop are illustrated in Fig. 4. For training, we use Proximal Policy Optimization (PPO) [17] to learn the policy network. PPO is a policy-gradient algorithm that improves stability and sample efficiency by constraining the size of policy updates through a clipped surrogate objective. At each control step, the observation $\mathbf{O}_a = [\Delta s_t, \mathbf{H}]$ is fed to both the actor and critic. The actor outputs the desired motor thrusts $\mathbf{a}_t = [f_1^{\mathrm{des}}, f_2^{\mathrm{des}}, f_3^{\mathrm{des}}, f_4^{\mathrm{des}}]^{\mathrm{T}}$, which are executed through the bidirectional motor–propeller model with dead-zone dynamics. The critic estimates the value function, which is used to compute PPO updates from collected rewards.

2) Domain Randomization: In simulation, many factors such as parametric uncertainties, noise, and external disturbances are often neglected due to the use of idealized kinematic and dynamic models. To improve robustness and generalization under such modeling uncertainties, we adopt a domain randomization strategy so that the policy is trained in environments with different dynamic parameters and initial states, thereby improving its robustness to parameter variations and external disturbances.

Specifically, at the beginning of each training episode, we randomly perturb several key dynamic parameters of the UAV, including the arm length, thrust coefficient, drag coefficient, motor constant, and the motor speed ramp time constant w_{ramp}. These parameters are independently sampled within $\pm 30\%$ of their nominal values. This design encourages the policy to maintain stable control performance across models with different propulsion characteristics and inertial properties, which reduces dependence on a single nominal model.

Furthermore, to broaden the distribution of states visited during training and enhance the network's ability to handle

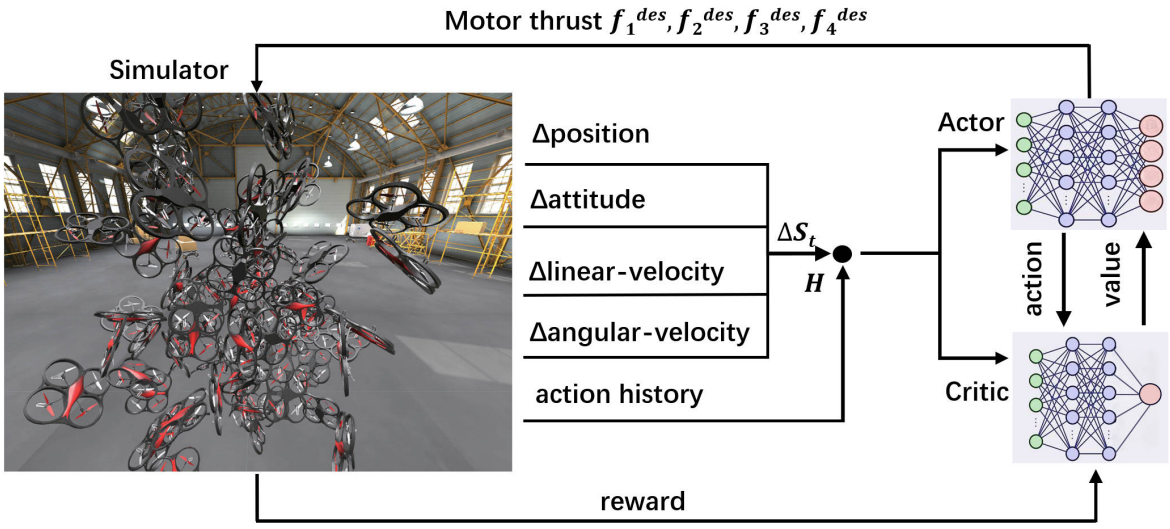


Fig. 4. Overview of the DRL-based RBTC controller and the PPO training loop. At each control step, the state error and action history are provided to both the actor and critic. The actor outputs desired motor thrusts $\mathbf{a}_t = [f_1^{\mathrm{des}}, f_2^{\mathrm{des}}, f_3^{\mathrm{des}}, f_4^{\mathrm{des}}]^{\mathrm{T}}$, which are executed through the bidirectional motor–propeller model with dead-zone dynamics. The critic estimates the value function for computing PPO updates from collected rewards.

TABLE I
PPO HYPERPARAMETERS.

Parameter	Value
Learning rate	3×10^{-4}
Discount factor γ	0.99
GAE weight λ	0.95
Training epochs	10.0
Policy network	MLP[64,64]
Value network	MLP[64,64]
Clip range	0.2
Entropy coefficient	0.0
Batch size	1.0

large state deviations, we also randomize the initial states of the UAV. The target hover position is set to $[0, 0, 5]^{\mathrm{T}}$. At the beginning of each episode, the position is sampled uniformly within a cube of side length $2\,\mathrm{m}$ centered at this target. The initial attitude is drawn uniformly from the feasible orientation set. The initial linear velocity along each axis is sampled from $[-1, 1]\,\mathrm{m\,s^{-1}}$. The initial angular velocity along each axis is sampled from $[-1, 1]\,\mathrm{rad\,s^{-1}}$.

3) Training Environment and Parameters: The simulation is conducted in the high-reality physics engine Flightmare [18]. We integrate the bidirectional motor–propeller model into Flightmare and update the UAV dynamics according to the motor thrust–speed response curve. The PPO hyperparameters are summarized in Table I.

Simulation runs on an Intel Core i9-12700K CPU and an NVIDIA RTX 3090 GPU. Each trajectory τ has a horizon of $10.0\,\mathrm{s}$ with a time step of $\Delta t = 0.02\,\mathrm{s}$, which yields 500 steps per trajectory. A trajectory is terminated upon collision. Each iteration consists of $50{,}000$ environment steps. The total number of iterations is set to $10{,}000$. To accelerate data collection, we adopt parallel training. Ten environment threads are executed concurrently, and 100 UAVs collect samples in parallel. Following the above training procedure, the training results are shown in Fig. 5, where the policy exhibits stable convergence.

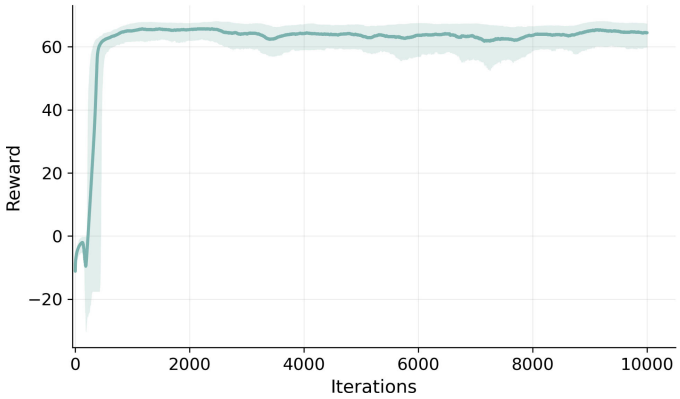


Fig. 5. Reward curve of RBTC.

III. EXPERIMENTS

In this section, we conduct comparative experiments with the BTC and OPTC method, in order to demonstrate that our approach achieves superior maneuverability and robustness. We then perform ablation studies to show that each proposed component is necessary for realizing RBTC.

A. Hover Recovery

To verify the effectiveness and superiority of RBTC, we design two evaluation scenes for hover recovery experiments under highly aggressive conditions, and compare RBTC against OPTC and an idealized BTC baseline. In the first scene, the initial states are set to position $[0, 0, 5]^{\top}$, velocity $[0, 0, 0]^{\top}$, angular velocity $[0, 0, 0]^{\top}$, and Euler angles (in z–y–x order) $[0, 0, \pi]^{\top}$. Under this initial condition, the UAV starts from a completely inverted attitude at a height of $5\,\mathrm{m}$, so it has only about $1\,\mathrm{s}$ to perform a recovery maneuver before crashing. This setup poses an aggressive demanding test of the control authority under limited altitude.

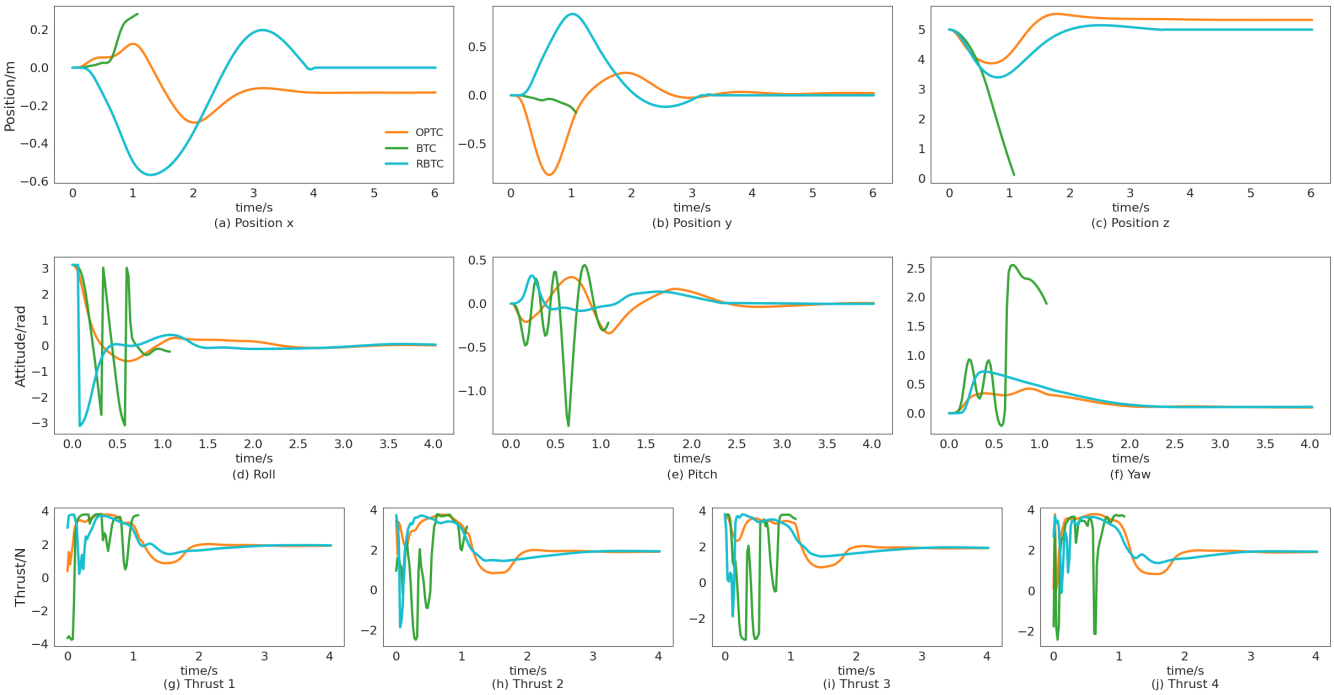


Fig. 6. Hover recovery from an inverted initial attitude. The three rows show the variations of the UAV position, attitude, and motor thrusts, respectively. Curves end at crash or stable hovering. BTC terminates early due to crash.

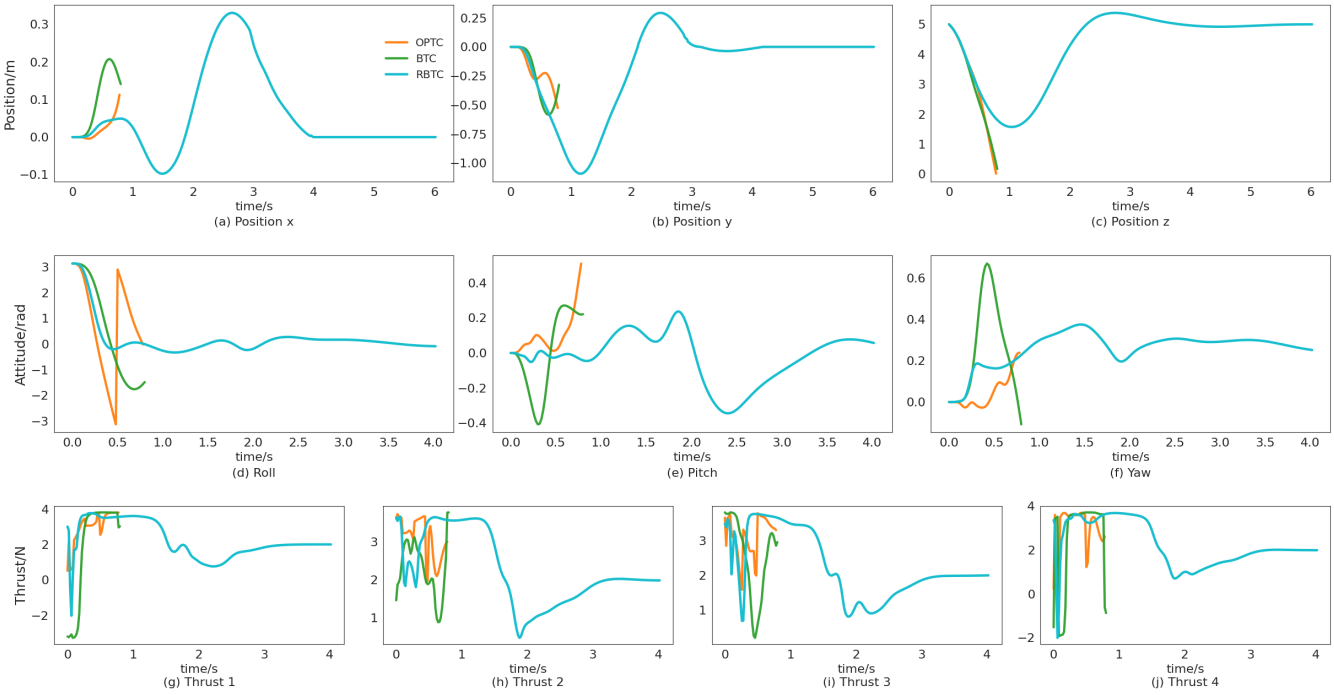


Fig. 7. Hover recovery with an additional initial downward velocity ($v_z = -2$ m/s). Only RBTC avoids ground impact and recovers to hover. OPTC and BTC terminate early due to crash.

The results of the hover-recovery experiment are summarized in Fig. 6. Fig. 6(a)–(c) report the position evolution, Fig. 6(d)–(f) show the attitude response, and Fig. 6(g)–(j) plot the four motor thrust commands. The trajectories are truncated at the crash time for failed methods. RBTC and OPTC successfully accomplish hover recovery, whereas the BTC rollout terminates early due to crash. RBTC reaches the hover position faster and shows smaller residual errors in the horizontal axes. OPTC exhibits a noticeable steady-state bias, which is consistent with the limited action space when only positive thrust is available. The attitude traces in Fig. 6(d)–(f) indicate that RBTC completes the flip from the inverted initial attitude to near-upright within the first second and then damps out the remaining oscillations, whereas OPTC shows larger transient oscillations and a longer settling process. The thrust profiles in Fig. 6(g)–(j) show that RBTC produces brief reverse-thrust pulses during the flip and then transitions to smoother positive thrust commands for hovering. BTC changes thrust signs aggressively. It loses effective control authority around the zero-thrust interval because it does not explicitly handle the thrust-reversal dead zone.

To further evaluate RBTC under a more aggressive condition, we add an additional initial downward velocity of $v_z = -2$ m/s on top of the above setting. Fig. 7 presents the corresponding time histories in the same format as Fig. 6. In the vertical motion (Fig. 7(c)), RBTC initially loses altitude but successfully arrests the descent before ground impact and climbs back to the hover height, whereas both OPTC and BTC fail to arrest the descent and crash. The attitude responses in Fig. 7(d)–(f) show that RBTC is able to complete the reorientation while keeping the attitude bounded throughout the maneuver, while the BTC and OPTC diverge during the initial recovery phase. The thrust profiles in Fig. 7(g)–(j) reveal that RBTC performs a short reverse-thrust phase at the beginning of recovery, followed by a transition to forward thrust, enabling both rapid reorientation and sufficient vertical support for a successful hover recovery. The results demonstrate that RBTC is more robust under aggressive conditions.

To better understand how RBTC achieves recovery in the more aggressive setting, we obtain Fig. 8 by uniformly sampling the full trajectory of OPTC, BTC, and RBTC, which visualizes the attitude and position evolution from the initial condition to stable hovering. RBTC generates reverse thrust to rapidly reorient the UAV back to an upright attitude, and then transitions to forward thrust to settle at the target hover state.

B. Ablation studies

To illustrate the importance of each component for RBTC, we conduct ablation studies under the more aggressive hover-recovery setting in Fig. 7 by removing one component at a time. Fig. 9 plots the time histories of the position, attitude, and motor thrusts, where each ablated variant is labeled in the legend by the removed component (RBTC includes all components). When the dead-zone model or the action-history augmentation is removed, the rollouts terminate within the first second. The altitude in Fig. 9(c) rapidly drops to the ground,

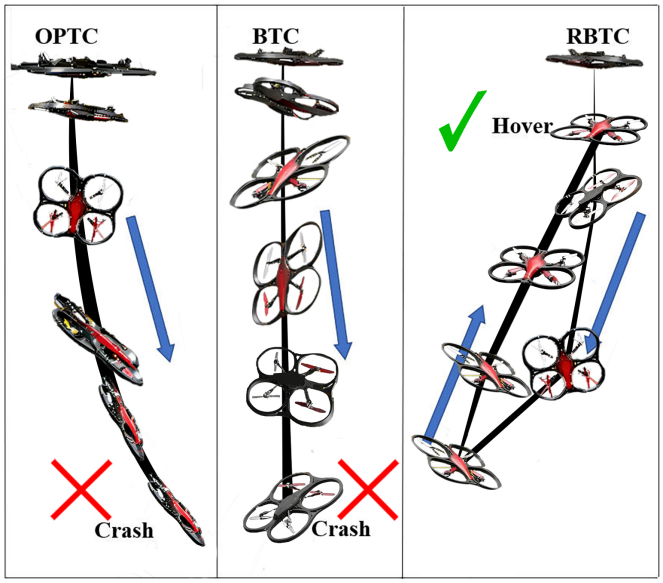


Fig. 8. Attitude and position evolution of the UAV under the more aggressive condition in Fig. 7. OPTC and BTC terminate early due to crash. RBTC adjusts the vehicle attitude by producing brief reverse thrust to flip the UAV upright, and it then applies forward thrust to stabilize near the target hover.

and the attitude signals in Fig. 9(d)–(f) become unstable. When domain randomization is removed, the UAV can still recover, but the transients degrade noticeably. Fig. 9(a)–(c) show larger position overshoot, and Fig. 9(e)–(f) show stronger attitude oscillations. Table II confirms these trends. The time-to-hover increases from 3.98 s to 5.96 s. The integral of absolute position error (IAE) increases from 5.34 to 7.39 m·s. The minimum-height (MH) margin decreases from 1.59 m to 1.40 m. The steady-state error (ess) increases from 0.004 m to 0.167 m. Overall, Fig. 9 and Table II suggest that dead-zone modeling and action-history observation are necessary for feasibility under thrust reversal, while domain randomization is important for achieving faster, smoother, and safer recoveries.

TABLE II
ABLATION STUDY

Ablation	SR (%)	Time-to-hover (s)	IAE (m·s)	MH (m)	e_{ss} (m)
All components	**100**	**3.98**	**5.34**	**1.59**	**0.004**
DR	100	5.96	7.39	1.40	0.167
Action history	0	/	/	/	/
Dead-zone model	0	/	/	/	/

The ablation entry indicates the removed component, except for the first row (all components). SR: success rate. Time-to-hover: the first time when the UAV reaches the hover condition and remains in this condition for W seconds. IAE: time integral of the position error magnitude. MH: minimum height above ground before termination. e_{ss}: steady-state position error over the last W seconds. "/" indicates crash and thus unavailable metrics.

IV. CONCLUSION

In this paper, we presented a robust UAV hovering control method under aggressive conditions via bidirectional thrust and deep reinforcement learning, which directly outputting the motor thrust commands. The proposed method explicitly accounts for the motor dead-zone delay during thrust reversal by modeling the dead-zone process, incorporating history

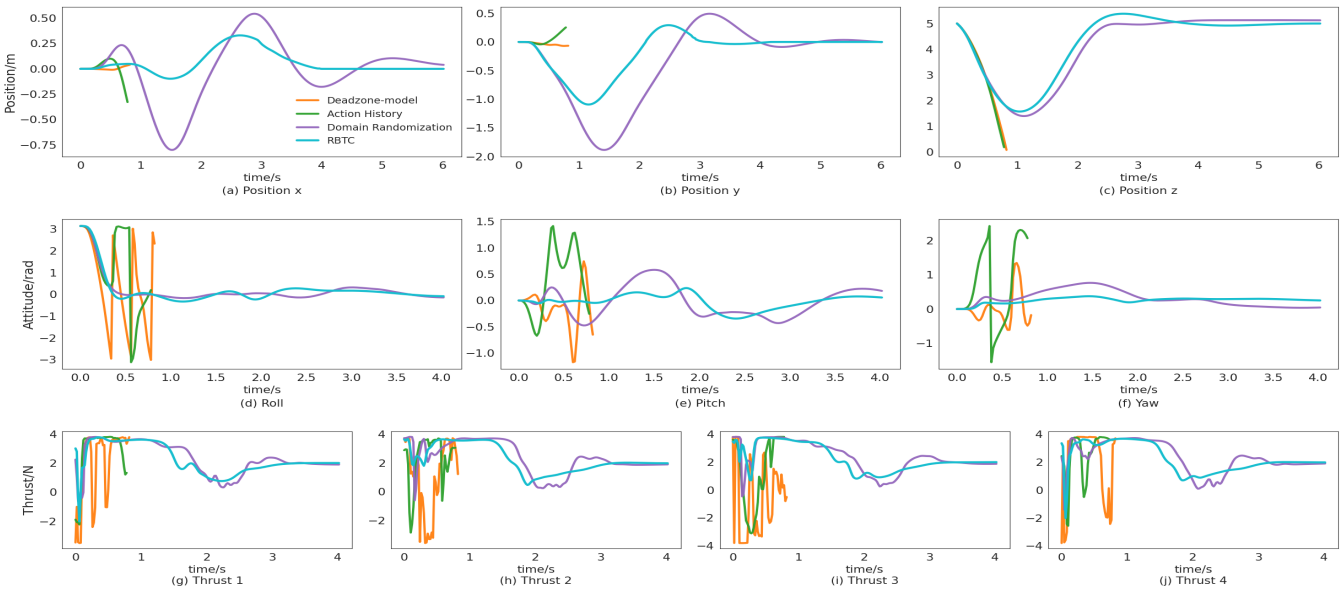


Fig. 9. Ablation results under the aggressive setting in Fig. 7. The legend indicates the removed component, except for RBTC, which includes all components. Removing the dead-zone model or action history leads to crash; removing domain randomization degrades transients and increases state fluctuations.

information into the observation space, and applying domain randomization during training, thereby improving control robustness in highly dynamic maneuvers. Experimental results demonstrate that our approach can successfully accomplish hover recovery under highly aggressive conditions, outperforming representative OPTC and the idealized BTC baseline. Future work will focus on real-world deployment and further bridge the sim-to-real gap. Moreover, we are interested in extending the policy to handle even more aggressive and rapidly changing environments.

Acknowledgment

This research is funded by National Natural Science Foundation of China (Grant No. 62576355). The corresponding author is Zhihong Liu.

References

[1] T. Wu, Y. Chen, T. Chen, G. Zhao, and F. Gao, "Whole-body control through narrow gaps from pixels to action," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 11 317–11 324.

[2] Y. Chen and N. O. Pérez-Arancibia, "Controller synthesis and performance optimization for aerobatic quadrotor flight," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2204–2219, 2019.

[3] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *arXiv preprint arXiv:2006.05768*, 2020.

[4] B. Michini, J. Redding, N. K. Ure, M. Cutler, and J. P. How, "Design and flight testing of an autonomous variable-pitch quadrotor," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2978–2979.

[5] W. Jothiraj, C. Miles, E. Bulka, I. Sharf, and M. Nahon, "Enabling bidirectional thrust for aggressive and inverted quadrotor flight," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 534–541.

[6] W. Jothiraj, I. Sharf, and M. Nahon, "Control allocation of bidirectional thrust quadrotor subject to actuator constraints," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 932–938.

[7] M. Maier, "Bidirectional thrust for multirotor mavs with fixed-pitch propellers," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[8] W. Jothiraj, "Control of a bidirectional quadrotor for aggressive maneuvers," McGill University, Montréal, Canada, 2020.

[9] J. Wehbeh and I. Sharf, "Nonlinear scenario-based model predictive control for quadrotors with bidirectional thrust," *International Journal of Robust and Nonlinear Control*, vol. 34, no. 18, pp. 12 450–12 475, 2024.

[10] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[11] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[12] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[13] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 59–66.

[14] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[15] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[16] X. Li, Z. Liu, G. Wang, and X. Wang, "Bidirectional thrust control of quadrotor UAVs based on deep reinforcement learning," *Robot*, vol. 47, no. 3, pp. 305–314, 2025, doi: 10.13973/j.cnki.robot.240329. (in Chinese)

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

[18] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*. PMLR, 2021, pp. 1147–1157.