Aerial-Balance-Bench: A Controlled and Reproducible Drone-Ball Balancing Benchmark for Indirect Dynamic Aerial Manipulation

Mingjiang Liu[1], Guanzhong Zhou[1], Chengchen Zhang[1], Stephen L. Smith[2], and Hailong Huang[1,*]

***Abstract*—Aerial manipulation studies and benchmarks have mostly emphasized direct interaction and static or quasi-static tasks, leaving indirect dynamic manipulation insufficiently studied. We present Aerial-Balance-Bench, a controlled and reproducible drone-ball balancing benchmark in which a vertically constrained tethered drone tilts a beam to drive a rolling ball to a target position or along a reference trajectory. The benchmark unifies: 1) target-position balancing and trajectory-tracking tasks; 2) thrust-, velocity-, and position-command interfaces with corresponding dynamic models; 3) a Gym-style Isaac Lab environment with standardized observations, actions, and task-dependent rewards; and 4) evaluation metrics and robustness tests. To demonstrate the benchmark and provide reference results, we develop a delay-aware hierarchical control framework for the velocity-command interface, combining delay identification with model-based state prediction to compensate for low-level tracking latency. Within this framework, we compare three high-level baselines: cascaded PID, nonlinear model predictive control (MPC), and model-free reinforcement learning (RL). Simulation and real-world experiments show the benchmark's feasibility. These results establish Aerial-Balance-Bench as a practical benchmark and baseline suite for studying the core balancing dynamics in indirect dynamic aerial manipulation.**

***Index Terms*—Aerial Systems: Mechanics and Control; Aerial Systems: Applications; Aerial Manipulation; Performance Evaluation and Benchmarking.**

I. Introduction

RECENTLY, aerial robots have gained increasing attention in both research and practical applications. Many aerial tasks are conducted at high altitudes, presenting challenges and safety risks for human workers. These scenarios may also require active environmental interaction, necessitating aerial robotic operations that go beyond passive roles. In response, aerial manipulation has emerged as a key research focus, aiming to equip aerial robots with the ability to physically interact with their surroundings [1].

Unmanned aerial vehicles (UAVs), such as multirotors, are commonly equipped with robotic arms or custom grippers to form typical aerial manipulation systems. These systems are capable of performing a wide range of manipulation tasks. For instance, one study addressed disturbances in a quadrotor equipped with a robotic arm to achieve millimeter-level peg-in-hole insertion [2]. To overcome the limited manipulation workspace of underactuated multirotors, omnidirectional platforms have been employed as floating bases, accompanied by whole-body controllers designed for robust manipulation [3], [4]. Additionally, an aerial manipulator incorporating a custom soft tendon-actuated gripper was developed to enable in-flight grasping [5]. Alternatively, objects can also be manipulated by UAVs through direct attachment or cable suspension, where a notable example is cable-suspended payload transportation [6]–[8]. To mitigate the impact effects of suspended payloads on quadrotors, an impact-aware planning and control framework was developed [6]. To enhance payload capacity, multi-quadrotor systems have been developed [7], and an admittance-controller-based system that enables adaptive human-robot interaction during payload transport [8]. More recently, a hand-like autonomous flying robot has been introduced, enabling versatile tasks such as airborne grasping, door opening, perching, object transport, and human interaction [9].

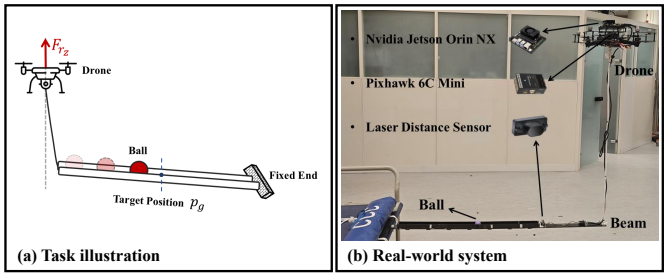

Fig. 1. Illustration of the drone-ball balancing task.

While these studies demonstrate the versatility of aerial manipulation, most existing systems are tightly coupled to specific tasks and hardware configurations. Such task-specific implementations are often difficult to reproduce, which makes fair comparison across methods challenging and slows the development of general algorithmic insights. Benchmark-oriented studies can help address this issue, yet they remain scarce in aerial manipulation. The work in [10] introduced hardware-oriented benchmarks for aerial manipulators equipped with robotic arms, evaluating system performance in terms of accuracy, execution time, manipulation capability, and impact response. RotorTM [11] provided an open-source simulator for aerial transportation and manipulation with single or multiple UAVs connected to payloads through passive mechanisms such as cables or rigid links. More recently, AIR-

Manuscript received March 2, 2026; revised May 29, 2026; accepted June 24, 2026. This paper was recommended for publication by Editor Cosimo Della Santina upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Research Centre for Low Altitude Economy (RCLAE), The Hong Kong Polytechnic University. (*Corresponding author: Hailong Huang.)

[1]Mingjiang Liu, Guanzhong Zhou, Chengchen Zhang, and Hailong Huang are with the Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China (e-mails: mingjiangaae.liu@connect.polyu.hk; newtt.zhou@connect.polyu.hk; chengchen-john.zhang@connect.polyu.hk; hailong.huang@polyu.edu.hk)

[2]Stephen L. Smith is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada (e-mail: stephen.smith@uwaterloo.ca)

Code is available at: https://github.com/Wenminggong/aerial_balance_bench.

Digital Object Identifier (DOI): see top of this page.

TABLE I
COMPARISON OF REPRESENTATIVE AERIAL MANIPULATION STUDIES AND BENCHMARKS RELATED TO THIS LETTER.

Category	Work	System Configuration	Manipulation Mode	Object Dynamics	Simulation	Real System	Task Description
Specific	Aerial Pick-and-Peg [2]; Flying Hand [3]; SE(3) Aerial Manipulation [4]	UAV + robotic arm	Direct	Static	×	✓	Pick-and-Peg; writing; changing light bulbs; grasping; pulling
	Soft Aerial Gripper [5]	UAV + soft gripper	Direct	Static/Dynamic	×	✓	High-speed aerial grasping
	IMPACTOR [6]; Cooperative Transportation [7]; Human-Aware Transportation [8]	UAV(s) + cable-suspended payload	Direct	Dynamically attached	✓	✓	Payload transportation
	HI-ARM [9]	Hand-like UAV	Direct	Static	×	✓	Grasping; door opening; perching
Benchmark	Aerial Manipulation Benchmarks [10]	UAV + robotic arm	Direct	Static	×	✓	Grasping; force control
	RotorTM [11]	UAV(s) + cable/rigid-link payload	Direct	Dynamically attached	✓	✓	Payload transportation
	AIR-VLA [12]	UAV + robotic arm	Direct	Static	✓	×	Picking; placing; twisting
	Aerial-Balance-Bench (ours)	UAV + cable-suspended beam	Indirect	Dynamic	✓	✓	Ball balancing

VLA [12] introduced a benchmark for vision-language-action systems in aerial manipulation, providing a physics-based simulation environment and a multimodal dataset. Nevertheless, existing efforts mainly target direct interaction through arms/grippers or payloads physically attached to aerial robots. In contrast, many real-world scenarios require aerial robots to manipulate dynamic objects indirectly through an intermediate tool or carrier, which remains insufficiently studied.

To fill this gap, inspired by the canonical ball-and-beam system designed for studying nonlinear control theory and developing advanced control algorithms [13], [14], we propose Aerial-Balance-Bench, a drone-ball balancing benchmark that serves as a controlled and simplified testbed for dynamic manipulation through indirect actuation. In this benchmark, a vertically constrained tethered drone is required to manipulate a beam to guide a ball either to a target position or along a desired trajectory (see Fig. 1). Unlike the canonical ball-and-beam system, where the beam is driven by a directly controlled actuator, the proposed task keeps the drone in the control loop. Consequently, the beam motion is shaped by the drone response dynamics, low-level tracking errors, and actuator limits, thereby preserving key nonidealities of practical aerial actuation. Owing to the inherent aerial instability and highly dynamic motion of the ball, precise ball balancing or trajectory tracking poses a unique challenge. The proposed benchmark therefore provides a focused platform for systematically studying and comparing methods for indirect dynamic aerial manipulation under controlled physical assumptions.

To make this task a practical and reusable benchmark, we formalize Aerial-Balance-Bench from the perspectives of task definition, control abstraction, environment design, and evaluation. Specifically, we define the target-position balancing and trajectory-tracking tasks, and introduce three control interfaces: thrust, velocity, and position command, together with the corresponding dynamic models. We further specify a unified observation and action design, task-dependent reward definitions for RL training, and standardized evaluation and robustness protocols.

Building on this benchmark, we develop a reference control framework for the target-position balancing task based on the velocity-command interface. The velocity-command interface preserves a practical separation between high-level balancing and low-level flight control, but it also introduces action delay because the commanded velocity must be tracked by the onboard controller. We therefore equip the reference framework with delay-aware components, including model-based state prediction and delay identification, to explicitly compensate for the latency induced by the hierarchical architecture.

Finally, to establish meaningful performance references for future studies, we design and implement three representative high-level baselines: cascaded PID, nonlinear MPC, and model-free RL. These controllers represent classical feedback control, optimization-based control, and learning-based control, respectively, and are instantiated under the same reference framework. We evaluate them in both simulation and on a physical system for the target-position balancing task, enabling a systematic comparison of their accuracy, robustness, and practical deployability.

In short, this letter makes the following key contributions:

- We introduce the drone-ball balancing task as a benchmark for studying indirect dynamic aerial manipulation under a controlled vertical-motion setup, capturing the combined challenges of underactuated flight dynamics, tool-mediated interaction, and dynamic object manipulation. Table I compares this benchmark against representative aerial manipulation studies and benchmarks.
- We provide a complete and configurable benchmark specification, including two task families, three control interfaces, a unified environment specification, task-dependent evaluation metrics, and a robustness test suite. These elements make Aerial-Balance-Bench controlled, reproducible, and extensible for studying indirect dynamic aerial manipulation.
- We propose a reference control framework for the target-position balancing task based on the velocity-command interface and augment it with a delay-aware design, thereby mitigating the adverse effects of action delay in hierarchical aerial manipulation.
- We implement three representative baselines and compare them systematically in simulation and on a real platform, providing reference results.

II. AERIAL-BALANCE-BENCH

A. Overview and Design Goals

To broaden aerial manipulation benchmarks, we introduce Aerial-Balance-Bench. Unlike conventional aerial manipulation settings that emphasize direct interaction through robotic arms or grippers, the proposed task requires the drone to regulate the motion of a ball through an intermediate beam. This setup provides a representative testbed for studying aerial manipulation beyond direct and static settings. An overview of the benchmark is shown in Fig. 2.

The benchmark is designed around two primary principles. The first is controllability and reproducibility, so that different control and learning algorithms can be evaluated under the same task definitions, initial conditions, and environmental variations. The second is to approximate real-world deployment as closely as possible, so that conclusions obtained in

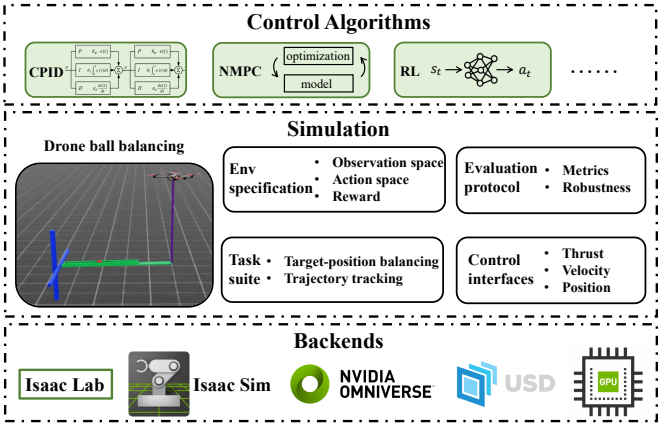

Fig. 2. Overview of Aerial-Balance-Bench.

simulation remain informative for physical systems. To follow these principles, we build the benchmark on Isaac Lab [15] and provide a Gym-style interface [16] for algorithm development and evaluation. The environment is specified through modular configuration files, which define the task setup, system parameters, and testing conditions in a transparent and repeatable manner, thereby enabling controlled and reproducible experiments. At the same time, the simulator incorporates the drone's low-level control and actuation process, making the environment better aligned with practical aerial platforms and more suitable for sim-to-real transfer. In addition, the implementation supports GPU-parallelized training and evaluation, which allows efficient large-scale testing across different tasks, controller designs, and robustness settings.

B. Benchmark Definition

1) Physical Setup and Assumptions: The proposed drone-ball balancing task is illustrated in Fig. 1(a), where a drone manipulates a beam through coordinated tilting to drive the ball to a target position or track a desired trajectory. The system has three main features: 1) the drone is connected to one end of the beam by a rope; 2) the drone is constrained to move only along the vertical axis; and 3) the opposite end of the beam is fixed during normal operation but can also be vertically actuated to introduce disturbances. The vertical-motion constraint decouples lateral drone translation from beam inclination, thereby isolating the core dynamic interaction between drone actuation and ball motion. Although this setting does not cover arbitrary 3-D flight, it retains the vertical actuation channel that directly determines beam inclination and ball rolling. Thus, the resulting controllers and insights can inform the balancing layer of unconstrained aerial systems.

2) Task Suite: To comprehensively evaluate the drone's capability for manipulating the ball, we define two complementary tasks in the benchmark: *Target-position Balancing* and *Trajectory Tracking*. These two tasks correspond to a static regulation problem and a dynamic tracking problem, respectively, and therefore impose different requirements on the controller.

a) Target-position Balancing: In this task, the objective is to drive the ball from an initial position to a given target position on the beam as quickly and accurately as possible. Let $p_b(t)$ denote the ball position and p_g denote the desired target position. The control objective can be written as

$$\lim_{t\to\infty} |p_b(t) - p_g| = 0. \tag{1}$$

In the benchmark, the initial ball position and the target position are randomly sampled for each episode, and the controller is evaluated over a fixed control horizon. This task primarily assesses the controller's regulation capability under different initial-goal pairs, including its transient response and final stabilization performance.

b) Trajectory Tracking: In this task, the controller is required to balance the ball along a time-varying reference trajectory throughout the entire evaluation horizon. Let $p_g(t)$ denote the time-varying reference ball position, and define the control goal as minimizing the episode-level tracking error. Specifically, the tracking objective is written as

$$\min J = \int_{t=0}^{T} |p_b(t) - p_g(t)|\, dt. \tag{2}$$

We set a constant initial ball position. Three types of reference trajectories are considered in the benchmark: sinusoidal, trapezoidal, and triangular trajectories. These three trajectory families introduce distinct tracking challenges, including smooth periodic motion, abrupt reference changes, and piecewise linear variations, thereby enabling a more comprehensive evaluation of dynamic tracking performance.

TABLE II
NOTATION USED IN THE DYNAMIC MODELS.

Notation	Description	Notation	Description
p_b	Ball position	θ	Beam angle w.r.t. horizontal
J_b	Ball inertia	β	Rope angle w.r.t. vertical
m_b	Ball mass	m_r	Drone mass
r_b	Ball radius	τ_r	Rope-induced torque on the beam
J_b'	Beam inertia	F_{r_z}	Drone vertical thrust
m_b'	Beam mass	$F_{\max}$	Max. drone vertical thrust
g	Gravity	v_{r_z}	Drone vertical velocity
L	Beam length	d_{r_z}	Drone vertical displacement
l	Rope length	$a_{\max}$	Max. drone vertical acceleration

3) Control Interfaces and Dynamic Models: The drone's motion serves as the control input of the ball-balancing system. However, how this motion is commanded can be abstracted at different control layers, which leads to different trade-offs between responsiveness and controller complexity. A lower-level interface can provide more direct and timely actuation over the beam dynamics, but it also couples the ball-balancing policy more tightly with low-level flight control. In contrast, a higher-level interface simplifies controller design by further decoupling the benchmark task from the drone's low-level controller, but its effective behavior depends more strongly on tracking dynamics and may therefore introduce additional delay. To support a broad range of controller designs, the benchmark provides three control interfaces: thrust command, velocity command, and position command. Under the assumptions of nominal system mentioned in Section II-B1, the rope geometry satisfies

$$\sin\beta = \frac{L}{l}(1 - \cos\theta). \tag{3}$$

Below we summarize the corresponding core dynamic models. The model parameters are listed in Table II.

a) Thrust-command Interface: In this interface, the system input is the drone's vertical thrust F_{r_z}. During execution, this command is converted into the desired vertical acceleration and attitude, then tracked by a low-level SE(3) attitude controller [17]. This interface offers the most direct actuation and fastest response, but it also yields the strongest coupling between ball balancing and low-level flight control. The corresponding core model can be written as

$$(\frac{J_b}{r_b^2}+m_b)\ddot{p_b} = m_b(p_b-L)\dot{\theta}^2 - m_b g\sin\theta, \tag{4a}$$

$$\begin{aligned}[J_b'+m_b(L-p_b)^2]\ddot{\theta} =& (m_b g\cos\theta + 2m_b\dot{\theta}\dot{p_b})(L-p_b)\\ &+\frac{L}{2}m_b' g\cos\theta - \tau_r,\end{aligned} \tag{4b}$$

$$\begin{aligned}\tau_r =& [F_{r_z} - m_r g + m_r l(\cos\beta\dot{\beta}^2 + \sin\beta\ddot{\beta})\\ &- m_r L(\sin\theta\dot{\theta}^2 - \cos\theta\ddot{\theta})]\frac{\cos(\beta-\theta)}{\cos\beta}L,\end{aligned} \tag{4c}$$

$$0 \le F_{r_z} \le F_{\max}. \tag{4d}$$

Here, the high-level command directly enters the beam dynamics through the rope-induced torque τ_r, so this interface retains the richest coupling with the drone actuation dynamics.

b) Velocity-command Interface: In this interface, the system input is the drone's vertical velocity v_{r_z}, which is tracked by a low-level SE(3) velocity controller [17]. The resulting abstract dynamic model is

$$(\frac{J_b}{r_b^2}+m_b)\ddot{p_b} = m_b(p_b-L)\dot{\theta}^2 - m_b g\sin\theta, \tag{5a}$$

$$L\dot{\theta} = -v_{r_z}\frac{\cos\beta}{\cos(\beta-\theta)}, \tag{5b}$$

$$|\dot{v}_{r_z}| \le a_{\max}. \tag{5c}$$

Compared with direct thrust command, this interface removes most of the system-level complexity from the high-level policy and achieves a compromise between control authority and design simplicity. Hence, the benchmark controller only needs to generate feasible velocity commands, while the low-level controller handles the corresponding velocity tracking. However, this hierarchical architecture will introduce longer control delays.

c) Position-command Interface: In this interface, the system input is the drone's vertical displacement d_{r_z}. This command is tracked by a low-level SE(3) position controller [17]. The abstract dynamic model is written as follows:

$$(\frac{J_b}{r_b^2}+m_b)\ddot{p_b} = m_b(p_b-L)\dot{\theta}^2 - m_b g\sin\theta, \tag{6a}$$

$$d_{r_z} = l(\cos\beta - 1) - L\sin\theta, \tag{6b}$$

$$|\ddot{d}_{r_z}| \le a_{\max}. \tag{6c}$$

This interface provides the highest level of abstraction and has the easiest dynamics formulation. However, its effective response depends on the derivatives of the commanded displacement and on the tracking behavior of the low-level position controller, making it more susceptible to lag than the thrust- and velocity-command interfaces.

C. Environment Specification

1) Observation Space: We use a unified observation space for both *Target-position Balancing* and *Trajectory Tracking*. At control step t_k, the observation is defined as

$$s_k = [p_b, v_b, a_b, \theta, \omega, \alpha, d_{r_z}, v_{r_z}, a_{r_z}, p_g, a_{k-1}]^T, \tag{7}$$

where v_b and a_b denote the ball velocity and acceleration along the beam, ω and α denote the beam angular velocity and angular acceleration, a_{r_z} represents the drone's vertical acceleration, and a_{k-1} represents the action taken at step t_{k-1}. Here, p_g denotes the desired ball position: it is constant in *Target-position Balancing* and is obtained from the reference trajectory in *Trajectory Tracking*.

2) Action Space: The action space is tied to the selected control interface. Instead of directly outputting an absolute control command, the agent outputs an increment of the current command. This design yields a unified interface across different control abstractions and makes it easier to enforce smoothness and actuator feasibility. Let u_k denote the current high-level command and let a_k denote the action at step t_k. The command update is defined as

$$u_k = u_{k-1} + \mathrm{clip}_{\mathcal{A}}(a_k), \tag{8}$$

where $\mathcal{A}$ is the feasible command set associated with the chosen control interface. Accordingly, the action is defined as a relative increment:

$$a_k = \Delta F_{r_z,k}, \quad \Delta F_{r_z,k} \in [-\Delta F_{\max}, \Delta F_{\max}], \tag{9a}$$

$$a_k = \Delta v_{r_z,k}, \quad \Delta v_{r_z,k} \in [-\Delta v_{\max}, \Delta v_{\max}], \tag{9b}$$

$$a_k = \Delta d_{r_z,k}, \quad \Delta d_{r_z,k} \in [-\Delta d_{\max}, \Delta d_{\max}], \tag{9c}$$

for the thrust-command, velocity-command, and position-command interfaces, respectively. Here, the exact bounds are configurable through the environment specification files, which allows the benchmark to support different actuation ranges and difficulty settings while preserving a consistent action semantics.

3) Reward Design: To support RL policy training, we define task-dependent rewards that share a common structure but reflect the different objectives of regulation and tracking. All the reward weights are configurable in the environment files and can therefore be tuned for different training regimes without changing the benchmark definition itself. The reference reward weights used in this letter are provided in the released repository. Detailed definitions are listed below.

a) Target-position Balancing: Let $e_k = p_b(t_k) - p_g(t_k)$ be the ball-position error at time t_k. The reward is defined as

$$r_k^{\mathrm{bal}} = r_{\mathrm{object},k}^{\mathrm{bal}} + r_{\mathrm{control},k}^{\mathrm{bal}} + r_{\mathrm{failure},k}^{\mathrm{bal}} + r_{\mathrm{goal},k}, \tag{10}$$

where

$$r_{\mathrm{object},k}^{\mathrm{bal}} = -k_1 e_k^2 - k_2 v_b^2, \tag{11a}$$

$$r_{\mathrm{control},k}^{\mathrm{bal}} = -k_3 u_k^2 - k_4 a_k^2, \tag{11b}$$

$$r_{\mathrm{failure},k}^{\mathrm{bal}} = \begin{cases} -k_5 & \text{if } (|\theta| > \theta_{\max}) \vee (|e_k| > e_{\max}) \\ 0 & \text{otherwise} \end{cases}, \tag{11c}$$

$$r_{\text{goal},k} = \begin{cases} (c - k_7|e_k|)\exp(-k_6|v_b|) & \text{if } |e_k| < e_{\text{goal}} \\ 0 & \text{otherwise} \end{cases}. \tag{11d}$$

b) Trajectory Tracking: We define a tracking-oriented reward using both position and velocity matching. Let $v_{g,k}$ denote the reference velocity obtained from the desired trajectory at time t_k. The reward is defined as

$$r_k^{\text{track}} = r_{\text{object},k}^{\text{track}} + r_{\text{control},k}^{\text{track}} + r_{\text{failure},k}^{\text{track}} + r_{\text{progress},k}, \tag{12}$$

where

$$r_{\text{object},k}^{\text{track}} = -\bar{k}_1 e_k^2 - \bar{k}_2 (v_b - v_{g,k})^2, \tag{13a}$$

$$r_{\text{control},k}^{\text{track}} = -\bar{k}_3 u_k^2 - \bar{k}_4 a_k^2, \tag{13b}$$

$$r_{\text{failure},k}^{\text{track}} = \begin{cases} -\bar{k}_5 & \text{if } (|\theta| > \theta_{\max}) \vee (|e_k| > e_{\max}) \\ 0 & \text{otherwise} \end{cases}, \tag{13c}$$

$$r_{\text{progress},k} = \bar{k}_6 (|e_{k-1}| - |e_k|). \tag{13d}$$

D. Evaluation Protocol

1) Evaluation Metrics:

a) Target-position Balancing: Each evaluation episode lasts 10 s. We generate 500 test episodes by randomly sampling the initial ball position and target position from the feasible range. We evaluate the controller using five metrics: success rate (SR), steady-state error (SE), convergence time (CONT), climb time (CLIT), and computation time (COMT). Here, ϵ denotes the target-position error tolerance. An episode is successful if the ball absolute position error falls below ϵ before the episode ends and remains within this tolerance thereafter; SR is the percentage of successful episodes. SE is the mean absolute position error over a fixed terminal time window W. CONT denotes the first time at which the error enters and subsequently remains within the tolerance band, while CLIT denotes the first entry time into the tolerance band. COMT measures the controller computation time. The tolerance ϵ and terminal time window W can be configured according to the evaluation requirements.

b) Trajectory Tracking: We also evaluate 500 test episodes, using stratified random sampling across the three reference families (sinusoidal, trapezoidal, and triangular) so that no single family dominates the aggregate score. Within each family, the trajectory amplitude and frequency are randomly sampled from predefined ranges. We evaluate tracking performance using four metrics: mean absolute position error (MAE), root-mean-square position error (RMSE), maximum absolute position error (MAXE), and compute time (COMT). MAE characterizes average tracking accuracy, RMSE penalizes occasional large deviations more strongly, and MAXE captures the worst-case tracking excursion.

2) Robustness Test Suite: Beyond nominal task evaluation, the benchmark includes a robustness test suite designed to probe how well a controller tolerates uncertainty and mismatch between training/designing and deployment conditions. The central idea is to introduce controlled variations into the environment, primarily through changes in system parameters and execution imperfections, so that algorithms can be evaluated not only by their nominal performance but also by their ability to generalize to non-ideal conditions. Specifically, we design four robustness tests.

a) Ball-mass Variation: The first robustness test varies the ball mass. Since the ball mass directly affects the inertial and gravitational terms in the system dynamics, changes in this parameter alter the difficulty of balancing and tracking. By evaluating controllers on balls with different masses, the benchmark measures whether the learned or designed policy can generalize beyond a single nominal object and remain effective under payload-related parameter shifts.

b) Low-level Gain Variation: The second robustness test varies the control gains of the drone's low-level controllers. In practice, high-level commands are never executed directly; instead, they must be tracked by an onboard flight controller whose behavior depends strongly on its gain setting. By reducing or perturbing these gains, the benchmark simulates imperfect or changed low-level control behavior and evaluates whether the high-level algorithm remains effective when command tracking becomes less accurate or less responsive.

c) Action Delay: The third robustness test introduces explicit action delay. Action delay is pervasive in real robotic systems due to sensing, communication, computation, and actuation latencies, and it is particularly harmful in highly dynamic tasks. In this benchmark, robustness to delay is assessed by inserting a configurable delay buffer between the high-level controller output and the executed command, so that the command applied at the current step corresponds to a past decision. This setting enables systematic evaluation of how controller performance degrades in the presence of realistic execution lag.

d) Disturbance Injection: The fourth robustness test injects external disturbance by allowing the otherwise fixed beam endpoint to undergo random vertical motion. This perturbation mimics environmental disturbance or support-point vibration and directly affects the beam-ball dynamics. To model temporally correlated disturbance rather than independent impulsive noise, we generate the endpoint motion using an Ornstein-Uhlenbeck (OU) process. The disturbed endpoint motion is then superimposed on the nominal fixed-end configuration to assess the controller's resilience to temporally correlated external perturbations.

III. Reference Control Framework

In this letter, we propose a reference control framework for the *Target-position Balancing* task based on the velocity-command interface. The high-level controller outputs the desired vertical velocity command for the drone, while a low-level velocity controller tracks the commanded motion. Although this hierarchy is practical for real drone systems, it inevitably introduces action delay caused by actuation latency. This delay is particularly harmful in drone-ball balancing because the ball dynamics are sensitive to timing errors. Therefore, the proposed framework explicitly addresses action delay through model-predictive state prediction and delay identification.

A. Model-predictive Delay Compensation

To compensate for action delay, we follow the idea of a d-step-ahead state predictor [18], using the constructed high-level dynamic model in Equation (5) to predict the future system state. We refer to this component as the model-based predictor. The control structure with the model-based predictor is illustrated in Fig. 3. Specifically, let the current observed system state be s_k and the historical action sequence be $\{a_{k-d}, a_{k-d+1}, \ldots, a_{k-1}\}$, where d denotes the number of delay steps. Using the high-level dynamic model (Equation (5)), we predict the system state d steps ahead, $\hat{s}_{k+d}$, based on s_k and the historical action sequence $\{a_{k-d}, a_{k-d+1}, \ldots, a_{k-1}\}$. This predicted state $\hat{s}_{k+d}$ is then fed to the high-level controller to generate timely commands.

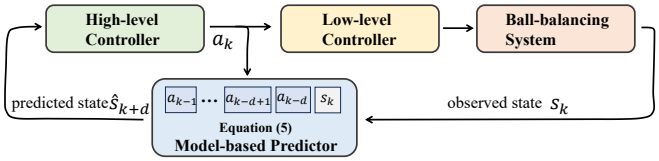

Fig. 3. The delay-aware control structure.

B. Delay Identification

To estimate the delay step d, we employ a time-domain method based on the cross-correlation function. Specifically, we execute the high-level policy on the real-world system and record the commanded vertical velocity (denoted as Sequence A) and the actual vertical velocity of the drone (denoted as Sequence B). The cross-correlation value $R_{AB}[q]$ for a candidate delay step q is computed as:

$$R_{AB}[q] = \sum_{i=0}^{T} A[i]B[i+q], \tag{14}$$

where T denotes the length of the signal sequences. The most probable delay step d corresponds to the delay that maximizes the cross-correlation:

$$d = \arg\max_{q \in \mathbb{N}^+} R_{AB}[q]. \tag{15}$$

In the real-world system, the identified delay is $d = 16$ control steps. Given the 60 Hz control frequency, this corresponds to a delay duration of approximately 0.267 s. Although this letter only presents the offline delay-identification method for constant-delay tasks, this method can be readily extended to an online form and therefore has the potential to handle time-varying delays.

IV. Baselines

For the high-level controller, we design and implement three baselines: a cascaded PID controller, a nonlinear MPC controller, and a model-free RL controller. All baselines are instantiated under the same reference framework and are evaluated on the *Target-position Balancing* task in both simulation and the real-world system.

A. The Cascaded PID Controller

Given the nonlinear dynamics of the system, we employ a cascaded PID controller as the high-level controller. The outer loop computes the desired beam angular displacement θ based on the ball's position error e, while the inner loop generates the required drone vertical-velocity increment Δv_{r_z} from the current beam-angle error. Both loops are implemented using an incremental PID formulation. In discrete time, the output of the incremental PID controller at step t_k is given by:

$$\Delta u_k = K_p e'_k + \frac{K_p \Delta t}{T_i} e_k + \frac{K_p T_d}{\Delta t} e''_k, \tag{16}$$

where e_k represents the system error at timestep t_k, $e'_k = e_k - e_{k-1}$ is its first-order backward difference, and $e''_k = (e_k - e_{k-1}) - (e_{k-1} - e_{k-2})$ is the second-order backward difference. Here, Δt denotes the sampling time interval, K_p is the proportional gain, T_i and T_d are the integral and derivative time constants, respectively. The PID parameters were initialized empirically and then refined through local sampling, with the overall tuning process taking several hours. Specifically, the tuned parameters are: the outer-loop controller uses $K_p = 0.5, T_i = 1.5, T_d = 0.6$; the inner-loop controller uses $K_p = 10.0, T_i = 1{,}000.0, T_d = 0.0$.

B. The Nonlinear MPC Controller

We formulate the high-level ball balancing task as an optimal control problem:

$$\min_{\mathbf{x}_{[0:N]}, \mathbf{u}_{[0:N]}, \mathbf{z}_{[0:N]}} \sum_{k=0}^{N} l(\mathbf{x}_k, \mathbf{u}_k, \mathbf{z}_k) + l_N(\mathbf{x}_N, \mathbf{u}_N, \mathbf{z}_N) \tag{17a}$$

s.t.:

$$\mathbf{x}_0 = \hat{\mathbf{x}}, \tag{17b}$$

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{z}_k) \ \forall\ k \in \{0, \cdots, N-1\}, \tag{17c}$$

$$\mathbf{g}_{\text{alg}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{z}_k) = 0 \ \forall\ k \in \{0, \cdots, N-1\}, \tag{17d}$$

$$\mathbf{g}_{\text{con}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{z}_k) \le 0 \ \forall\ k \in \{0, \cdots, N-1\}, \tag{17e}$$

where $\mathbf{x}_{[0:N]}$, $\mathbf{u}_{[0:N]}$, and $\mathbf{z}_{[0:N]}$ represent states, inputs, and algebraic states from stage 0 to N, respectively, $\hat{\mathbf{x}}$ is the measured state, $l(\cdot)$ is the running cost, $l_N(\cdot)$ is the terminal cost, $\mathbf{f}(\cdot)$ is the discrete-time dynamics, $\mathbf{g}_{\text{alg}}$ and $\mathbf{g}_{\text{con}}$ are the equality and inequality constraints. The definitions are:

$$\mathbf{x} = [p_b, v_b, \theta, v_{r_z}]^T, \tag{18a}$$

$$\mathbf{u} = [a_{r_z}], \tag{18b}$$

$$\mathbf{z} = [\omega, \beta]^T, \tag{18c}$$

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{z}) = \mathbf{x} + \begin{bmatrix} v_b \\ \frac{1}{\frac{J_b}{r_b^2} + m_b}(m_b(p_b - L)\omega^2 - m_b g \sin\theta) \\ \omega \\ a_{r_z} \end{bmatrix} \Delta t, \tag{18d}$$

$$\mathbf{g}_{\text{alg}}(\mathbf{x}, \mathbf{u}, \mathbf{z}) = \begin{bmatrix} L\omega + v_{r_z} \frac{\cos\beta}{\cos(\beta - \theta)} \\ \sin\beta - (1 - \cos\theta)\frac{L}{l} \end{bmatrix}, \tag{18e}$$

$$\mathbf{g}_{\text{con}}(\mathbf{x},\mathbf{u},\mathbf{z}) = \begin{bmatrix} -p_b \\ p_b - 0.7 \\ |\theta| - 40^\circ \\ |a_{r_z}| - a_{\max} \end{bmatrix}, \tag{18f}$$

$$l(\mathbf{x},\mathbf{u},\mathbf{z}) = (p_b - p_g)^2 + 0.25v_b^2 + 0.5v_{r_z}^2 + 0.05\Delta a_{r_z}^2, \tag{18g}$$

$$l_N(\mathbf{x}_N,\mathbf{u}_N,\mathbf{z}_N) = (p_{b,N} - p_g)^2 + v_{b,N}^2 + \theta_N^2 + v_{r_z,N}^2. \tag{18h}$$

The implementation utilizes the do-mpc toolbox [19]. We empirically evaluated six prediction horizon lengths (15, 20, 25, 30, 35, and 40 time steps) and report three representative cases (15, 25, and 35) to illustrate the trade-off between computational complexity and control performance.

C. The Model-free RL Controller

In this baseline, we empirically use an 8-dimensional state space: $[e, e', e'', v_b, \theta, \omega, v_{r_z}, a]$ instead of the given observation space in Section II-C. In addition, the original action space and reward function are retained. As empirical results demonstrate that Robust Policy Optimization (RPO) [20] outperforms Proximal Policy Optimization (PPO) [21] in improving task performance, this letter selects RPO to train the ball-balancing policy. We directly use the implementation provided by the SKRL library [22]. For training the RL-based high-level policy, we designed both actor and critic networks as three-layer MLPs with 256 neurons per layer, where the actor's output layer employed a *tanh* activation function. We trained each policy over six random seeds and report the mean and standard deviation. Each run uses 90,000 environment steps with 1,024 parallel environments and takes about 3 h on a single NVIDIA GeForce RTX 4060 Ti GPU with 8 GB memory.

D. Simulation Results

1) Standard Performance Evaluation: We first evaluate the nominal performance of each controller without activating the robustness tests. In this evaluation, the simulation runs at 60 Hz with maximum acceleration $a_{\max} = 0.5\,\mathrm{m/s^2}$, ball mass $m_b = 0.0005\,\mathrm{kg}$, low-level velocity-controller gain 10, target-position error tolerance $\epsilon = 0.01\,\mathrm{m}$, and SE window $W = 1\,\mathrm{s}$. Since action delay is observed in the real system, the standard evaluation also includes a fixed 15-step delay. All controllers are designed or trained in the delay-free setting and then directly deployed under delayed execution, either without compensation or with the proposed model-predictive delay compensation. This setup evaluates how well each controller transfers from the nominal delay-free environment to delayed environments without delay-specific retuning. Quantitative results are summarized in Table III. It shows that introducing action delay substantially degrades all controllers when no compensation is used. In the delay-free setting, Cascaded PID, NMPC with horizon 35, and RL all achieve strong balancing performance. However, after direct deployment to the delay-compensated framework, only Cascaded PID maintains consistently high SR and low SE. Overall, Cascaded PID provides a strong and practical baseline under delayed execution.

TABLE III
STANDARD PERFORMANCE EVALUATION RESULTS; ORANGE INDICATES COMT VIOLATES THE 60HZ COMPUTATION REQUIREMENT.

Type	Controller	SR(%)↑	SE(mm)↓	CONT(s)↓	CLIT(s)↓	COMT(ms)↓
Delay-free	Cascaded PID	100	0.209 ± 0.122	3.278 ± 1.345	2.352 ± 0.138	0.444
	NMPC (horizon=15)	0	66.419 ± 35.714	10.017 ± 0.000	8.451 ± 1.849	13.239
	NMPC (horizon=25)	99.600	14.536 ± 7.838	9.369 ± 0.727	5.644 ± 2.106	17.510
	NMPC (horizon=35)	100	1.458 ± 0.793	6.615 ± 1.644	5.083 ± 1.193	22.999
	RL	100 ± 0	4.597 ± 2.641	4.158 ± 1.501	2.783 ± 0.898	0.767
Delay w/o comp	Cascaded PID	4.400	144.326 ± 83.208	10.013 ± 0.023	1.558 ± 0.109	0.468
	NMPC (horizon=15)	2.600	160.492 ± 113.996	10.012 ± 0.039	5.373 ± 2.237	16.897
	NMPC (horizon=25)	3.400	101.514 ± 102.729	10.005 ± 0.083	5.276 ± 1.879	26.042
	NMPC (horizon=35)	0	223.654 ± 81.406	10.017 ± 0.013	1.823 ± 0.798	53.574
	RL	0 ± 0	216.298 ± 69.126	10.017 ± 0.003	1.518 ± 0.176	0.840
Delay w/ comp	Cascaded PID	100	0.795 ± 0.410	3.899 ± 1.285	2.443 ± 0.108	14.059
	NMPC (horizon=15)	9.000	58.718 ± 39.991	9.975 ± 0.154	7.695 ± 2.318	11.827
	NMPC (horizon=25)	17.800	56.832 ± 45.293	9.952 ± 0.244	3.736 ± 0.287	17.482
	NMPC (horizon=35)	10.200	15.251 ± 8.413	9.908 ± 0.323	3.419 ± 1.202	23.495
	RL	82.800 ± 32.828	6.198 ± 5.274	7.056 ± 2.172	2.508 ± 0.496	11.717

TABLE IV
ROBUSTNESS EVALUATION RESULTS.

Type	Controller	SR(%)↑	SE(mm)↓	CONT(s)↓	CLIT(s)↓
Mass	Cascaded PID	28.200	16.341 ± 8.812	8.751 ± 2.227	3.719 ± 2.823
	NMPC (horizon=15)	0.400	258.485 ± 129.922	10.016 ± 0.000	4.533 ± 2.028
	NMPC (horizon=25)	1.800	209.995 ± 116.766	10.014 ± 0.020	6.474 ± 3.541
	NMPC (horizon=35)	7.600	83.680 ± 73.668	9.995 ± 0.088	5.962 ± 3.623
	RL	5.033 ± 5.662	32.897 ± 18.612	9.957 ± 0.380	5.275 ± 3.629
Gain	Cascaded PID	92.200	5.648 ± 11.465	5.778 ± 2.466	2.320 ± 0.091
	NMPC (horizon=15)	8.600	72.754 ± 53.077	9.992 ± 0.093	7.544 ± 2.535
	NMPC (horizon=25)	7.800	105.193 ± 77.616	9.991 ± 0.158	3.846 ± 1.059
	NMPC (horizon=35)	0.400	75.333 ± 95.653	10.013 ± 0.065	2.734 ± 0.141
	RL	26.033 ± 13.980	45.202 ± 65.999	9.683 ± 0.971	2.358 ± 0.195
Disturbance	Cascaded PID	14.000	53.425 ± 40.160	9.905 ± 0.429	3.773 ± 2.531
	NMPC (horizon=15)	0.600	248.262 ± 121.605	10.016 ± 0.016	4.355 ± 2.360
	NMPC (horizon=25)	0.200	226.220 ± 87.548	10.016 ± 0.014	4.154 ± 2.178
	NMPC (horizon=35)	2.000	196.699 ± 109.675	10.014 ± 0.030	4.087 ± 2.651
	RL	6.100 ± 1.050	125.050 ± 100.728	9.991 ± 0.148	3.795 ± 2.700

TABLE V
ROBUSTNESS EVALUATION WITH LONG DELAYS.

Delay	Controller	SR(%)↑	SE(mm)↓	CONT(s)↓	CLIT(s)↓
30 (0.5s)	Cascaded PID	84.000	7.958 ± 4.838	7.955 ± 2.047	2.637 ± 0.083
	NMPC (horizon=15)	11.000	52.121 ± 36.245	9.967 ± 0.152	7.779 ± 2.243
	NMPC (horizon=25)	4.200	68.035 ± 38.087	9.995 ± 0.171	4.006 ± 0.473
	NMPC (horizon=35)	52.000	31.296 ± 33.580	9.879 ± 0.209	3.108 ± 0.162
	RL	37.900 ± 15.109	27.074 ± 18.950	9.861 ± 0.457	2.652 ± 0.262
45 (0.75s)	Cascaded PID	33.800	18.103 ± 19.096	9.759 ± 0.513	2.925 ± 0.086
	NMPC (horizon=15)	15.200	47.756 ± 32.725	9.967 ± 0.132	7.856 ± 2.175
	NMPC (horizon=25)	1.400	64.664 ± 34.857	10.004 ± 0.114	4.414 ± 0.964
	NMPC (horizon=35)	2.200	67.687 ± 67.625	10.014 ± 0.022	3.373 ± 0.172
	RL	0.767 ± 0.559	100.575 ± 61.400	10.016 ± 0.027	2.900 ± 0.179
60 (1.0s)	Cascaded PID	2.600	53.045 ± 76.236	9.993 ± 0.177	3.228 ± 0.101
	NMPC (horizon=15)	15.000	47.354 ± 29.214	9.982 ± 0.095	7.928 ± 2.114
	NMPC (horizon=25)	5.600	47.965 ± 27.404	9.983 ± 0.141	5.025 ± 1.574
	NMPC (horizon=35)	0.800	42.323 ± 74.530	10.010 ± 0.078	3.698 ± 0.200
	RL	1.500 ± 1.399	248.859 ± 102.127	10.016 ± 0.004	3.198 ± 0.204

2) Robustness Evaluation: We then evaluate zero-shot robustness by varying one factor at a time while keeping the remaining settings consistent with the standard evaluation. All controllers are designed or trained in the nominal environment and then directly tested under perturbed conditions without retuning. Because action delay is a fundamental issue in the real system, robustness is assessed under the 15-step delay condition by default. To further examine the effect of increasing latency, we also test controller performance under multiple longer delay settings. Quantitative results are summarized in Tables IV and V. Table IV shows that mass variation, reduced low-level gain, and external disturbance all lead to substantial performance degradation across controllers. Table V further shows that controller performance generally deteriorates as the delay increases. These results indicate that practical system imperfections, model mismatch, and external disturbances remain major challenges for robust drone-ball balancing.

TABLE VI
REAL-WORLD RESULTS.

Goal	Controller	SR$^{\uparrow}$	SE(mm)$^{\downarrow}$	CONT(s)$^{\downarrow}$	CLIT(s)$^{\downarrow}$
0.15	Cascaded PID	2/9	85.454 ± 52.853	19.176 ± 2.201	5.978 ± 7.390
	NMPC (horizon=25)	0/9	284.003 ± 82.575	20.017 ± 0.000	9.124 ± 9.743
	RL	1/9	113.055 ± 64.221	19.987 ± 0.084	7.696 ± 8.007
0.35	Cascaded PID	6/9	21.944 ± 16.780	13.641 ± 7.008	4.024 ± 5.668
	NMPC (horizon=25)	0/9	232.710 ± 47.256	20.017 ± 0.000	2.644 ± 6.150
	RL	5/9	33.617 ± 17.146	16.848 ± 5.290	2.137 ± 0.876
0.55	Cascaded PID	0/9	89.232 ± 11.810	20.017 ± 0.000	11.910 ± 9.077
	NMPC (horizon=25)	0/9	329.621 ± 61.211	20.017 ± 0.000	6.945 ± 9.247
	RL	0/9	71.738 ± 27.767	20.017 ± 0.000	5.042 ± 5.519

E. Real-world Results

The real system is shown in Fig. 1(b). We directly transfer the simulation-designed/trained controllers to the real system without real-world fine-tuning. For nonlinear MPC, we use a horizon of 25 as a representative trade-off between control performance and computational cost. Experiments are conducted at three target positions, $p_g = 0.15$, 0.35, and 0.55, with target-position error tolerance $\epsilon = 0.03\,\mathrm{m}$ and SE window $W = 5\,\mathrm{s}$. For each target, 9 initial ball positions are randomly sampled, and each episode lasts $20\,\mathrm{s}$. Quantitative results are summarized in Table VI. It shows that cascaded PID achieves the best overall control performance at $p_g = 0.35$. For the off-center targets, $p_g = 0.15$ and $p_g = 0.55$, all controllers exhibit degraded performance, likely due to gravity-induced beam bending. These results confirm the baseline real-world balancing capability of cascaded PID, while also revealing its limited adaptability to unmodeled or non-ideal dynamics. A detailed analysis and mitigation of beam bending will be considered in future work.

V. CONCLUSION, LIMITATIONS, AND FUTURE WORK

This letter presents Aerial-Balance-Bench, a controlled and simplified drone-ball balancing benchmark for indirect dynamic aerial manipulation. By formalizing task definitions, command interfaces, simulation environments, evaluation metrics, robustness tests, and reference implementations, the benchmark provides a reproducible platform for developing and comparing algorithms that manipulate dynamic objects through intermediate mechanisms under controlled physical assumptions. We hope this benchmark can broaden aerial manipulation research beyond direct and static interaction, and support future studies on control, learning, robustness, and sim-to-real transfer in indirect dynamic manipulation.

A current limitation of Aerial-Balance-Bench is that the physical setup is intentionally simplified: the drone motion is restricted to the vertical direction and the opposite end of the beam is fixed. Although these assumptions allow the benchmark to focus more specifically on the core problem of ball manipulation, they also abstract away practical factors (e.g., full three-dimensional flight and moving supports). Future work will relax these assumptions and extend the benchmark toward more realistic and interactive scenarios, including multi-drone cooperative transportation and balancing, and human-drone collaborative transportation with balancing.

REFERENCES

[1] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2021.
[2] M. Wang, Z. Chen, K. Guo, X. Yu, Y. Zhang, L. Guo, and W. Wang, "Millimeter-level pick and peg-in-hole task achieved by aerial manipulator," *IEEE Transactions on Robotics*, vol. 40, pp. 1242–1260, 2023.
[3] G. He, X. Guo, L. Tang, Y. Zhang, M. Mousaei, J. Xu, J. Geng, S. Scherer, and G. Shi, "Flying hand: End-effector-centric framework for versatile aerial manipulation teleoperation and policy learning," *arXiv preprint arXiv:2504.10334*, 2025.
[4] D. Lee, B. Kim, and H. J. Kim, "Autonomous aerial manipulation at arbitrary pose in SE(3) with robust control and whole-body planning," *The International Journal of Robotics Research*, 2025.
[5] S. Ubellacker, A. Ray, J. M. Bern, J. Strader, and L. Carlone, "High-speed aerial grasping using a soft drone with onboard perception," *npj Robotics*, vol. 2, no. 1, p. 5, 2024.
[6] H. Wang, H. Li, B. Zhou, F. Gao, and S. Shen, "Impact-aware planning and control for aerial robots with suspended payloads," *IEEE Transactions on Robotics*, vol. 40, pp. 2478–2497, 2024.
[7] S. Sun, X. Wang, D. Sanalitro, A. Franchi, M. Tognon, and J. Alonso-Mora, "Agile and cooperative aerial manipulation of a cable-suspended load," *Science Robotics*, vol. 10, no. 107, p. eadu8015, 2025.
[8] G. Li, X. Liu, and G. Loianno, "Human-aware physical human-robot collaborative transportation and manipulation with multiple aerial robots," *IEEE Transactions on Robotics*, vol. 41, pp. 762–781, 2025.
[9] Y. Wu, F. Yang, R. Jin, Y. Zhong, J. Wang, X. Wu, and F. Gao, "Hand-like autonomous flying robot for airborne grasping and interaction," *Nature Communications*, vol. 17, p. 2200, 2026.
[10] A. Suarez, V. M. Vega, M. Fernandez, G. Heredia, and A. Ollero, "Benchmarks for aerial manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2650–2657, 2020.
[11] G. Li, X. Liu, and G. Loianno, "RotorTM: A flexible simulator for aerial transportation and manipulation," *IEEE Transactions on Robotics*, 2023.
[12] J. Sun, B. Tian, Q. Zhang, C. Li, Z. Song, Z. Cui, Y. Lv, and Y. Tian, "AIR-VLA: Vision-language-action systems for aerial manipulation," *arXiv preprint arXiv:2601.21602*, 2026.
[13] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear control via approximate input-output linearization: The ball and beam example," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 392–398, 1992.
[14] J. Huang and W. J. Rugh, "An approximation method for the nonlinear servomechanism problem," *IEEE Transactions on Automatic Control*, vol. 37, no. 9, pp. 1395–1398, 1992.
[15] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, *et al.*, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.
[16] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.
[17] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.
[18] R. Lozano, P. Castillo, P. Garcia, and A. Dzul, "Robust prediction-based control for unstable delay systems: Application to the yaw control of a mini-helicopter," *Automatica*, vol. 40, no. 4, pp. 603–612, 2004.
[19] F. Fiedler, B. Karg, L. Lüken, D. Brandner, M. Heinlein, F. Brabender, and S. Lucia, "do-mpc: Towards fair nonlinear and robust model predictive control," *Control Engineering Practice*, vol. 140, p. 105676, 2023.
[20] M. M. Rahman and Y. Xue, "Robust policy optimization in deep reinforcement learning," *arXiv preprint arXiv:2212.07536*, 2022.
[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[22] A. Serrano-Muñoz, D. Chrysostomou, S. Bøgh, and N. Arana-Arexolaleiba, "skrl: Modular and flexible library for reinforcement learning," *Journal of Machine Learning Research*, vol. 24, no. 254, pp. 1–9, 2023.