

# asRoBallet: Closing the Sim2Real Gap via Friction-Aware Reinforcement Learning for Underactuated Spherical Dynamics

Fang Wan

Southern University of Science and Technology  
Shenzhen, China  
Email: wanfang@ieee.org

Guangyi Huang, Tianyu Wu, Zishang Zhang, Bangchao Huang  
Southern University of Science and Technology  
Shenzhen, China

Haoran Sun

The University of Hong Kong  
Hong Kong SAR, China

Mingdong Chen

AncoraSpring Robotics  
Shenzhen, China  
Email: chen@AncoraSpring.com

Chaoyang Song

Mohamed bin Zayed University of Artificial Intelligence  
Abu Dhabi, United Arab Emirates  
Email: songcy@ieee.org

**Abstract**—We introduce asRoBallet, to the best of our knowledge, the first end-to-end reinforcement learning (RL) locomotion policy deployed on a humanoid ballbot hardware platform. Historically, ballbots have served as a canonical benchmark for underactuated and nonholonomic control, which are characterized by a reality gap in complex friction models for wheel-ball-floor interactions. While current literature demonstrates successful handling of 3D balancing with LQR and MPC, transitioning to actual hardware for a humanoid ballbot using RL is currently hindered by critical gaps in contact modeling, actuator latency & jitter, and safe hardware exploration. This study proposes a high-fidelity MuJoCo simulation that explicitly models the discrete roller mechanics of ETH-type omni-wheels, thereby capturing parasitic vibrations and contact discontinuities that have previously been ignored. We also developed a *Friction-Aware Reinforcement Learning* framework that achieves zero-shot Sim2Real transfer by mastering the coupled rolling, lateral, and torsional friction channels at the wheel-ball and ball-floor interfaces. We designed asRoBallet through subtractive reconfiguration, repurposing key components from an overconstrained quadruped and integrating them into a newly designed structural frame to achieve a robust research platform at low cost. We also developed a generalized iOS ecosystem that transforms consumer electronics into a low-latency interface, enabling a single operator to orchestrate expressive humanoid maneuvers via intuitive natural motion.

**Index Terms**—Humanoid BallBot, Tribology Modeling, Reinforcement Learning, Zero-Shot Sim2Real

## I. INTRODUCTION

The single-ball-wheel robot, or “ballbot,” presents a theoretically optimal morphology for robotic assistants in human-centric environments [1, 2]. Its omnidirectional mobility allows for agile navigation in constrained domestic spaces, while its tall, slender profile facilitates compliant physical interaction and social presence [3, 4]. However, its widespread deployment remains limited compared to quadrupedal or differential-drive systems due to the *prohibitive complexity of custom hardware integration* and the *Sim2Real gap inherent*

*in controlling underactuated spherical dynamics.*

### A. Motivation and Related Work

The *first* barrier lies in hardware architecture. Traditional ballbot platforms typically rely on expensive, custom-built drivetrains utilizing high-gear-ratio motors to ensure precise kinematic constraints [5, 6, 7]. While effective at minimizing control uncertainty, this approach yields rigid, mechanically complex systems that lack impact resilience. Furthermore, perception often relies on dedicated industrial-grade LiDARs and robotic vision, driving costs beyond the reach of many research labs. This stands in stark contrast to recent advances in quadrupedal robotics, which have converged on proprioceptive quasi-direct-drive actuators [8] and smartphone-based computing [9]. These architectures offer high torque density, back-drivability, and significant cost reductions. However, the application of such compliant actuation ecosystems to the inherently unstable dynamics of a ballbot remains underexplored.

The *second*, and perhaps more critical, barrier is the *Reality Gap* in control. Classical control approaches for ballbots, such as Linear Quadratic Regulators (LQR) [2, 6] or Model Predictive Control (MPC) [10, 11], rely on linearized dynamic models (e.g., the Planar Inverted Pendulum). These controllers operate under strict assumptions: infinite friction at the contact point (no slip), negligible rolling resistance, and ideal actuation. In reality, the interface between a rubber-coated ball and omni-directional rollers is governed by complex tribology, specifically the chaotic coupling of rolling resistance, lateral slip, and torsional friction [12].

When coupled with the passive dynamics of the rollers and the floating ball, the system explodes in complexity. A single ETH-type omni-wheel with multiple rows of rollers could introduce 10 to 12 passive Degrees of Freedom (DoF). Standard model-based controllers struggle to account for these

microdynamics. Reinforcement learning has enabled end-to-end sim-to-real locomotion policies, particularly for quadruped and biped robots. However, RL-based ballbot locomotion remains underexplored, with prior studies limited to restricted control settings rather than deployable locomotion policies. [13] used RL to refine an iLQG-based controller on the Rezero ballbot, demonstrating hardware adaptation but relying on a model-based initialization and optimizing a finite-horizon motion controller. [14] proposed a compound controller in which a DDPG policy augments a conventional feedback controller to improve balance recovery, but the method was evaluated only in simulation and remained limited to the recovery task. [15] et al. trained a PPO policy for fall-damage reduction on a simulated ballbot, where the policy controlled only the arms during falling. More recently, [16] introduced a MuJoCo-based ballbot simulation by modeling omniwheels as capsules with anisotropic tangential friction and demonstrated PPO-based navigation only in simulation as a proof of concept, leaving sim-to-real transfer to future work.

Finally, a significant usability gap exists in the teleoperation of such high-Degree-of-Freedom systems, which typically require complex, non-intuitive interfaces (e.g., joystick arrays) and often necessitate multiple operators to coordinate locomotion and upper-body gestures separately. There is a distinct lack of lightweight interfaces that leverage ubiquitous consumer hardware to provide unified, whole-body control.

## B. Key Contributions

To bridge these gaps in hardware accessibility, control robustness, and interface usability, this paper presents asRoBallet, a holistic ballbot system for robust humanoid balancing. Our specific contributions are as follows:

- 1) We present a cost-effective, high-bandwidth humanoid ballbot design achieved by reconfiguring the proprioceptive actuation, computation core, and mobile perception of an overconstrained quadruped into an underactuated spherical morphology.
- 2) We introduce granular dynamics that explicitly simulate the discrete roller mechanics of ETH-type omni-wheels, successfully capturing the parasitic vibrations and roller handovers critical to physical balancing stability.
- 3) We propose a friction-aware reinforcement learning framework that achieves zero-shot hardware deployment by mastering the tribological couplings of rolling, lateral, and torsional friction at the wheel-ball-floor interfaces, enabling robust whole-body locomotion.

We also release an iOS app (asMagic: <https://apps.apple.com/us/app/asmagic/id6661033548>) that leverages consumer-grade sensing for low-latency, markerless whole-body teleoperation, facilitating intuitive human-robot interaction across diverse humanoid platforms. Public release of the asRoBallet will be updated here ([https://github.com/asRoBallet/asRoBallet\\_mujoco](https://github.com/asRoBallet/asRoBallet_mujoco)).

## II. RECONFIGURABLE DESIGN OF ASROBALLET

We employ a *subtractive reconfiguration* methodology to construct asRoBallet, transforming the open-source *asOverDog* quadruped [17, 18] into a humanoid ballbot. By retaining the high-bandwidth actuation, computation, and perceptual core while redesigning the structural chassis, we achieve a cost-effective, impact-resilient platform, as shown in Fig. 1.

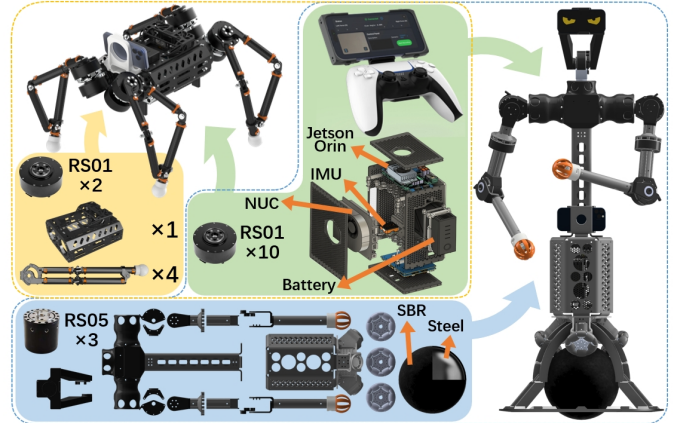


Fig. 1: Structural transformation from the asOverDog quadruped (top left) to the asRoBallet humanoid (right).

### A. Mechanical Architecture: From Quadruped to Ball

The design transformation primarily replaces the chassis and four overconstrained limbs to build asRoBallet.

1) *Spherical Wheel Base*: Three ETH-type omni-wheels (12 rollers each, offset by 120°) drive a 220 mm SBR-coated hollow steel ball. To minimize starting torque and cogging, we replaced the original quadruped drive units with lower-torque actuators (RobStride 05) rated at 5.5 Nm.

2) *Humanoid Upper Body & Frame*: The remaining high-torque actuators (RobStride 01, 17 Nm) are reconfigured to form a 2-DoF head and two 4-DoF arms with tactile spheres capable of Vision-based Deformable Perception (VBDeformP) [19, 20]. These are mounted on a lightweight T-shaped sheet-metal frame with a 3D printed shoulder designed to minimize rotational inertia relative to the Center of Mass (CoM). The final system comprises 49 degrees of freedom (13 active, 36 passive), presenting a highly underactuated control challenge.

### B. Proprioceptive & Computational Core

We transplant the electro-mechanical stack directly to accelerate commissioning. The quasi-direct-drive actuators provide intrinsic proprioception via current-based torque estimation, which is essential for our friction-aware policy. The self-contained computational unit houses a DJI TB47S battery, a 2 kHz industrial IMU (XSense MTI630), an Intel NUC for real-time control, and a Jetson Orin for neural inference, eliminating the need for custom power electronics.

### C. Consumer-Grade Perception & Interaction

To minimize cost and complexity, we replace industrial sensing arrays with a unified consumer-grade ecosystem.

1) *Visual-Inertial Odometry (VIO)*: An iPhone 12 Pro mounted on the spine (Fig. 2) serves as the primary state estimator. By offloading VIO to the device’s Neural Engine, we achieve low-drift 6-DoF localization that feeds directly into the controller, eliminating the need for discrete LiDARs.

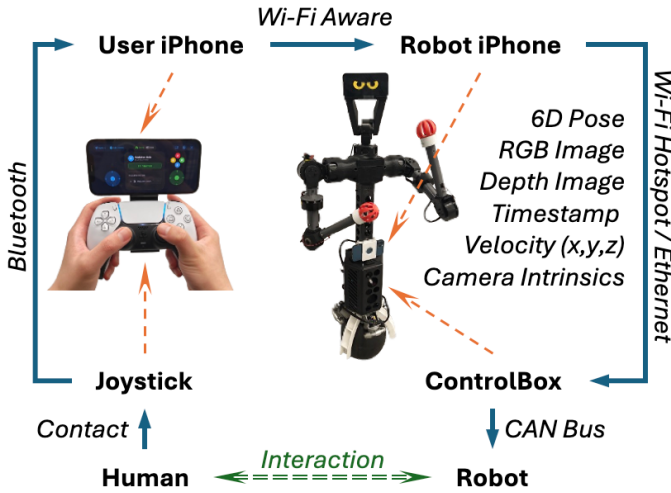


Fig. 2: Repurpose mobile perception & computing from an iPhone for robotics, the same for asRoBallet & asOverDog.

2) *Scalable Teleoperation*: We developed a custom iOS application that uses Wi-Fi Aware for low-latency peer-to-peer communication. The interface ranges from simple gamepad control to embodied telepresence, in which a distributed network of Apple devices (Watch, AirPods, iPhone) captures markerless human motion, enabling future generalization for intuitive whole-body retargeting.

### III. FRICTION-AWARE MODELING AND REINFORCEMENT LEARNING OF UNDERACTUATED SPHERICAL DYNAMICS

#### A. High-Fidelity Modeling in MuJoCo

Analytical models typically assume ideal rolling constraints (no slip) and negligible rolling resistance, thereby ignoring complex tribological phenomena, including the hysteresis of the rubber coating, the anisotropic slip of the omni-rollers, and the nonlinear friction of the actuators. To bridge the reality gap, we construct a high-fidelity MuJoCo multibody simulation (see Fig. 3) that models these uncertainties.

We model asRoBallet with a floating base comprising the upper torso, three omni-wheels, and a spherical wheel. For training, we added a virtual, low-damped ball joint between the torso and the ball to ensure stable contact during action exploration. The ball joint is set to ensure that the total contact force between wheels and ball equals the torso’s weight, thereby reproducing the real robot’s contact state. We experimentally found that training fails without the ball joint. Once the omni-wheels are no longer in contact with the ball, it is almost impossible to further improve the model or the reward. We will later demonstrate that the policy trained with this virtual ball joint is sufficiently robust to be directly transferred to simulation and real-world hardware tests.

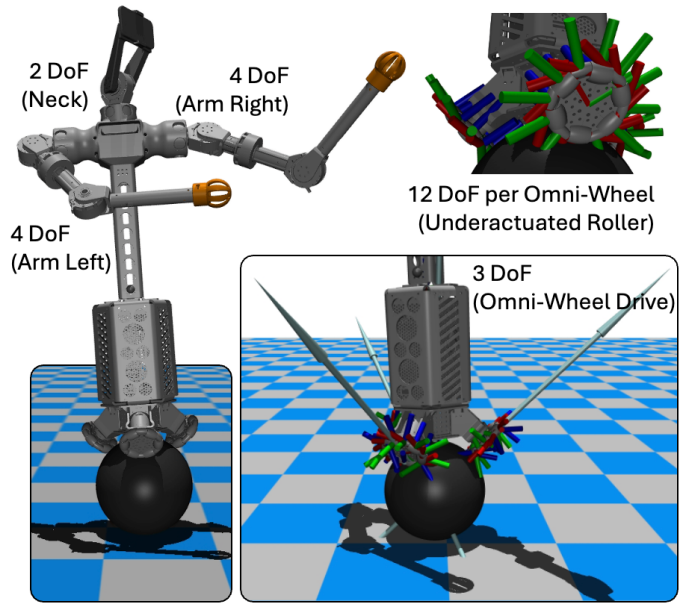


Fig. 3: High-fidelity MuJoCo simulation of asRoBallet’s highly underactuated spherical dynamics.

**Omni-wheel Modeling.** We explicitly model omni-wheels’ physical geometry. Each wheel is constructed as a ring of 12 free-spinning rollers (alternating between load-bearing and spacer rollers) attached to the main hub via hinge joints. This geometric fidelity is crucial for capturing *lateral compliance*. Holonomic motion in the asRoBallet hardware is achieved by allowing the rollers to spin freely orthogonal to the drive direction. In our simulation, this is not a mathematical abstraction but a physical outcome of the physics of roller–ball contact. We employ MuJoCo’s soft-constraint convex solver with an elliptic friction cone (`cone='elliptic'`) and a high constraint impedance ratio (`impratio=10`) to resolve the overconstrained nature of the three-point drive system without numerical instability. To handle the stick-slip transitions inherent to balancing, we enable the Newton solver with `noslip_iterations=1`.

**Solver & Contact Formulation.** We carefully model frictions at two critical contact interfaces. For the ball–floor contact pair, we specify full six-dimensional contact (normal, two tangential directions, torsional spin, and two rolling resistances) with relatively high sliding  $\mu_{slide} = 1$  and spin  $\mu_{torsion} = 0.01$  (interpreted as the effective diameter of the contact patch) friction but small rolling resistance  $\mu_{roll} = 0.0001$ . This choice prevents the ball from spinning freely in place, yields a realistic rolling onset, and helps the robot “bite” the floor when generating yaw torques. At the wheel–ball interface, we use standard three-dimensional contact (normal plus two tangential directions). We model a soft *rubber-like* layer on the ball by making contacts start stiff (impedance 0.85) and become very stiff (0.99) by 3 mm penetration (`solimp='0.85 0.99 0.003'`). Lateral compliance is provided by the rollers’ free joints rather than by anisotropic friction coefficients, thereby keeping the model both faithful

and numerically robust.

**Actuator Frictions.** We further identify the friction parameters of the omni-wheel actuators under no-load, including the viscous friction coefficient  $D_v$  and the Coulomb friction torque  $D_c$ , which is the minimum torque required to rotate the omni wheel. An increasing torque was given to the motor, and the motion of the motor can be described by [2]:

$$I_{motor}\ddot{\theta} = \tau(t) - D_v\dot{\theta} - D_c, \dot{\theta} > 0. \quad (1)$$

In the station-keeping task, the omni-wheels rotate slightly, with small motor torques typically oscillating around zero. Hence, it is especially sensitive to the Coulomb friction torque  $D_c$ . We experimentally determine  $D_c$  in our system to be within the range of 80 to 120 mNm, which is much higher than those reported for Rezero (42 mNm) [21] and Kugle (26 mNm) [6], which use much more expensive motors. The relatively large breakaway torque creates a dead zone in our wheel torques. Tiny corrective torques from the controller/policy do not move the wheels until the error grows enough to exceed that threshold, then the wheel *breaks free*, producing a bigger-than-intended correction. This manifests as position limit-cycle jitter.

We add a body frame  $B$  at the ball center, but rigidly attached to the body, providing linear velocity in local coordinates. We also add an IMU frame to host the gyro and orientation sensors, and an iPhone frame providing localization in world coordinates  $W$ . Another interesting finding is the selection of actuation types. Following previous work on simulating robotic arms and quadrupeds, the joints in the two arms and neck are actuated using position servos [22]. However, the three omni-wheels are actuated using direct motor torques. We experimentally find that training fails when a motor-type actuator is replaced with a velocity-type actuator.

## B. Reinforcement Learning

**Task Formulation.** We define two distinct locomotive tasks: *Velocity Tracking* and *Station Keeping* for long-term hovering. We decouple the upper-body manipulation from the base stabilization to simplify the learning landscape. During locomotion training, the arms and head are treated as passive inertial payloads subject to randomized configurations. This makes the base policy robust to significant shifts in the Center of Mass (CoM) without requiring a unified whole-body policy, thereby reducing the dimensionality of the learning problem.

**State and Action Space.** The action space  $\mathcal{A} \in \mathbb{R}^3$  are continuous torque commands for the three omni-wheel motors. The observation vector  $o_t$  differs slightly between tasks.

- **Velocity Tracking** ( $o_t^{VT} \in \mathbb{R}^{16}$ ): are defined with respect to the body coordinate, thus requiring only proprioceptive measurements.

$$o_t^{VT} = [\mathbf{v}_B, \hat{\mathbf{v}}_{cmd}, \mathbf{q}_{tilt}, \mathbf{x}_{COM}, \boldsymbol{\omega}_B, \mathbf{a}_{t-1}]^T \quad (2)$$

Here,  $\mathbf{v}_B \in \mathbb{R}^2$  is the body frame’s estimated linear velocity,  $\hat{\mathbf{v}}_{cmd}$  is the commanded velocity ( $\hat{v}_x, \hat{v}_y, \hat{\omega}_z$ ),  $\mathbf{q}_{tilt}$  is the roll and pitch tilting relative to gravity,  $\boldsymbol{\omega}_B$  is the gyroscope angular rate, and  $\mathbf{a}_{t-1}$  is the previous

action history.  $\mathbf{x}_{COM}$  is Center of Mass (COM) in the body frame  $B$  such that its value only reflects the change of arm configurations, given by

$${}^B\mathbf{x}_{COM} = {}^W R^T ({}^W\mathbf{x}_{COM} - {}^W\mathbf{x}_{ball}), \quad (3)$$

$${}^W\mathbf{x}_{COM} = \frac{1}{M} \sum_{i=1}^{n_b} m_i {}^W\mathbf{x}_i, \quad (4)$$

$$M = \sum_{i=1}^{n_b} m_i \quad (5)$$

where  ${}^W R^T$  is the rotation matrix of the body with reference to the world frame  $W$ ,  $m_i$  and  ${}^W\mathbf{x}_i$  are the mass and world-frame COM position of body  $i$  in MuJoCo.

- **Station Keeping** ( $o_t^{SK} \in \mathbb{R}^{17}$ ): We augment proprioceptive state with exteroceptive position data provided by the iPhone’s VIO pipeline.

$$o_t^{SK} = [\mathbf{v}_B, \mathbf{p}_W, \sin \psi, \cos \psi, \mathbf{q}_{tilt}, \boldsymbol{\omega}_B, \mathbf{a}_{t-1}]^T \quad (6)$$

where  $\mathbf{p}_W \in \mathbb{R}^2$  is the planar position error in the world frame and  $\psi$  is the yaw error.

A critical challenge in the observation design is estimating  $\mathbf{v}_B$ . In simulation, the ball’s linear velocity in the body frame is measured by a velocimeter mounted at the body frame  $B$ . In real experiments, we approximate this via non-slip forward kinematics at the wheel-ball interface.

**Reward Design.** As shown in Table I, the rewards for velocity tracking consist of (a) tracking reward of the target linear and angular velocities ( $\sigma=0.07$ ), (b) penalty for roll/pitch angular velocities, (c) an uprightness penalty to prevent excessive leaning, and (d) a smoothness penalty on control inputs (e.g.  $-\lambda|u_t - u_{t-1}|^2$  to discourage abrupt changes in action). The smoothness term directly promotes graceful motion by implicitly minimizing jerk. For station-keeping, the rewards include terms for (a) minimizing the distance of the robot from the goal position (current position when asRoBallet is asked to stay station), (b) minimizing the tilt angle (uprightness), and (c) low control effort and smoothness.

TABLE I: Summary of reward terms.

Task	Term	Weight	Expression
Velocity Tracking	Linear Vel.	0.5	$\exp(-\ \mathbf{v}_{xy} - \hat{\mathbf{v}}_{xy}\ ^2 / \sigma_{xy}^2)$
	Yaw Ang. Vel.	0.5	$\exp(-\ \omega_z - \hat{\omega}_z\ ^2 / \sigma_w^2)$
	Angular Vel.	-0.1	$\ \boldsymbol{\omega}_{xy}\ ^2$
	Torque	-0.1	$\ \boldsymbol{\tau}\ ^2$
	Action Rate Termination	-0.1 / 1	$\ \mathbf{a}_t - \mathbf{a}_{t-1}\ ^2 / \mathcal{K}(\phi, \theta \leq 20^\circ)$
Station Keeping	Pos. & Heading	1	$1 - \tanh(\ (x, y, \psi)\ )$
	Angular Vel.	-0.001	$\ \boldsymbol{\omega}\ ^2$
	Body Tilt	-0.001	$\ (\phi, \theta)\ ^2$
	Torque	-0.01	$\ \boldsymbol{\tau}\ ^2$
	Action Rate Termination	-0.01 / 1	$\ \mathbf{a}_t - \mathbf{a}_{t-1}\ ^2 / \mathcal{K}(\phi, \theta \leq 20^\circ)$

**Episode Reset & Termination.** For velocity tracking, episode-wise target commands are sampled by a masked update rule. A fresh proposal  $y_k \sim \mathcal{U}(V_{min}, V_{max})^3$  is filtered

by two independent Bernoulli masks  $z_k, w_k \sim \text{Bernoulli}(0.5)$  per dimension, yielding

$$x_{k+1} = x_k - w_k \odot (x_k - z_k \odot y_k), \quad (7)$$

which either holds, zeros, or replaces each component with a new value drawn from the distribution. This command is injected at reset and remains constant during the episode. The pose and velocity of the body are reset at every episode, following:

- 1) yaw:  $\psi \sim \mathcal{U}(-\pi, \pi)$ ,
- 2) roll/pitch:  $\theta_x, \theta_y \sim \mathcal{U}(-5, 5)$  degree,
- 3) horizontal linear velocities:  $v_x, v_y \sim \mathcal{U}(-0.5, 0.5)$  m/s,
- 4) angular velocities:  $w_x, w_y, w_z \sim \mathcal{U}(-0.1, 0.1)$  rad/s,
- 5) head/arms configurations uniformly sampled from the control range of each joint.

Episodes terminate early if roll or pitch exceeds  $20^\circ$ ; otherwise, they truncate at a horizon of 1,000 steps (equal to 10 seconds). The arm collision is ignored during configuration randomization, since we do not use the joint values; instead, we randomize the COM dynamics.

For station-keeping tasks, the targeted position and heading direction are set to zero. The reset is similar to velocity tracking except the first two items are replaced by resetting the initial pose in yaw of  $\psi \sim \mathcal{U}(-\pi/6, \pi/6)$  and location of  $x, y \sim \mathcal{U}(-0.5, 0.5)$ m.

**Domain Randomization & Robustness.** Previous work has shown that ballbots are highly sensitive to noises in the body’s angular velocities, resulting in tremors during hardware deployment, thereby requiring post-processing filters, such as low-pass and Kalman filters. Classic model-based controllers for ballbots, such as those at CMU and Kugle, are typically designed under the no-slip (high-friction) assumption and linearize the dynamics around that constraint to enable clean LQR/MPC designs with precise torque tracking. When friction drops (due to dust, smooth tile, or moisture), their assumptions may break. We implement a randomization routine that samples (a) the ball-floor tangential friction coefficient from  $\mathcal{U}(0.6, 1.2)$ , and (b) the motor’s Coulomb friction torque from  $\mathcal{U}(0.08, 0.12)$  Nm, (c) gyroscope sensor noises from  $N(0, 0.009)$  rad/s, matching real IMU’s noise level.

**Learning Algorithm & Training Setup.** All policies were trained with PPO using Stable-Baselines3. We used 8 parallel MuJoCo environments for data collection, with 2048 rollout steps per environment, yielding 16,384 samples per PPO update. The policy was optimized with a mini-batch size of 512 for 10 epochs per update. Each environment step corresponds to `frame_skip=5` MuJoCo simulation steps with a simulator time step of 0.002 s, resulting in a 100 Hz control frequency. The velocity-tracking task used `max_episode_steps = 1000`, corresponding to a maximum episode duration of 10 s, while the station-keeping task doubles the episode length. The longer horizon for station keeping encourages sustained balance and drift suppression, whereas the shorter horizon for velocity tracking increases the diversity of sampled command-following trials during training. Both policies were trained for

$4.0 \times 10^6$  environment steps. Training was performed on a workstation with an Intel Core i9-13900KF CPU; the wall-clock training time was approximately 2.5 hours.

The actor and critic use separate multilayer perceptrons with the same architecture: two hidden layers of 64 units with Tanh activations, followed by a linear output layer. The actor outputs three continuous omni-wheel torque commands, clipped to  $[-1, 1]$ .

#### IV. HUMANOID INTERACTION WITH AN IOS APP

This section introduces an iOS application we developed for asRoBallet, asMagic, providing (1) an All-in-One real-time sensory suite, (2) a scalable interface for humanoid interaction, and (3) a general-purpose visualization for humanoids.

##### A. iPhone as an All-in-One Real-time Sensing Suite

Robots require rich real-time sensory feedback for physical intelligence, but building such perceptual capability remains costly and difficult to reuse across embodiments [23, 24, 25]. Recent advances in mobile computing offer a low-cost alternative by integrating diverse sensors into consumer smartphones. In particular, the iPhone provides mature embedded sensing, software APIs, and deployment infrastructure. In this work, we present an iOS app that explores this opportunity, leveraging mobile perception as an *All-in-One* sensing solution for advanced robotics.

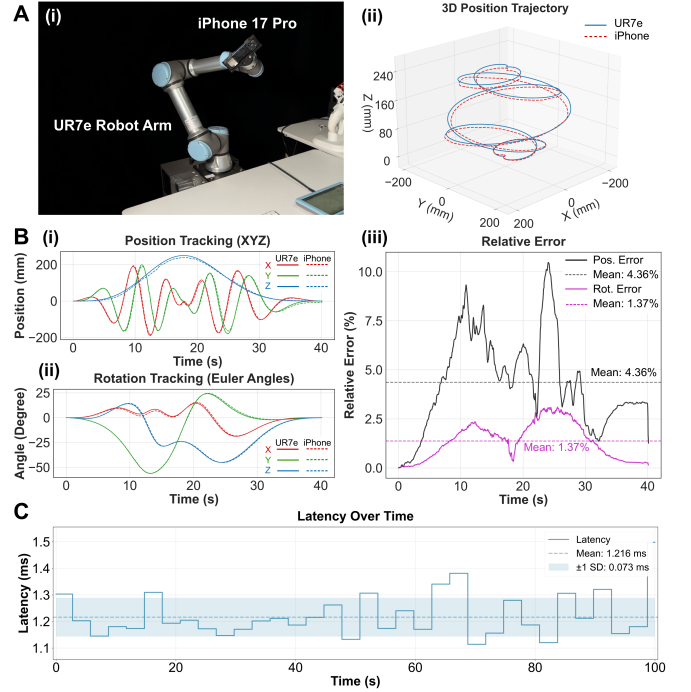


Fig. 4: iPhone’s ARKit pose estimation performance and peer-to-peer latency results.

The app provides extensive APIs to access sensors on Apple devices, including iPhone, Apple Watch, iPad, AirPods, and Mac. We evaluate of iPhone 17 Pro’s pose estimation performance against ground truth collected from an UR7e

robotic arm (Fig. 4A). We observed promising relative errors in position tracking (within 5%) and orientation tracking (within 1.5%) in Fig. 4B, both of which are sufficiently competitive to consider the iPhone Pro as a high-performing sensor for advanced robotics. We also tested latency performance between the iPhone 16 Pro Max and the iPhone 17 Pro. The average latency was approximately 1.2 ms under common usage conditions, measured over a 100 s communication session at 100 Hz, as shown in Fig. 4C. This performance is consistent over an extended period of use but drops significantly with heating. Adding active cooling resolves this issue.

### B. A Scalable Interface for Humanoid Interaction

The anthropomorphic design of humanoids poses interaction challenges due to the lack of an interface to retarget expressive human movements to the humanoid’s multi-DoF execution. A recent report from Disney Research indicates that 4 operators, each using a different interface, are required to achieve expressive, lively interaction with the H.E.R.B.I.E. humanoid in filmmaking. The majority of commercial humanoids are controlled using joystick-like devices for preprogrammed actions, which requires additional support for whole-body interaction.

We propose a scalable iOS-based interface that enables three levels of direct humanoid interaction: (1) simple control as a joystick (Touchscreen control in Fig. 2 without using a physical joystick), (2) wearable motion capture as an IMU suit (Fig. 5A), and (3) markerless motion capture via whole-body pose tracking (Fig. 5B), all achieved by a single operator using iOS devices.

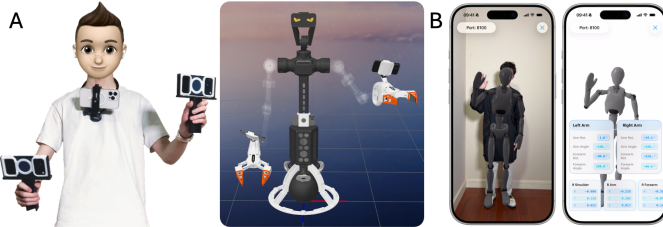


Fig. 5: Scalable Human-Robot Interaction via iOS & our app.

### C. A General Purpose Visualization for Humanoid Robots

We also developed a series of intuitive user interfaces to facilitate the visualization of human-robot interaction. Shown in Fig. 6A is the data streaming interface when using an iPhone Pro as a robot’s sensing suite, which is not limited to asRoBallet but generally applicable by attaching an iPhone to the robot. Fig. 6B is the MacOS version of our app, enabling users to load OpenUSD models to create a custom simulation environment, such as those in the BEHAVIOUR Challenge [26]. Fig. 6C is the iPad version of our app, showcasing a game-like visualization where users can either play with a virtual asRoBallet or load asRoBallet into a game for entertainment, where the robot model can be customized.



Fig. 6: Our iOS app ecosystem for multi-modal data streaming and sim-like interactive control.

## V. RESULTS, DEMONSTRATIONS, AND DISCUSSIONS

### A. Benchmark Whole-Body Performance of asRoBallet

Following the survey in [6], which identifies LQR and PI/PD controllers as among the most widely used strategies for ball-balancing robots, we adopt a fixed-gain LQR controller as a representative classical baseline. The LQR was designed and tuned around the nominal upright configuration and achieved stable performance in the nominal setting. We then compare it with the proposed RL policy on velocity-tracking and station-keeping tasks under four levels of increasing difficulty. These levels progressively violate the assumptions of the nominal linear model, including unmodeled changes in the center of mass due to random arm configurations, in the most difficult setting (a robustness stress test outside the fixed-gain LQR design envelope).

- 1) Fixed default arms with all joints fixed at zero.
- 2) Fixed default arms with torsional friction between the ball and the floor, randomly sampled from [0.01, 0.5]. The LQR controller relies on the common assumption that sufficiently large torsional friction prevents rotation about the vertical axis. However, this assumption is occasionally violated in practice, particularly on low-friction surfaces [6].
- 3) Fixed default arms with random initial body tilt within 5 degrees.
- 4) Random arm configurations representing the most challenging setting with significant variations in system inertia and coupling effects.

For each configuration, we report the success rate, mean absolute error (MAE), and standard deviation (STD) of the velocity-tracking error, as well as the success rate, steady-state range, and residual speed for station-keeping; the results are summarized in Table II.

Under nominal conditions (fixed-arm configuration), both controllers achieve a 100% success rate. However, the RL policy consistently achieves substantially lower tracking errors, with an MAE of 4.0 cm/s compared with 7.7 cm/s for

TABLE II: Comparison between LQR controller and RL policy simulation with different task complexities.

Methods	Arm Configuration	Velocity Tracking						Station Keeping			
		Success rate	Tracking MAE (cm/s, $10^{-2}$ rad/s)			STD			Success rate	Range (cm)	Speed (cm/s)
LQR	Fixed home position $P_0$	100	7.7	6.0	10.5	9.5	9.8	12.7	100	8.7	3.1
	Fixed $P_0$ & random friction	96	8.0	6.2	10.4	9.8	10.1	12.7	18	8.1	2.9
	Fixed $P_0$ & random orientation	56	7.5	5.9	10.5	9.1	9.6	12.7	0	/	/
	Random arm pose	86	23.1	23.0	10.8	28.7	27.7	12.1	6	8.4	3.3
RL	Fixed home position $P_0$	100	4.0	3.8	1.9	7.1	7.4	2.4	100	4.5	3.0
	Fixed $P_0$ & random friction	92	3.8	4.0	1.9	6.9	7.4	2.4	100	4.5	2.9
	Fixed $P_0$ & random orientation	100	4.1	3.8	1.9	7.2	7.0	2.4	100	4.3	2.4
	Random arm pose	100	6.6	6.2	1.7	8.8	8.9	2.1	100	5.0	3.1

TABLE III: Ablation study of RL policy in simulation.

Method	Velocity Tracking						Station Keeping			
	Success rate	Tracking MAE (cm/s, $10^{-2}$ rad/s)			STD			Success rate	Range (cm)	Speed (cm/s)
<b>Ablation on Obs/Reward</b>										
arm-obs	92	12.0	17.1	2.9	15.0	20.0	3.6	98	8.5	5.0
track-sigma-0.1	96	11.5	21.1	1.7	13.8	23.8	1.9	/	/	/
tanh-reward	96	6.6	7.8	1.9	8.5	10.1	2.3	<b>100</b>	<b>4.5</b>	<b>2.6</b>
exp-reward	<b>96</b>	<b>5.8</b>	<b>6.3</b>	<b>1.7</b>	<b>8.4</b>	<b>8.9</b>	<b>2.1</b>	96	7.7	4.4
<b>Ablation on Sim-to-Real</b>										
w/o-imu-noise	78	5.6	6.8	1.7	8.3	9.7	2.1	96	5.1	2.8
w/o-friction-randomization	82	7.2	7.1	2.0	8.6	9.4	2.3	98	5.9	3.4
asRoBallet	<b>96</b>	<b>5.8</b>	<b>6.3</b>	<b>1.7</b>	<b>8.4</b>	<b>8.9</b>	<b>2.1</b>	<b>100</b>	<b>4.5</b>	<b>2.6</b>

LQR, and a lower variance. As task difficulty increases, the performance gap becomes more pronounced. In the presence of random torsional friction, the LQR success rate drops to 96%, whereas RL maintains high accuracy with a lower MAE and STD. When random initial orientations are introduced, the LQR controller degrades significantly, achieving only a 56% success rate, while the RL controller remains robust with a 100% success rate. In the most challenging scenario with random arm configurations, the LQR controller exhibits large tracking errors (MAE exceeding 23 cm/s) and high variability, whereas the RL controller preserves stable performance with a 100% success rate and comparatively low error.

A similar trend is observed for station keeping. While both controllers perform well under nominal conditions, the LQR controller degrades rapidly when assumptions are violated. In scenarios involving random friction or orientation perturbations, the LQR success rate drops sharply (to 18% and 0%, respectively) and, in some cases, fails entirely. In contrast, the RL controller consistently achieves a 100% success rate across all configurations, maintaining a smaller steady-state position range and lower residual velocity. Notably, even under random arm configurations, RL preserves stable station-keeping behavior, whereas LQR performance degrades substantially.

### B. Ablation Study

We further conducted ablation studies to isolate the contribution of key design choices in the RL policy, covering: (i) observation/reward shaping, and (ii) domain randomization for sim-to-real robustness. All ablated variants were trained under the same protocol as the full RL method (denoted asRoBallet) and evaluated on the same velocity-tracking and station-keeping benchmarks reported in Table III. All evaluation tasks

were executed after removing the virtual ball joint between the torso and the ball. Each ablation task was repeated 50 times, with randomizations applied to the episode reset.

**Ablation on Observation/Reward Design.** Arm-obs: replaces the body-centric state representation by directly using the 8 arm joints’ observed positions, instead of the CoM-related features used in the full design. This ablation test assesses whether the policy benefits more from a task-relevant global representation (CoM) than from high-dimensional arm-configuration signals. Tanh-reward: uses a tanh-shaped tracking reward, which saturates for large errors and therefore reduces gradient sensitivity to outliers. Exp-reward: uses an exponential tracking reward, providing stronger shaping near the target region. Track-sigma-0.1: changes the exponential reward sharpness by setting  $\sigma = 0.1$ .

**Ablation on Domain Randomization.** w/o-imu-noise: removes injected IMU noise during training, to test whether sensor perturbations are necessary for policy robustness. w/o-friction-randomization: removes friction randomization (e.g., contact/torsional friction variations) during training, to test whether contact uncertainty is a key driver of generalization.

Replacing the CoM-related state with raw arm joint positions substantially degrades performance (velocity-tracking success drops to 92% with a much larger MAE/STD, and station-keeping shows greater drift), indicating that a compact, task-relevant global representation is critical. Reward shaping mainly affects the tracking–holding trade-off: the exponential tracking reward achieves the best tracking accuracy (MAE 5.8/6.3/1.7 cm/s with low STD, matching the full model), while the tanh-shaped reward slightly reduces tracking accuracy but yields the best station keeping (100% success, 4.5 cm range, 2.6 cm/s residual speed). In contrast, an overly

sharp exponential reward ( $\sigma = 0.1$ ) harms tracking, suggesting brittle optimization when deviations are penalized too aggressively. Removing domain randomization markedly reduces robustness, especially without IMU noise (tracking success 78%) and without friction randomization (tracking success 82%), highlighting that realistic sensing/contact perturbations are essential for reliable generalization.

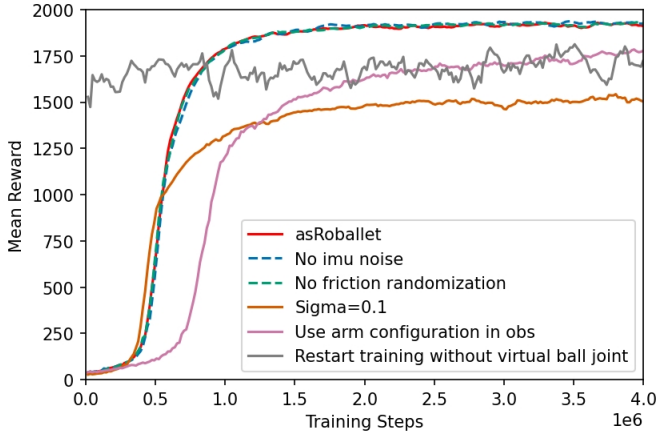


Fig. 7: Velocity-tracking learning curves (mean episode reward) for the proposed RL policy and ablations.

Figure 7 shows the learning curves (mean episode reward) for the velocity-tracking task under the proposed RL design and several ablations. Although removing IMU noise or applying friction randomization during training yields curves that are nearly indistinguishable from those in the proposed setting in terms of final reward and convergence speed, these modifications are important for achieving robust success rates at evaluation time.

In addition, we examined whether the policy could be further fine-tuned after removing the virtual ball joint. Specifically, we restarted training from the checkpoint at  $4.0 \times 10^6$  steps, where a stable policy had already been obtained, and continued training without the virtual ball joint (gray curve, “restart without ball joint”). This setting produced highly variable rewards, showed no clear monotonic improvement, and remained consistently below the performance ceiling achieved by the proposed method.

These results suggest that the low-damping virtual ball joint serves as a training-time stabilizer that facilitates effective exploration. Due to the severe underactuation of the ballbot system, unstructured early exploration can easily drive the system away from valid wheel-ball contact states, thereby greatly reducing the probability of collecting informative transitions and weakening the learning signal. The two empirical observations, namely i) training from scratch without the virtual ball joint and ii) removing the joint after initial convergence, both indicate that in our current setup the virtual joint stabilizes the learning process, while its removal destabilizes further training. Importantly, the virtual joint is used only during training; all evaluations and hardware deployment are

conducted without it. We therefore regard the joint as a training curriculum/stabilizer rather than a deployment crutch.

### C. Sim2Real Transfer of Underactuated Spherical Dynamics

We achieved zero-shot Sim2Real transfer of our RL policy during hardware deployment across various floor textures under indoor and outdoor lighting conditions, as shown in Fig. 8. asRoBallet moves smoothly in these common floor conditions we found around the campus, and can even pass through slight obstacles or move across different floor textures. One example is the last two images of Fig. 8, where asRoBallet successfully moved down a 5-degree slope on office carpet, overcame a slight aluminum bump on the door frame, and then transitioned to a hard, anti-slip floor. Please refer to our supplementary video for further demonstration.

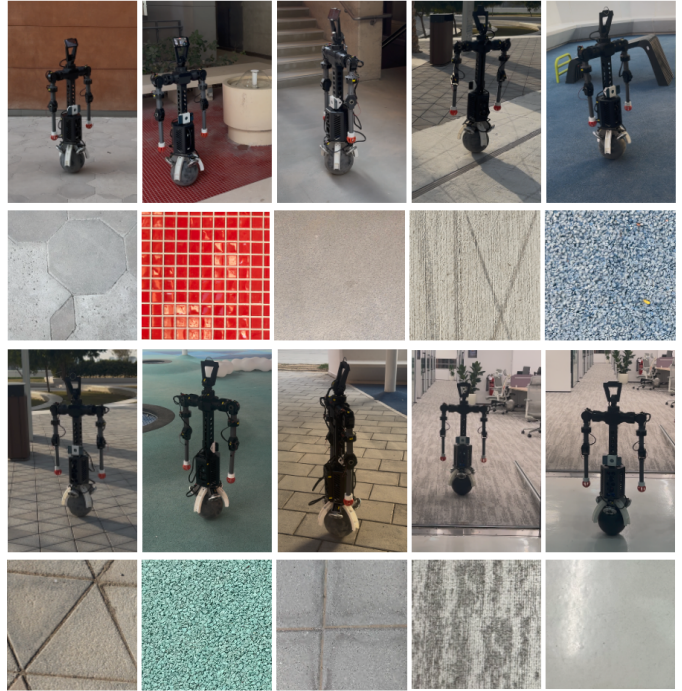


Fig. 8: Successful Sim2Real deployment of asRoBallet on various indoor & outdoor floor textures and lighting conditions.

The testing process is not without failure, especially when there is a sudden increase in floor height. When it falls, the 3D-printed neck connector appears to be the most vulnerable component. However, the whole system exhibited exceptional robustness and survived several strong impacts, remaining operational almost immediately after the fall. It should be noted that in the tests shown in Fig. 8, we *purposely* removed the protective ring around the ball in Fig. 1 to stress test asRoBallet’s performance and our proposed learning method.

We evaluated the stationary-keeping policy on asRoBallet across two floor surfaces: ceramic tiles and a yoga mat (Fig. 9A), where the position variation remained within 5 cm and 3 cm, respectively, and the average residual speeds were 5.1 cm/s and 4.4 cm/s, respectively. We further tested the push recovery on the yoga mat. asRoBallet managed to return to

the starting point after being pushed from different directions as far as 0.3 meters away in 7 trials (Fig. 9B). Hardware results on velocity tracking are shown in Fig. 9C, where asRoBallet is controlled by a human operator via a joystick on the common office floor carpet and achieved a mean absolute error of 0.05 m/s.

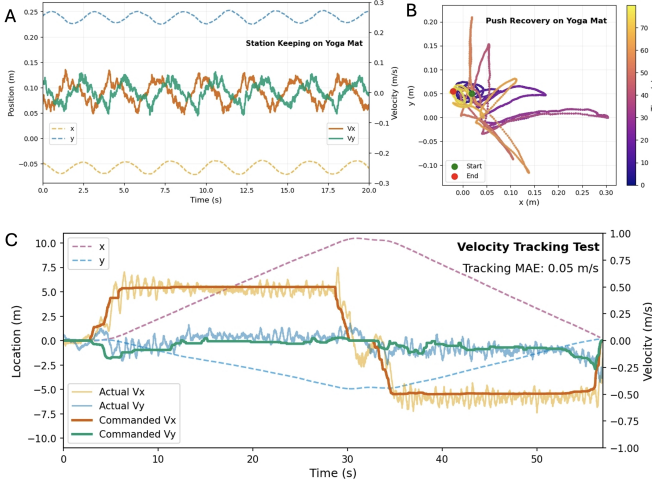


Fig. 9: Performance of real robot experiments.

## VI. CONCLUSION, LIMITATIONS, AND FUTURE WORK

This work presents *asRoBallet*, a holistic system that overcomes the barriers to deploying reinforcement learning on underactuated spherical robots. By closing the *Reality Gap* inherent in the complex tribology of wheel-ball-floor interactions, we, to the best of our knowledge, achieved the *first* end-to-end RL locomotion policy deployed on a humanoid ballbot hardware platform. We validated *subtractive reconfiguration* as a viable design strategy, demonstrating that high-performance humanoid research platforms can be constructed cost-effectively by repurposing the proprioceptive, computational, and perceptual cores of an overconstrained quadruped. We showed that the primary bottleneck for sim2real ballbot RL is not the learning algorithm itself, but the fidelity of the contact model. By explicitly simulating the discrete mechanics of the roller and the parasitic vibrations of ETH-type omni-wheels in MuJoCo, our *Friction-Aware* framework outperforms classical LQR baselines in robustness to unknown surface textures and calibration errors. We bridged the usability gap by integrating consumer-grade mobile computing directly into the control loop. The *iOS app* ecosystem demonstrates that pervasive devices (i.e., iPhones) can serve as low-latency, high-bandwidth interfaces for whole-body telepresence, democratizing access to advanced humanoid control.

Despite these successes, several *limitations* remain in the current implementation. As noted in our ablation studies, the policy fails to converge without a virtual ball joint between the torso and the ball during training. This dependence suggests that the reward landscape for spherical balancing is exceptionally sparse, requiring artificial constraints to guide the agent

through the initial exploration phase. While our policy manages station-keeping effectively, the Coulomb friction creates a *dead zone* that necessitates continuous micro-oscillations for stabilization. While our RL policy learns to manage this better than LQR does, it still yields residual jitter inherent to the current drivetrain configuration.

Future work will extend asRoBallet from robust locomotion to mobile manipulation by incorporating the currently passive upper-body joints into the active control loop. We plan to train unified policies over the spherical base and 4-DoF arms for dynamic tasks such as door opening and object transport, where the robot must exploit body motion to generate manipulation forces. We also aim to reduce the reality gap by learning vision-to-torque policies from raw iPhone inputs, enabling implicit adaptation to surface properties such as carpet or tile. Finally, using the peer-to-peer Wi-Fi Aware capabilities of the iOS ecosystem, we will explore multi-agent collaborative balancing for large-object transport and navigation in crowded social spaces.

## ACKNOWLEDGMENTS

This work was partially supported by Shenzhen AncoSpring Robotics Technology Co., Ltd., National Natural Science Foundation of China (62473189), and Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515010424.

## REFERENCES

- [1] Tom B. Lauwers, George A. Kantor, and Ralph L. Hollis. A Dynamically Stable Single-Wheeled Mobile Robot with Inverse Mouse-Ball Drive. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2884–2889, 2006.
- [2] Umashankar Nagarajan, George Kantor, and Ralph L. Hollis. The Ballbot: An Omnidirectional Balancing Mobile Robot. *The International Journal of Robotics Research*, 33(6):917–930, 2014.
- [3] Seung Yun Song, Nadja Marin, Chenzhang Xiao, Ryu Okubo, Joao Ramos, and Elizabeth T. Hsiao-Wecksler. Hands-Free Physical Human-Robot Interaction and Testing for Navigating a Virtual Ballbot. In *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 556–563, 2023.
- [4] Chenzhang Xiao, Mahshid Mansouri, David Lam, Joao Ramos, and Elizabeth T. Hsiao-Wecksler. Design and Control of a Ballbot Drivetrain with High Agility, Minimal Footprint, and High Payload. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 376–383, 2023.
- [5] Christoph Skrabel. *Mechanical Design of a Ballbot Platform*. Bachelor thesis, ETH Zurich, 2013.
- [6] Thomas Kølbæk Jespersen. *Kugle - Modelling and Control of a Ball-balancing Robot*. Bachelor thesis, Aalborg University, 2019.
- [7] Roberto Shu and Ralph Hollis. Momentum-Based Whole-Body Optimal Planning for a Single-Spherical-

- Wheeled Balancing Mobile Manipulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3221–3226, 2021.
- [8] Nathan Kau, Aaron Schultz, Natalie Ferrante, and Patrick Slade. Stanford Doggo: An Open-Source, Quasi-Direct-Drive Quadruped. In *IEEE International Conference on Robotics & Automation (ICRA)*, pages 6309–6315, 2019.
- [9] Nicolas Oros and Jeffrey L Krichmar. Smartphone Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers. Technical report, University of California, Irvine, 2013.
- [10] Thomas K Jespersen, Mohammad al Ahdab, F Mendez Juan de Dios, Malte R Damgaard, Karl D Hansen, Rasmus Pedersen, and Thomas Bak. Path-Following Model Predictive Control of Ballbots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1498–1504, 2020.
- [11] Max Studt, Ievgen Zhavzharov, and Hossam S Abbas. Parameter Identification and LQR/MPC Balancing Control of a Ballbot. In *European Control Conference (ECC)*, pages 1315–1321, 2022.
- [12] Bharat Bhushan. *Introduction to Tribology*. John Wiley & Sons, Ltd, 2013.
- [13] Farbod Farshidian, Michael Neunert, and Jonas Buchli. Learning of Closed-Loop Motion Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1441–1446, 2014.
- [14] Yifan Zhou, Jianghao Lin, Shuai Wang, and Chong Zhang. Learning Ball-Balancing Robot through Deep Reinforcement Learning. In *International Conference on Computer, Control and Robotics (ICCCR)*, pages 1–8, 2021.
- [15] Giulia Buzzetti, Davide Zappetti, and Giovanni Iacca. A Reinforcement Learning Method to Minimize the Damage on a Falling Ballbot. In *European Robotics Forum (ERF)*, pages 81–86, 2024.
- [16] Achkan Salehi. Reinforcement Learning for Ballbot Navigation in Uneven Terrain, 2025. arXiv:2505.18417.
- [17] Haoran Sun, Bangchao Huang, Zishang Zhang, Ronghan Xu, Guojing Huang, Guangyi Huang, Jiayi Yin, Nuofan Qiu, Hua Chen, Wei Zhang, Jia Pan, Fang Wan, and Chaoyang Song. Overconstrained Locomotion. In *International Symposium of Robotics Research (ISRR)*, 2024.
- [18] Yuping Gu, Bangchao Huang, Haoran Sun, Ronghan Xu, Jiayi Yin, Wei Zhang, Fang Wan, Jia Pan, and Chaoyang Song. One-DoF Robotic Design of Overconstrained Limbs with Energy-Efficient, Self-Collision-Free Motion. *Fundamental Research*, 21(5):1571, 2025.
- [19] Fang Wan and Chaoyang Song. SeeThruFinger: See and Grasp Anything with a Multi-Modal Soft Touch, 2025. arXiv:2312.09822 [cs.RO].
- [20] Xudong Han, Ning Guo, Ronghan Xu, Fang Wan, and Chaoyang Song. Anchoring Morphological Representations Unlocks Latent Proprioception in Soft Robots. *Advanced Intelligent Systems*, 7(12):e202500444, 2025.
- [21] Simon Doessegger, Peter Fankhauser, Corsin Gwerder, Jonathan Huessy, Jerome Kaeser, Thomas Kammermann, Lukas Limacher, and Michael Neunert. Rezero, Focus Project Report. Technical report, ETH Zurich, 2010.
- [22] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo Playground, 2025. arXiv:2502.08844 [cs.RO].
- [23] Andrea Bonci, Pangcheng David Cen Cheng, Marina Indri, Giacomo Nabissi, and Fiorella Sibona. Human-Robot Perception in Industrial Environments: A Survey. *Sensors*, 21(5):1571, 2021.
- [24] Tianyu Wang, Dwight Bhatt, Xiaolong Wang, and Nikolay Atanasov. Cross-Embodiment Robot Manipulation Skill Transfer using Latent Space Alignment, 2024. arXiv:2406.01968 [cs.RO].
- [25] Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling Cross-Embodied Learning: One Policy for Manipulation, Navigation, Locomotion and Aviation. In *Conference on Robot Learning (CoRL)*, 2024.
- [26] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A Human-Centered, Embodied AI Benchmark with 1,000 Everyday Activities and Realistic Simulation, 2024. arXiv:2403.09227 [cs.RO].