2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Learning Robust Control Policies for Inverted Pose on Miniature Blimp Robots

Yuanlin Yang, Lin Hong, and Fumin Zhang†

***Abstract*— The ability to achieve and maintain inverted poses is essential for unlocking the full agility of miniature blimp robots (MBRs). However, developing reliable inverted control strategies for MBRs remains challenging due to their complex and underactuated dynamics. To address this challenge, we propose a novel framework that enables robust control policy learning for inverted pose on MBRs. The proposed framework consists of three core stages. First, a high-fidelity three-dimensional (3D) simulation environment is constructed and calibrated using real-world MBR motion data. Second, a robust inverted control policy is trained in simulation using a modified Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm combined with a domain randomization strategy. Third, a mapping layer is designed to bridge the sim-to-real gap and facilitate real-world deployment of the learned policy. Comprehensive evaluations in the simulation environment demonstrate that the learned policy achieves a higher success rate compared to the energy-shaping controller. Furthermore, experimental results confirm that the learned policy with a mapping layer enables an MBR to achieve and maintain a fully inverted pose in real-world settings.**

I. INTRODUCTION

The potential of agile flying in Unmanned Aerial Vehicles (UAVs) has been extensively demonstrated using proportional–integral–derivative (PID) control [1], model predictive control (MPC) [2], and deep reinforcement learning (DRL) [3], [4]. However, for MBRs, a distinct category of aerial platforms, a significant gap remains in the development of advanced control strategies capable of delivering comparable agility. UAVs typically rely on high-speed rotating propellers for lift and maneuvering, which inherently results in high energy consumption and potential safety risks when operating in proximity to humans. In contrast, MBRs utilize buoyant gas to offset their weight and employ low-power thrusters for fine-grained motion control. This unique design has positioned MBRs as a promising solution for various applications, such as entertainment and advertising [5], warehouse inventory management [6], indoor environmental monitoring [7], and infrastructure inspection [8].

Existing research on MBRs has predominantly centered on innovative structural design, including optimization of the envelope shape, gondola layout, and payload integration to enhance operational stability and adaptability [9], [10].

The work described in this paper was partially supported by grants AoE/E-601/24-N, 16203223, N_HKUST677/24, C6029-23G, and C6078-25G from the Research Grants Council of the Hong Kong SAR, China.

Yuanlin Yang, Lin Hong, and Fumin Zhang are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. E-mails: yyanghy@connect.ust.hk, eelinhong@ust.hk, eefumin@ust.hk

†Corresponding author: Fumin Zhang (email: eefumin@ust.hk)

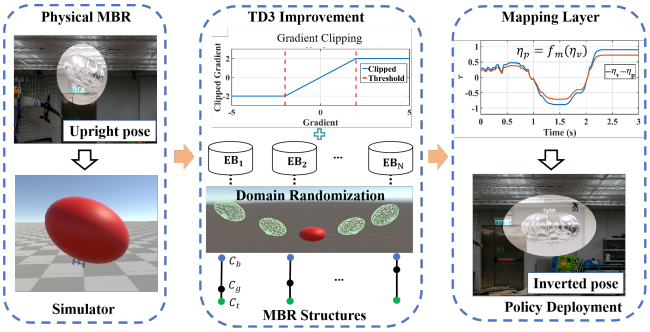

Fig. 1: Overview of the proposed method for inverted pose on MBRs. The MBR with an upright pose can reach and maintain the inverted pose with the learned robust policy.

Control studies have mainly addressed small pitch or yaw adjustments for basic hovering [11] or low-speed navigation [12]. However, fully agile motion control of MBRs, a capability that would enable rapid attitude transitions and wide-range position adjustments, remains an open challenge. This challenge arises from the unique dynamic properties of MBRs. In low-speed small UAV applications, aerodynamic drag is typically negligible relative to total thrust, as UAVs rely on high-power propellers to counteract their full weight; consequently, their thrust output significantly exceeds drag forces during low-speed motion. For MBRs, two key characteristics reverse this relationship: (1) Dominant aerodynamic drag due to their large envelope volume; (2) Weak thrust output, since buoyant gas offsets most of their weight, eliminating the need for high thrust to counteract gravity.

These distinct dynamic properties make MBR attitude control fundamentally different from that of UAVs, rendering conventional UAV control strategies largely inapplicable. In addition, a review of MBR designs [9], [10], [13]–[16] further highlights a structural constraint: most MBRs adopt a gondola-envelope configuration, where the gondola, housing sensors, thrusters, and controllers, is suspended below or attached to the envelope. This structure inherently exhibits both stable and unstable equilibrium points: for example, the "upright" pose (gondola hanging below the envelope) is a stable equilibrium, while the "inverted" pose (gondola above the envelope) is unstable and difficult to maintain.

Against this backdrop, the primary task in this paper is to enable MBRs to reach and maintain an inverted pose based on DRL, as illustrated in Fig. 1. To clarify the problem formulation, we define *inverted control* as the ability of an MBR to achieve and stabilize a fully upside-

down pose—corresponding to an unstable equilibrium state in which the center of buoyancy lies below the center of gravity. The work most closely related is a recent study by Wang and Zhang [17], which explicitly tackles the challenge of inverted control for MBRs. Their approach successfully demonstrates the achievement and maintenance of a stable inverted pose using model-based control: an energy-shaping controller to tailor the system's energy landscape for state transitions, paired with a linear state feedback controller to suppress deviations from the target inverted state. However, the energy calculation at the core of their controller depends on time-invariant MBR dynamics, yet the model parameters are highly dynamic in real-world operations, leading to performance degradation or even loss of inverted stability under environmental disturbances. Recent progress in DRL-based control has shown promise for addressing the parameter variability and disturbance susceptibility of outdoor large-size blimp robots. Liu et al. [18] proposed a deep residual reinforcement learning method integrated with a PID controller in a closed loop. This hybrid approach improved trajectory tracking accuracy but remained limited to small-range attitude control, with no consideration of inverted states. Zou et al. [19] designed a hybrid control framework that combines robust H_∞ control with proximal policy optimization (PPO), which enhances robustness against wind disturbances and variations in buoyancy.

Despite these advances, developing DRL-based methods for inverted control of MBR remains largely unexplored. In this paper, we address this gap by presenting a robust policy specifically designed for inverted control of MBRs. Our approach combines domain randomization, multi-buffer experience replay, and a sim-to-real transfer strategy to expand the applicability of learning-based control to MBRs. By focusing on inverted control as a cornerstone of large-envelope agility, this study aims to unlock new capabilities for MBRs. The main contributions are as follows:

- To the best of our knowledge, this work presents the first Unity-based 3D simulation environment [20] specifically designed for inverted control of MBRs. The simulator captures MBR-specific dynamics and enables diverse scenario generation for robust policy training.
- We propose a learning framework for robust inverted control of MBRs. The framework integrates domain randomization to improve robustness against parameter variations and disturbances, and introduces refinements to TD3 to enhance training stability.
- We develop a sim-to-real transfer strategy with a mapping layer to compensate for discrepancies between simulated and physical dynamics. Experimental results demonstrate that the learned policy reliably achieves inverted stabilization on a real MBR without additional policy retraining.

II. Problem Formulation

A. Dynamic Model of MBRs

As shown in Fig. 2, the MBR consists of an envelope and a gondola: the envelope is to supply the buoyancy, while the

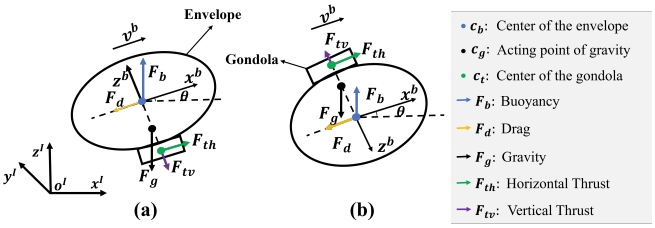

Fig. 2: Dynamic model analysis of the MBR: (a) Upright pose; (b) Inverted pose.

gondola is to provide a platform for housing thrusters and other electrical devices. Based on the first principle and [12], [21], the dynamic model of the MBR can be expressed as:

$$(\boldsymbol{M_{rb}}+\boldsymbol{M_a})\dot{\boldsymbol{\nu}}^b_{b/e}+(\boldsymbol{C_{rb}}+\boldsymbol{C_a})\boldsymbol{\nu}^b_{b/e}+\boldsymbol{\Gamma}^d+\boldsymbol{\Gamma}^{gb}+\boldsymbol{\Gamma}^e=\boldsymbol{\Gamma}^t, \tag{1}$$

where $\boldsymbol{\Gamma}^d$ denotes air drag, $\boldsymbol{\Gamma}^t$ and $\boldsymbol{\Gamma}^e$ correspond to thruster-generated and environmental forces and torques, respectively. $\boldsymbol{\Gamma}^{gb}$ involves the restoring force and torque.

The dynamic models near the upright and inverted poses are shown in Figures 2 (a) and (b), respectively. In both cases, the MBR operates at a constant velocity while holding a stable attitude. Based on the force analysis, we have

$$\mathbf{F}_{th}r^b_t+m_{rb}\mathbf{g}r^b_z\sin(\theta)=\mathbf{0},\mathbf{F}_b+\mathbf{F}_g=\mathbf{0},\mathbf{F}_{th}+\mathbf{F}_d=\mathbf{0}, \tag{2}$$

where r^b_t is the distance between c_b and c_t, r^b_z is the distance between c_b and c_g, and m_{rb} is the MBR total mass.

B. Inverted Control Problem Statement

According to (1) and (2), the MBR exhibits highly nonlinear dynamics, and its attitude control performance is sensitive to variations in model parameters. The objective of this study is to design a robust control policy, denoted as π_α, that drives the MBR from its stable equilibrium state ($\boldsymbol{\Theta}_0$) to the unstable equilibrium state ($\boldsymbol{\Theta}_d$), and maintains stabilization in its vicinity. The orientation of the MBR is represented as $\boldsymbol{\Theta}=[\phi,\theta,\psi]^T$, where ϕ, θ, and ψ denote roll, pitch, and yaw angles, respectively. The control objective can therefore be formulated as:

$$\lim_{t\to\infty}||\boldsymbol{\Theta}_d-\boldsymbol{\Theta}_t||_{M_\Theta}=0,\boldsymbol{\Theta}_t=T_b(\boldsymbol{\Theta}_{t-1},\boldsymbol{a}_t,\mathbf{S}_b), \tag{3}$$

where T_b is the dynamics transition from $\boldsymbol{\Theta}_{t-1}$ to $\boldsymbol{\Theta}_t$ under the control command $\boldsymbol{a}_t$ generated by the designed policy π_α. $\mathbf{S}_b$ involves the parameters in the MBRs' dynamic model.

The policy π_α aims to maximize the total cumulative reward the MBR receives over the long run through interaction with the environment, formulated as:

$$\max_\alpha \quad G_t \tag{4}$$

$$\text{s.t.} \quad G_t=\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}, r_t=f_r(\mathbf{s}_t,\mathbf{s}_{t+1},\boldsymbol{a}_t), \tag{5}$$

$$s_{t+1}=T^v_b(\mathbf{s}_t,\boldsymbol{a}_t,\mathbf{S}^v_b),\boldsymbol{a}_t=\pi_\alpha(\mathbf{s}_t), \tag{6}$$

where G_t is the cumulative reward, γ is the discount factor, and f_r is the designed reward function to evaluate the action

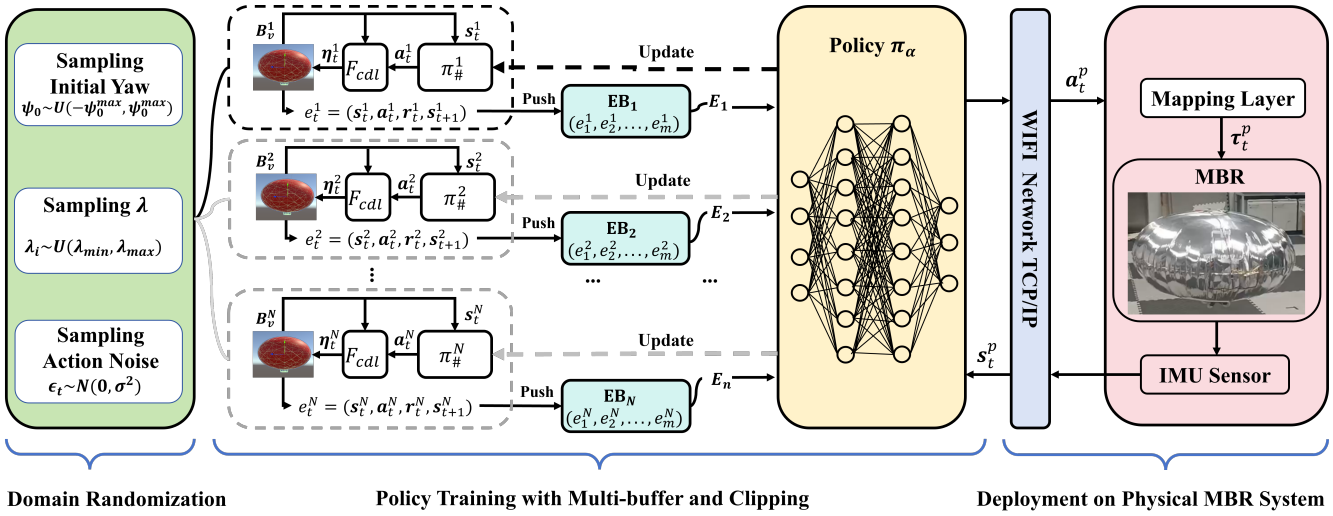

Fig. 3: Pipeline of developing and deploying robust policy for inverted control of MBRs.

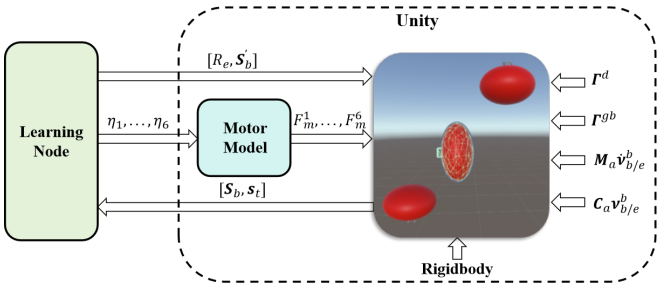

Fig. 4: 3D simulation environment designed for robust policy learning for inverted control of MBRs.

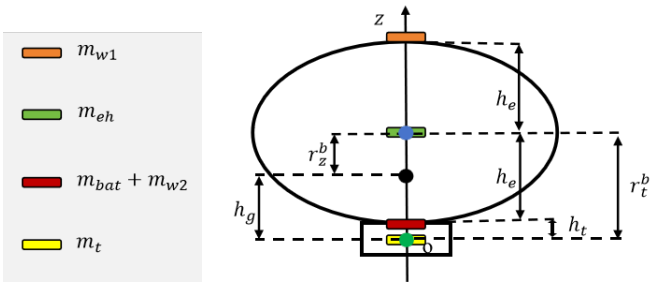

Fig. 5: The MBR's structure in the simulation environment.

$\boldsymbol{a}_t$ taken in state $\mathbf{s}_t$. T_b^v and $\mathbf{S}_b^v$ represent the dynamics transition and the parameters of the MBR in the simulation environment, respectively.

Due to unmodeled dynamics and parameter mismatches in the training environment, bridging the gap between the simulated environment and the physical setting needs to be considered in the policy deployment phase. Assuming there is a mapping function f_m that can bridge the gap, the problem is formulated as:

$$\lim_{t\to\infty} ||\mathbf{s}_d - \mathbf{s}_t||_{M_s} = 0, \tag{7}$$

$$\mathbf{s}_{t+1} = T_b^p(\mathbf{s}_{t-1}, \boldsymbol{a}_t^p, \mathbf{S}_b^p), \boldsymbol{a}_t^p = f_m(\pi_\alpha(\mathbf{s}_t)), \tag{8}$$

where the dynamics transition and parameters in the physical system are different from those in the simulated environment, $T_b^p \neq T_b^v$ and $\mathbf{S}_b^v \neq \mathbf{S}_b^p$. $\mathbf{s}_d$ denotes the desired state.

III. Methodology

The pipeline of learning a robust policy for inverted control of MBRs is illustrated in Fig. 3, which incorporates three core stages: (1) 3D simulation environment creation, (2) Physics-informed domain randomization strategy design, and (3) TD3 with multi-buffer and clipping.

A. Simulation Environment

As illustrated in Fig. 4, Unity was adopted as the simulation platform to implement the MBR dynamics and construct the policy training environment. The Rigidbody component was used to reproduce the dynamics. Custom force and torque terms were implemented via the APIs AddForceAtPosition and AddRelativeTorque, including aerodynamic drag $\mathbf{\Gamma}^d$, restoring force and torque $\mathbf{\Gamma}^{gb}$, and added-mass and added-inertia effects ($\mathbf{M}_a$, $\mathbf{C}_a$). Model parameters were identified following [22], [23]. To improve suitability for inverted control training, three enhancements were introduced. First, a refined motor model was developed using experimental data and calibrated according to [24]:

$$F_m = g_m(-0.0292\eta^2 + 0.1118\eta - 0.0039), \tag{9}$$

where g_m is the motor gain and $\eta \in [0, 1]$ the control input. Varying g_m enables simulation of actuator variability.

Second, the simulated MBR structure was modified (Fig. 5) by decomposing the total additional mass into two components, m_{w_1} and m_{w_2}, facilitating inverted control training. Third, a Python-based learning node was implemented to manage training interaction. It supports episode reset via the command R_e and online configuration of MBR parameters $\mathbf{S}_b^{'}$. The variables $\mathbf{S}_b$ and $\mathbf{s}_t$ denote the real-time system parameters and state, respectively.

B. Physics-informed Domain Randomization

As analyzed in Section II-A, the distances between the three center points (c_b, c_g, and c_t) play a dominant role in the dynamics of the MBR. Accordingly, the proposed domain randomization strategy perturbs these distances while preserving physical consistency. As shown in Fig. 5, the resultant gravitational force acts at c_g. The distance between c_g and c_t is

$$h_g = \frac{(m_{bat} + m_{w2})h_t + (m_e + m_h)r_t^b + m_{w1}(2h_e + h_t)}{m_{rb}}, \tag{10}$$

where the total mass $m_{rb} = m_t + m_{bat} + m_e + m_h + m_{w1} + m_{w2}$. m_t and m_{bat} are the weights of the gondola and the battery, respectively. $m_w = m_{w1} + m_{w2}$ is the extra weight to ensure that the MBR is in a neutrally buoyant state. m_e is the weight of the envelope in the deflated state, while m_h is the weight of the filled helium, calculated by $m_h = \rho_h V$, where ρ_h is the density of helium and V is the total volume of the inflated envelope. The distance between c_g and c_b is $r_z^b = r_t^b - h_g$, where $r_t^b = h_t + h_e$. h_t and h_e are half-heights of the gondola and the inflated envelope, respectively. Denote $m_{w1} = \lambda m_w$ and $m_{w2} = (1-\lambda)m_w$; h_g can be expressed in a more simplified way as

$$h_g = \frac{d_m + m_w(h_t + 2\lambda h_e)}{m}, \tag{11}$$

where $d_m = m_{bat}h_t + (m_e + m_h)r_z^b$. Adjusting m_w and λ can modify the distances between these center points. The key distinction is that only varying λ allows m_w to remain constant while altering h_g.

C. TD3 with Multi-buffer and Clipping

TD3 [25] consists of two interrelated processes: environment interaction and policy optimization. During environment interaction, trajectories of the MBR are sampled under different actions, and the outcome of each action is evaluated through a corresponding reward signal. As described in Algorithm 1, N replay buffers are constructed to store MBR trajectories generated under different values of λ.

Once the replay buffers are sufficiently populated, a separate training thread is initiated to update the policy, as detailed in Algorithm 2. The overall training framework follows the standard TD3 architecture and incorporates gradient clipping operations (c_α and c_β), adopted from PPO [26], to further improve training stability. Instead of updating the policy using a single replay buffer, the proposed method leverages N distinct replay buffers, each containing trajectories generated under different MBR dynamic configurations. This multi-buffer training strategy encourages the policy to learn more generalized features, thereby improving its robustness across a wide range of dynamic conditions.

The state representation of the MBR consists of the rotation matrix $\mathbf{R}$ and the angular velocity vector $\boldsymbol{\omega}$. The action space is defined as the desired control torques about the three rotational axes. These torques are subsequently mapped to motor commands through the functional module

Algorithm 1: Environmental Interaction with Domain Randomization

Data: Behavior policies $\pi_\#$, N replay buffers $\mathcal{B}_1, \ldots, \mathcal{B}_N$, running time of each episode t_e, maximum episode N_e, gaussian noise parameters σ, ξ and n_σ, N variable parameters $\lambda_1, \ldots, \lambda_N$, $k = 0$

Result: N replay buffers with experiences

```
for i = 1 to N_e do
    if i mod n_σ = 0 then
        σ ← ξσ
    t_s ← GetCurrentSystemTime
    ψ^i ← U(−ψ_0, ψ_0)
    UpdateBlimpParameter(λ_k, ψ^i)
    while True do
        ε_t^k ~ N(0, σ^2)
        Observe state s_t^k, select action
          a_t^k = π_#^t(s_t^k) + ε_t^k
        φ_t^k, θ_t^k ← GetAttitudeAngles(s_t^k)
        η_t^k ← F_cdl(a_t^k, φ_t^k, θ_t^k)
        Execute η_t^k, observe reward r_t^k, next state
          s_{t+1}^k, done flag d_t^k
        Store (s_t^k, a_t^k, r_t^k, s_{t+1}^k, d_t^k) in B_k
        k ← k + 1
        if k == N then
            k = 0
        t_c ← GetCurrentSystemTime
        if t_c − t_s ≤ t_e or over range then
            break
```

F_{cdl} shown in Fig. 3, implemented following the method described in [24].

The reward function comprises three components: an orientation reward r_{rot}, an angular velocity cost r_ω, and an action cost r_a, which can be expressed as

$$r_{total} = r_{rot} + r_\omega + r_a, \tag{12}$$

where $r_\omega = -\frac{\sum_{i\in\{x,y,z\}} g_{\omega_i}|\omega_i|}{\omega_{max}}$. The parameter g_{ω_i} represents the weight assigned to each channel, and ω_{max} denotes the maximal angular velocity. The action cost is denoted by $r_a = -\sum_{i\in\{x,y,z\}} g_{ai}|\tau_i|$, where g_{ai} is the parameter to shape the reward, and τ_i represents the expected torque of the i-th axis. The orientation reward is defined as

$$\begin{aligned} r_{rot} = \exp(-\text{clip}(g_\phi|e_\phi| + g_\theta|e_\theta| + g_\psi|e_\psi|, 0, g_n)) \\ + \mathbb{I}_{\{\varphi<\zeta\}} \cdot \left(1 - \frac{\varphi}{\zeta}\right), \end{aligned} \tag{13}$$

where $e_\phi = \frac{\varphi \mathbf{v}_x}{\pi}$, $e_\theta = \frac{\varphi \mathbf{v}_y}{\pi}$ and $e_\psi = \frac{\varphi \mathbf{v}_z}{\pi}$. The pair $(\mathbf{v}, \varphi)$ represents the axis-angle parameterization of the orientation error $\mathbf{R_e} = \mathbf{R}^T\mathbf{R}_d$, where $\mathbf{R}_d = \text{diag}(1, -1, -1)$. The rotation angle φ is calculated by $\varphi = \arccos\left(\min(\max(\frac{tr(\mathbf{R}_e)-1}{2}, -1), 1)\right)$, and the rotation axis is

Algorithm 2: TD3 with Multi-Buffer and Clipping

Data: N replay buffers $\mathcal{B}_1, \ldots, \mathcal{B}_N$, interval of policy delay update d_p, discount γ, target network update rate ϖ, thresholds for gradient clipping c_α and c_β, actor π_α, critics Q_{β_1}, Q_{β_2}, target networks $\pi_{\alpha'}, Q_{\beta'_1}, Q_{\beta'_2}$

Result: Learned policy π_α

while *True* **do**

  for *buffer* $k = 1$ **to** N **do**

    Sample batch $(s_i^k, a_i^k, r_i^k, s_{i+1}^k, d_i^k) \sim \mathcal{B}_k$

  Critics Update: Aggregate target Q-values: $Q_\mathrm{d} = \frac{1}{N}\sum_{k=1}^{N}\left[r_i^k + \gamma(1-d_i^k)\min_{j=1,2} Q_{\beta'_j}(s_{i+1}^k, \pi_{\alpha'}(s_{i+1}^k))\right]$

  Update gradients $(j = 1, 2)$: $\nabla_{\beta_j} = \frac{1}{N}\sum_{k=1}^{N}\sum_i (Q_{\beta_j}(s_i^k, a_i^k) - Q_\mathrm{d})^2$

  Clip gradients: clip$(\nabla_{\beta_j}, -c_\beta, c_\beta)$

  Delayed Actor Update:

  if $t \bmod d_p = 0$ **then**

    Update gradients π_α: $\nabla_\alpha = \frac{1}{N}\sum_{k=1}^{N}\sum_i Q_{\beta_1}(s_i^k, \pi_\alpha(s_i^k))$

    Clip gradients: clip$(\nabla_\alpha, -c_\alpha, c_\alpha)$

  Target Network Updates:

  Soft update: $\beta'_j \leftarrow \varpi\beta_j + (1-\varpi)\beta'_j$ for $j = 1, 2$

  Soft update: $\alpha' \leftarrow \varpi\alpha + (1-\varpi)\alpha'$

TABLE I: Parameters in the reward function

g_{ω_x}	g_{ω_y}	g_{ω_z}	g_{a_x}	g_{a_y}	g_{a_z}	g_ϕ	g_θ	g_ψ	ζ	g_n
0.01	0.01	0.01	0.001	0.001	0.001	5.0	5.0	0.5	0.1	10

given by: $\mathbf{v} = \frac{1}{2\sin\varphi}\begin{bmatrix}\mathbf{R}_{e32} - \mathbf{R}_{e23}\\ \mathbf{R}_{e13} - \mathbf{R}_{e31}\\ \mathbf{R}_{e21} - \mathbf{R}_{e12}\end{bmatrix}$. The symbol $\mathbb{I}$ denotes the indicator function where the value is equal to 1 if the condition is true, 0 otherwise. The parameter ζ represents the threshold for the precision bonus.

D. Implementation Details

Both the policy π_α and the action-value function Q_β were approximated using fully connected neural networks with two hidden layers of 256 neurons each. All hidden layers employ Leaky ReLU activation functions, while the output layer of the policy uses a hyperbolic tangent (Tanh) activation. The reward function parameters are summarized in Table I. Higher weights are assigned to ϕ and θ deviations than to ψ, reflecting the priority of inverted pose stabilization. The desired behavior for the MBR is to rapidly reach and maintain the inverted pose, and regulate the ψ to zero, while minimizing energy consumption. The hyperparameters used in Algorithms 1 and 2 are listed in Table II.

The desired MBR behavior is to rapidly reach and maintain an inverted pose, followed by adjusting the ψ to zero. At the same time, energy consumption must be minimized, as the reward function includes a penalty for action.

The parameters and their corresponding values used in Algorithms 1 and 2 are provided in Table II. Ten buffers were used to store experiences sampled across different values of λ, which were constrained to the interval $[0.6, 1.0]$, given the MBR parameters outlined in [24].

TABLE II: Parameters in policy training

N	t_e	N_e	σ	ξ	n_σ	d_p	γ	ϖ	c_a	c_b	λ	ψ_0
10	30s	500	0.15	0.95	100	2	0.98	0.01	0.1	0.1	[0.6, 1]	0.5

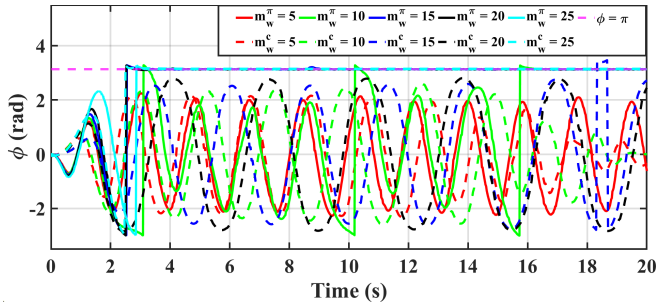

Fig. 6: Roll angle variations with changes in m_w. Solid lines represent the learned policy, while the dotted line represents the baseline controller.

IV. Evaluation and Experiment

To evaluate the performance of the learned policy, the parameters m_w, λ, and g_m were varied in the inverted pose stabilization task. The energy-shaping controller proposed in [17] was adopted as the baseline for comparison. Its control gains were tuned under nominal conditions of $m_w = 23.35$ g, $\lambda = 1$, and $g_m = 1.7$. The objective was to drive the ϕ from 0 to π, while maintaining both θ and ψ at zero.

A. Performance of the Learned Policy to m_w Variations

Table III summarizes the results obtained by varying the m_w, with $\lambda = 1$ and $g_m = 1.7$ held constant.

TABLE III: Comparison of the baseline controller and the learned policy with varied m_w

Method	$m_w = 5g$	$m_w = 10g$	$m_w = 15g$	$m_w = 20g$	$m_w = 25g$
Baseline	×	×	×	×	✓
Our policy	×	✓	✓	✓	✓

The parameter m_w was varied from 5 g to 25 g, covering conditions from buoyancy-dominant (buoyancy > gravity) to gravity-dominant (buoyancy < gravity). When $m_w = 5$ g, neither the learned policy nor the baseline controller completed the task. In this case, c_g is too close to c_t, preventing the generation of sufficient rotational moment to invert the MBR under the fixed motor gain $g_m = 1.7$. For all other values of m_w, the learned policy successfully achieved the inverted pose, whereas the baseline controller was effective only at $m_w = 25$ g, its nominal tuning condition. These results suggest that the baseline controllers are sensitive to parameter variations, while the learned policy exhibits stronger robustness across different dynamic configurations.

TABLE IV: Comparison of the baseline controller and the learned policy with varied λ

Method	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.9$	$\lambda = 1.0$
Baseline	×	×	×	×	✓
Our policy	✓	✓	✓	✓	✓

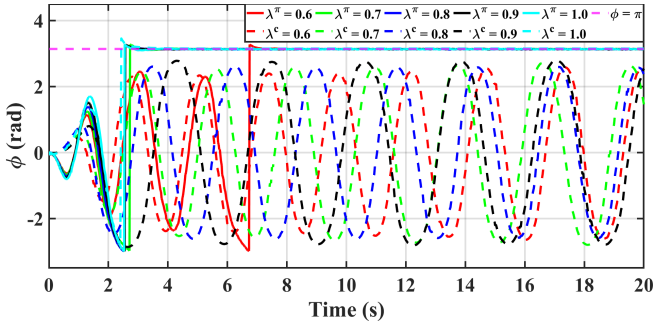

Fig. 7: Roll angle variations with changes in λ.

TABLE V: Comparison of the baseline controller and the learned policy with varied g_m

Method	$g_m = 0.5$	$g_m = 1.0$	$g_m = 1.5$	$g_m = 2.0$	$g_m = 2.5$
Baseline	×	✓	✓	✓	✓
Our policy	×	✓	✓	✓	✓

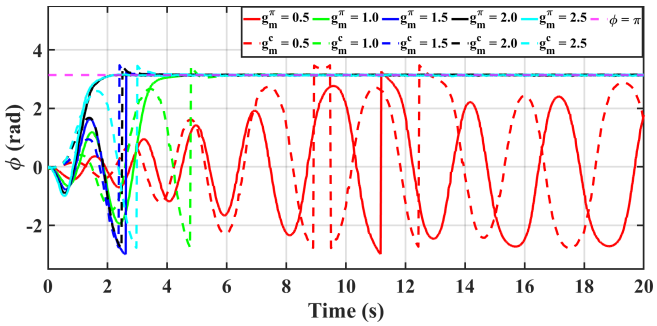

Fig. 8: Roll angle variations with changes in g_m.

The ϕ responses are shown in Fig. 6, where m_w^π and m_w^c denote the policy and baseline controller, respectively.

As m_w increases, the maximum achievable ϕ increases and eventually reaches π. The neutrally buoyant weight is approximately 23.35 g; thus, at $m_w = 25$ g, gravity exceeds buoyancy. In this regime, c_g moves closer to c_b, reducing the restoring moment and allowing a larger achievable ϕ.

B. Performance of the Learned Policy to λ Variations

Varying λ determines the position of c_g while maintaining the MBR in a neutrally buoyant state. In this experiment, the extra weight and motor gain were set to $m_w = 23.35$ g and $g_m = 1.7$, respectively. Table IV presents the results for both the baseline controller and the learned policy, indicating that the baseline controller was only successful when $\lambda = 1.0$.

The trained control policy completes the task for all $\lambda \in [0.6, 1.0]$, demonstrating its robustness against variations in MBR parameters. The variations in the roll angle under these conditions are depicted in Fig. 7, in which the parameter for the policy is denoted as λ^π and that for the controller as λ^c. Unlike the variation in m_w, changes in λ do not affect the magnitudes of buoyancy and gravity, but only influence the position of c_g. As λ increases, c_g moves closer to c_b. This results in larger maximum achievable roll angles (before reaching $\phi = \pi$) and reduces the time required to achieve the inverted state. When $\lambda = 1.0$, the controller completed the task more quickly, as its parameters were specifically fine-tuned for this configuration.

C. Performance of the Learned Policy to g_m Variations

To verify that the learned policy functions effectively across different motors, m_w and λ are configured to 23.35 g and 1.0, respectively. The results are shown in Table V.

Only when $g_m = 0.5$ did both the learned policy and the baseline controller fail. Although each method could temporarily drive the MBR to the inverted pose, the motor thrust was insufficient to maintain it. The controller achieved inversion at approximately 9 s, while the policy did so at around 11 s, but both subsequently lost stability. For all other tested values of g_m, both the baseline controller and the learned policy successfully completed the task. However, their control behaviors differ noticeably, as illustrated in Fig. 8. The results indicate that as g_m increases, the policy requires less time to complete the task, owing to the corresponding increase in total force. When $g_m \geq 2.0$, only two rotations are necessary, a result consistent with the controller method. In contrast, for the controller, a higher g_m results in a larger maximum achievable roll angle, which in turn requires more time to complete the task.

TABLE VI: Comparison of the baseline controller and the learned policy with varied m_w, λ, and g_m

Test Case	C1	C2	C3	C4	C5
m_w (g)	15	15	20	25	25
λ	0.8	1.0	0.9	0.7	0.8
g_m	1.7	1.0	1.6	1.5	1.4
Baseline	×	×	×	×	×
Our Policy	✓	✓	✓	✓	✓

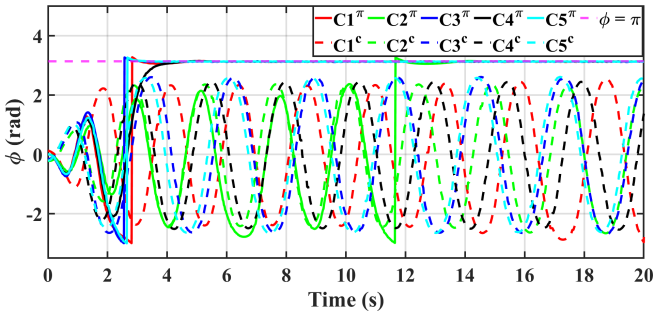

Fig. 9: Roll angle variations with changes in m_w, λ and g_m.

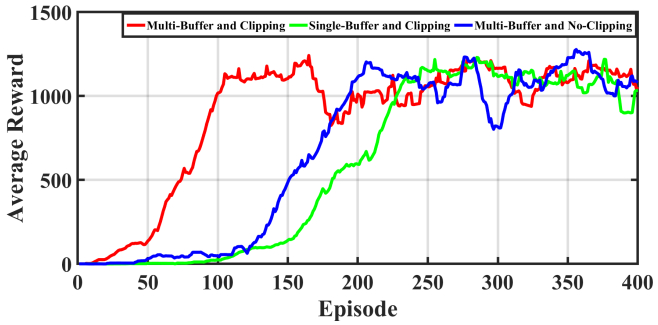

Fig. 10: Comparison of the average reward across three cases, computed using a moving average with a window size of 19.

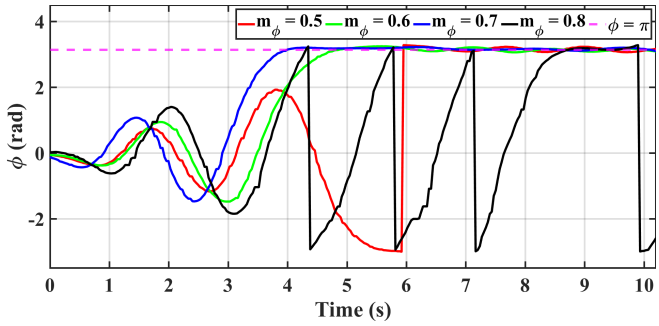

Fig. 11: Roll angle variations of the MBR with different m_ϕ.

D. Performance of the Learned Policy to m_w, λ and g_m Variations

The parameters m_w, λ, and g_m were varied simultaneously to further evaluate the robustness of the policy. The configurations and results are summarized in Table VI. The learned policy succeeded in all cases, whereas the baseline controller failed under these combined variations. The ϕ responses are shown in Fig. 9. In Case 2 (C2), the maneuver required a longer completion time due to the low motor gain g_m, which limited the available actuation capability.

E. Ablation Study

To evaluate the contribution of the multi-buffer strategy and gradient clipping to training stability, an ablation study was conducted. The results are shown in Fig. 10. The proposed method, which combines multi-buffer experience storage with gradient clipping, converged within approximately 100 episodes. Removing gradient clipping while retaining the multi-buffer structure increased the convergence time to nearly 200 episodes. In contrast, using a single buffer with gradient clipping required at least 250 episodes—2.5 times slower than the proposed approach. For fairness, the capacity of the single buffer was set equal to the total capacity of all buffers in the multi-buffer configuration. These results demonstrate that the combination of multi-buffer sampling and gradient clipping significantly improves training stability and sample efficiency. The average return exhibits persistent fluctuations due to the continuous injection of exploration noise ϵ_t^k throughout training.

F. Policy Deployment in a Physical MBR

We transfer the learned policy to the real platform with only minimal parameter adjustments, avoiding additional training on physical data. As shown in Fig. 3, a mapping layer is introduced to mitigate the sim-to-real discrepancy during the inverted transition:

$$\tau^p = \mathbf{M}_0 \boldsymbol{a}^p, \quad \Delta\phi < \varrho, \tag{14}$$

where $\tau^p = [\tau_x, \tau_y, \tau_z]^T$ denotes the physical torque command. The term $\Delta_\phi = \pi - \phi$ represents the roll angle deviation, and ϱ is the switching threshold. In experiments, $\mathbf{M}_0 = \mathrm{diag}(m\phi, m_\theta, m_\psi)$ with $\varrho = 0.8$. The parameters m_θ and m_ψ are fixed at 0.1, while m_ϕ varies from 0.5 to 0.8. The results are shown in Fig. 11. The learned policy drives the MBR to the inverted pose, after which a PD controller stabilizes the system once angular velocities approach zero. The transition sequence for $m_\phi = 0.7$ is illustrated in Fig. 12. Among the tested values, only $m_\phi = 0.8$ failed. These results indicate that the proposed mapping layer effectively bridges the sim-to-real gap without policy retraining. Using $m_\phi = 0.7$, additional physical experiments were conducted by varying m_{w_1} and m_{w_2} (Table VII). The MBR successfully achieved inversion in all cases (Fig. 13). Increasing m_{w_1} shifts the c_g toward the c_b, reducing transition time, whereas increasing m_{w_2} moves c_g toward the c_t, prolonging the maneuver. These observations are consistent with the simulation results in Sections IV-A and IV-B, further validating the effectiveness of the proposed method.

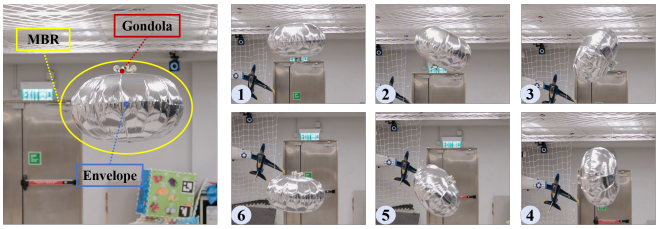

Fig. 12: (a) Experimental setup; (b) Visualization of the action sequence of an MBR for achieving an inverted pose.

TABLE VII: Configuration of the extra weights in the physical experiment

Weights	MBR_1	MBR_2	MBR_3	MBR_4	MBR_5
$m_{w1}(g)$	25	26.07	25	27.59	25
$m_{w2}(g)$	0	0	1.07	0	2.59

V. Conclusion

This paper proposes a new DRL-based method for inverted control of MBR, aiming to achieve its full agility. The method involves the construction of a virtual training environment, policy training using domain randomization, an improved TD3 method, and policy deployment via a designed mapping layer. Compared to the energy-shaping

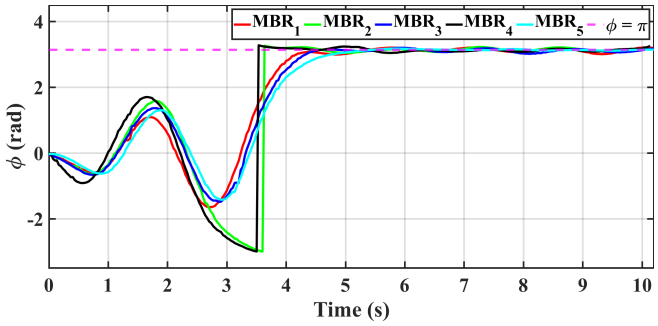

Fig. 13: Roll angle variation of the MBR with different m_w.

controller, the learned policy achieves a higher success rate across diverse scenarios. Although the mapping layer designed for policy deployment enables the policy to function in physical settings without further training, it constrains the performance of the learned policy. This indicates that a linear relationship alone cannot fully bridge the sim-to-real gap. Therefore, analyzing and quantifying the sim-to-real gap in inverted control remains an open problem for future work.

REFERENCES

[1] M. Wang, Q. Wang, Z. Wang, Y. Gao, J. Wang, C. Cui, Y. Li, Z. Ding, K. Wang, C. Xu, *et al.*, "Unlocking aerobatic potential of quadcopters: Autonomous freestyle flight generation and execution," *Science Robotics*, vol. 10, no. 101, p. eadp9905, 2025.

[2] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.

[3] Y. Xie, M. Lu, R. Peng, and P. Lu, "Learning agile flights through narrow gaps with varying angles using onboard sensing," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5424–5431, 2023.

[4] Z. Han, X. Huang, Z. Xu, J. Zhang, Y. Wu, M. Wang, T. Wu, and F. Gao, "Reactive aerobatic flight via reinforcement learning," *arXiv preprint arXiv:2505.24396*, 2025.

[5] S. Oh, S. Kang, K. Lee, S. Ahn, and E. Kim, "Flying display: Autonomous blimp with real-time visual tracking and image projection," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 131–136.

[6] H. Han and B. D. Song, "The flying warehouse delivery system with multi-commodity inventory management," *Available at SSRN 4988309*, 2024.

[7] D. Chaitanya, M. N. Reddy, V. M. Shaju, P. J. Bharadwaz, B. S. Jayanth, and M. Bukya, "Slam-enabled autonomous blimp for uav applications," in *2025 International Conference on Next Generation Communication & Information Processing (INCIP)*. IEEE, 2025, pp. 1034–1039.

[8] Y. Nitta, S. Inai, K. Matsumura, M. Ishida, T. Onai, and A. Nishitani, "The visual inspection methodology for ceiling utilizing the blimp," *Procedia Engineering*, vol. 188, pp. 256–262, 2017.

[9] Y. Zhang, J. Yan, X. Fan, S. Wang, X. Wang, W. Huang, and Z. Zhao, "A novel miniature omnidirectional multi-rotor blimp," in *2023 IEEE 18th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2023, pp. 1568–1573.

[10] S. Ye, J. Zhang, Q. Lu, Y. Xiao, J. Yuan, and S. Hu, "Design and simulation of a bio-inspired rigid-soft hybrid robotic blimp," in *2022 International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, 2022, pp. 599–604.

[11] J. Xu, D. S. D'antonio, D. J. Ammirato, and D. Saldaña, "Sblimp: Design, model, and translational motion control for a swing-blimp," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 6977–6982.

[12] Q. Tao, J. Wang, Z. Xu, T. X. Lin, Y. Yuan, and F. Zhang, "Swing-reducing flight control system for an underactuated indoor miniature autonomous blimp," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 1895–1904, 2021.

[13] S. Cho, V. Mishra, Q. Tao, P. Vamell, M. King-Smith, A. Muni, W. Smallwood, and F. Zhang, "Autopilot design for a class of miniature autonomous blimps," in *2017 IEEE conference on control technology and applications (CCTA)*. IEEE, 2017, pp. 841–846.

[14] H. Cheng, Z. Sha, Y. Zhu, and F. Zhang, "Rgblimp: robotic gliding blimp-design, modeling, development, and aerodynamics analysis," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7273–7280, 2023.

[15] J. Xu, T. Vu, D. S. D'Antonio, and D. Saldaña, "Mochiswarm: A testbed for robotic blimps in realistic environments," *arXiv preprint arXiv:2503.03077*, 2025.

[16] M. Pellegrino, M. Barciś, J. Simonjan, Zulkarnaen, C. F. Chiasserini, and E. Natalizio, "Tinyblimp: A promising frontier for autonomous miniature unmanned aerial vehicles," in *Proceedings of the 10th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2024, pp. 1–6.

[17] J. Wang and F. Zhang, "Achieving and maintaining inverted pose for miniature autonomous blimps," in *2024 American Control Conference (ACC)*. IEEE, 2024, pp. 338–343.

[18] Y. T. Liu, E. Price, M. J. Black, and A. Ahmad, "Deep residual reinforcement learning based autonomous blimp control," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 12 566–12 573.

[19] Y. Zuo, Y. T. Liu, and A. Ahmad, "Autonomous blimp control via h_∞ robust deep residual reinforcement learning," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023, pp. 1–8.

[20] Unity Technologies, "Unity: Real-time development platform — 3d, 2d, vr & ar engine." [Online]. Available: https://unity.com

[21] J. Dong, H. Yu, B. Lu, H. Liu, and Y. Fang, "Adaptive output feedback trajectory tracking control of an indoor blimp: Controller design and experiment validation," *IEEE Transactions on Industrial Electronics*, 2024.

[22] Q. Tao, J. Cha, M. Hou, and F. Zhang, "Parameter identification of blimp dynamics through swinging motion," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1186–1191.

[23] Q. Tao, M. Hou, and F. Zhang, "Modeling and identification of coupled translational and rotational motion of underactuated indoor miniature autonomous blimps," in *2020 16th international conference on control, automation, robotics and vision (ICARCV)*. IEEE, 2020, pp. 339–344.

[24] Q. Tao, "Design and control of an indoor miniature autonomous blimp," *Ph. D. dissertation, Georgia Institute of Technology*, 2020.

[25] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

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