

ZEST: Zero-shot Embodied Skill Transfer for Athletic Robot Control

JEAN-PIERRE SLEIMAN^{1†}, HE LI^{1†}, ALPHONSUS ADU-BREDU^{2†}, ROBIN DEITS^{2†}, ARUN KUMAR^{2†}, KEVIN BERGAMIN^{2‡}, MOHAK BHARDWAJ^{2‡}, SCOTT BIDDLESTONE^{1‡}, NICOLA BURGER^{1‡}, MATTHEW A. ESTRADA^{1‡}, FRANCESCO IACOBELLI^{1‡}, TWAN KOOLEN^{2‡}, ALEXANDER LAMBERT^{2‡}, ERICA LIN^{1‡}, M. EVA MUNGAI^{1‡}, ZACH NOBLES^{1‡}, SHANE ROZEN-LEVY^{2‡}, YUYAO SHI^{1‡}, JIASHUN WANG^{1‡}, JAKOB WELNER^{2‡}, FANGZHOU YU^{1‡}, MIKE ZHANG^{1‡}, ALFRED RIZZI^{1‡‡}, JESSICA HODGINS^{1‡‡}, SYLVAIN BERTRAND^{1‡‡}, YEUVI ABE^{2‡‡}, SCOTT KUINDERSMA^{2‡‡}, FARBOD FARSHIDIAN^{1‡‡*}

¹RAI Institute, USA

²Boston Dynamics, USA

[†]Core contributors are listed in order of relative contribution.

[‡]Additional contributors are listed alphabetically.

^{‡‡}Project leads are listed in reverse order of relative contribution.

*Corresponding author. Email: ffarshidian@rai-inst.com

Achieving robust, human-like whole-body control on humanoid robots for agile, contact-rich behaviors remains a central challenge, demanding heavy per-skill engineering and a brittle process of tuning controllers. We introduce *ZEST* (Zero-shot Embodied Skill Transfer), a streamlined motion-imitation framework that trains policies via reinforcement learning from diverse sources—high-fidelity motion capture, noisy monocular video, and non-physics-constrained animation—and deploys them to hardware zero-shot. *ZEST* generalizes across behaviors and platforms while avoiding contact labels, reference or observation windows, state estimators, and extensive reward shaping. Its training pipeline combines adaptive sampling, which focuses training on difficult motion segments, and an automatic curriculum using a model-based assistive wrench, together enabling dynamic, long-horizon maneuvers. We further provide a procedure for selecting joint-level gains from approximate analytical armature values for closed-chain actuators, along with a refined model of actuators. Trained entirely in simulation with moderate domain randomization, *ZEST* demonstrates remarkable generality. On Boston Dynamics’ Atlas humanoid, *ZEST* learns dynamic, multi-contact skills (e.g., army crawl, breakdancing) from motion capture. It transfers expressive dance and scene-interaction skills, such as box-climbing, directly from videos to Atlas and the Unitree G1¹. Furthermore, it extends across morphologies to the Spot quadruped, enabling acrobatics, such as a continuous backflip, through animation. Together, these results demonstrate robust zero-shot deployment across heterogeneous data sources and embodiments, establishing *ZEST* as a scalable interface between biological movements and their robotic counterparts.

INTRODUCTION

The design of humanoid robots is motivated by a simple fact: our environments are built around the shape, scale, and motion capabilities of the human body. A robot with a similar form can, in principle, operate within these environments without requiring extensive modifications, working alongside people to perform the same everyday tasks. However, to function effectively in such settings, it must first be capable of executing the kinds of motions that humans can perform. In this context, success is measured less by proficiency in a single, task-specific behavior and more by the generality of performing a broad repertoire of coordinated, whole-body movements and skills with human-like

fluidity. However, realizing human-level physical intelligence on humanoids presents significant challenges: their high degrees of freedom, intermittent and multi-contact interactions, and inevitable modeling mismatches demand control strategies that can capture the richness of human movement while withstanding real-world uncertainty. Here, we address this grand challenge with *ZEST* (Zero-shot Embodied Skill Transfer), a unified framework for learning physical intelligence. *ZEST* translates the diversity and expressiveness of human motion from heterogeneous data sources into a robust control policy. This approach enables a legged robot to execute a vast spectrum of skills zero-shot, without any per-task fine-tuning, circumventing the need for complex behavior discovery from scratch by leveraging human motion datasets as a source of general-purpose skills.

¹The work on Boston Dynamics’ Atlas and Spot involved all authors, while authors affiliated with the RAI Institute completed the work on the Unitree G1 robot.

Over the past decade, a popular approach for enabling diverse behaviors on legged robots has been to generate whole-body motion trajectories offline and then track them with a model-based controller. A prime example is Boston Dynamics’ Atlas humanoid, which has demonstrated impressive parkour and dance routines by leveraging references derived from keyframe animation. These motions are executed using a two-layer control architecture composed of a Model Predictive Control (MPC) online planner and an optimization-based whole-body controller [1–5]. The research community has widely adopted the same model-based *plan offline, track online* paradigm across a broad range of tasks for legged platforms. This includes humanoid robots using handrails to climb stairs [6], quadrupeds showing acrobatic behaviors [7], retargeting motions from animals to robots with different morphologies [8], and quadrupedal mobile manipulators tackling multi-contact tasks [9]. Across these efforts, the core principle remains the same: use offline trajectory optimization—often in a multi-contact setting—to discover nontrivial, dynamically feasible behaviors that are difficult to design manually, and track these motions online with a general-purpose stabilizing controller. This approach offers versatility and interpretability as a well-tuned controller can track a wide range of physically consistent motions without requiring skill-specific adjustments. However, the resulting controllers tend to have stronger requirements on environment modeling, estimation, and reference fidelity, making them more difficult to deploy in scenarios involving high uncertainty or complex contact.

Recently, Reinforcement Learning (RL) has emerged as a powerful approach for control synthesis, providing exceptional agility and robustness. By learning to manage complex contact dynamics implicitly, RL policies can determine when and where to make contact in real-time, eliminating the need for predefined contact schedules and simplifying control design. Moreover, the ability to train at scale in high-fidelity simulators and transfer policies to hardware zero-shot [10, 11] has accelerated progress in data-driven control [12, 13]. This success has led to state-of-the-art performance in legged locomotion, including robust navigation over challenging terrains in the wild [14–16], high-speed running [17, 18], and extreme parkour [19–21]. Beyond locomotion, RL has also proven effective for dexterous manipulation tasks [22–24], and whole-body mobile manipulation [25–28]. For instance, it has enabled various applications in whole-body mobile manipulation, such as a quadruped to play badminton [25], a humanoid robot to perform forceful tasks such as carrying heavy loads, pulling carts, and opening spring-loaded doors [26], and a quadrupedal manipulator to dynamically push and reorient large boxes [27]. Nonetheless, these *tabula rasa* RL techniques remain sample-inefficient and highly sensitive to reward design; without careful shaping and regularization, learned policies can exploit the reward structure and exhibit unnatural or overly aggressive behaviors.

To mitigate the limitations of *tabula rasa* RL, an effective strategy, pioneered in computer graphics, is to use motion data as a prior to regularize policy learning. This data-guided recipe guides controllers toward natural movements, reducing the need for extensive reward engineering. A seminal example is *DeepMimic* [29], which trained policies to imitate a collection of Motion Capture (MoCap) clips while relying on a single, consistent reward structure. This approach was later extended to enable user-steerable control with a motion matcher [30], achieve higher-fidelity tracking of complex motions [31], and scale imitation to massive motion libraries [32]. Adversarial imitation learning replaces direct imitation rewards with those inferred indirectly by matching the distribution of demonstrations. In Adversarial Motion Priors [33], a discriminator trained to differentiate between state transitions from the dataset and those from policy rollouts provides a style reward that encourages natural motions. Subsequent variants introduce hierarchical or latent-skill formulations to mitigate mode collapse and

improve reusability and steerability [34, 35].

More recent work closes the loop by coupling kinematic motion generators with physics-based controllers. For instance, text-driven diffusion models now generate plans for multi-task sequences [36], while other hierarchical approaches tackle agile terrain traversal by iteratively augmenting motion datasets [37]. This paradigm has also been extended to complex human-object interactions through methods like multi-teacher policy distillation [38]. These works from computer graphics highlight a key theme: data—whether as explicit reference clips, adversarial priors, or learned skill embeddings—acts as a powerful regularizer that simplifies reward design, improves motion quality, and scales controllers to broad behavior repertoires.

Despite strong results in simulation, deploying a unified tracking policy on hardware remains nontrivial. The sim-to-real gap—stemming from inevitable modeling discrepancies—is worsened by contact-rich behaviors and operations near the robot’s actuation limits: many dynamic or acrobatic motions become infeasible once real torque, speed, and safety constraints are enforced. In contrast, such limits are often ignored in computer graphics settings, allowing for aggressive exploration during training, which in turn facilitates learning. Policies must also operate under partial observability (e.g., missing global pose or root linear velocity), typically requiring either a state-estimation stack or a history-conditioned policy, both of which add complexity during training and deployment. Nevertheless, a growing body of work has begun to bridge this gap by leveraging real-to-sim strategies and various data sources. In [39], Peng et al. enable quadruped robots to perform diverse gaits by imitating real animal MoCap data. Grandia et al. [40] present an RL-based policy that executes expressive, real-time stage performances driven by animation engines. *VideoMimic* [41] utilizes everyday videos to train a whole-body policy to execute contextual skills through a real-to-sim-to-real pipeline. *ASAP* [42] learns to perform agile whole-body skills by pre-training motion-tracking policies from human videos, collecting trajectories on hardware, and learning a “delta-action” model that reduces the sim-to-real gap. *KungfuBot* [43] trains a policy for Unitree G1 to perform dynamic motions, such as acrobatic dance and kung-fu strikes, through multi-step motion processing (filtering, correction, retargeting with physical constraints). *Hub* [44] pushed the limits of the same hardware to achieve extreme balance in challenging one-legged stance tasks.

Using reference trajectories from model-based planners is another strategy for imitation-based RL. Methods like *Opt-Mimic* [45] and *DTC* [46] exemplify this approach by training policies to mimic optimized plans, thereby combining the precision of trajectory optimization with the robustness of an RL tracking controller. Similarly, a two-stage RL framework can leverage a single optimal trajectory as initial guidance and then fine-tune purely for task completion, yielding robust locomotion policies [47]. For multi-contact loco-manipulation, Sleiman et al. [48] utilize a single offline demonstration to train policies for tasks like traversing spring-loaded doors and manipulating heavy dishwashers. The resulting policy generalizes beyond the reference, learning to handle object variability and discover novel recovery maneuvers.

Another line of work focuses on human-driven and generalist control, grounded in large-scale motion tracking and teacher-student distillation. *OmniH2O* [49] treats kinematic pose as a universal control interface and learns an autonomous whole-body policy by imitating a privileged teacher trained on large-scale retargeted/augmented datasets. *HOVER* [50] distills a motion imitation teacher into a student policy while applying proprioception and command masking with mode-specific and sparsity-based masks, resulting in a unified multi-mode command space for a whole-body tracking policy. Most recently, *GMT* [51] combines an adaptive sampling strategy (biasing sampling towards harder-to-learn motions) with a Mixture-of-Experts teacher that enhances expressiveness and generalizability. Student policies dis-

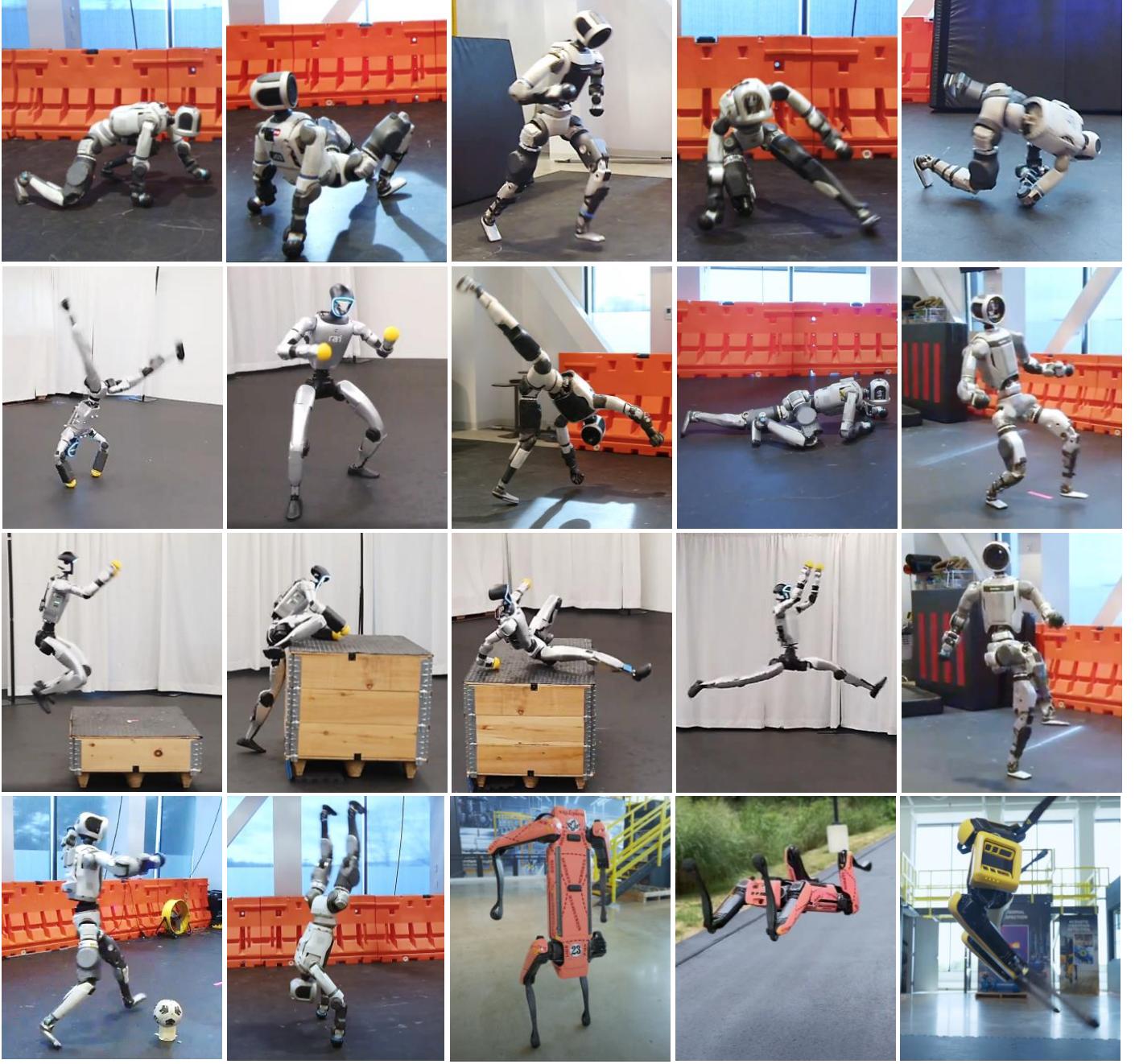


Fig. 1. Hardware deployment of ZEST across diverse data sources and robot morphologies. In order of appearance from top left to bottom right, the figure illustrates the following behaviors. From MoCap: Crawl on all fours (Atlas), roll on all fours (Atlas), jog (Atlas), breakdance (Atlas), forward roll (Atlas), cartwheel (G1), table-tennis (G1), cartwheel (atlas), army crawl (Atlas). From ViCap: Dance snippet A (Atlas), jump onto box (G1), climb up/down box (G1), ballet (G1), dance snippet C, soccer kick (Atlas). From Animation: handstand invert (Atlas), handstand balance (Spot), continuous backflip (Spot), barrel roll (Spot).

tilled from this teacher track a broad spectrum of motions achieving state-of-the-art real-world performance.

Despite these advancements, a significant obstacle to scalability and robustness persists: the dependence on intricate or multi-stage pipelines that are often composed of handcrafted components. These complexities can include auxiliary shaping reward terms, contact labeling, multi-stage training, access to non-proprioceptive signals or to histories of proprioception, and per-behavior retuning. In this work, we present *ZEST*, a unified, generic, and minimal recipe that trains policies in a single stage from heterogeneous motion data sources,

including MoCap, Video-Captured (ViCap) motions, and keyframe animations. By design, this approach eliminates complex, domain-specific techniques such as contact labeling, extensive reward shaping, future-reference windows, observation histories, and state estimators, while avoiding multi-stage training pipelines. The resulting policy, implemented with a simple feedforward network, robustly tracks diverse motion references using a single set of hyperparameters for each robot and deploys zero-shot to hardware. We apply *ZEST* to Boston Dynamics' electrically-actuated Atlas humanoid robot (approximately 1.8 m and 100 kg, comparable to an average adult human). The ex-

act formulation and training recipe are applicable across robots with different sizes, weights, and embodiments, from the full-scale Atlas humanoid to the smaller Unitree G1 humanoid and the Spot quadruped. This versatility demonstrates a level of generality that represents a significant advance over specialized controllers tightly coupled to a single robot or behavior.

Concurrent work reports closely related motion-imitation pipelines and transfer results on the G1 platform [52–54]. In this work, we evaluate a single, unified framework with a systematic, low-tuning training recipe across multiple robot platforms and multiple reference sources. This evaluation yields, to our knowledge, the first dynamic, multi-contact behaviors (e.g., army crawl, breakdance) on a full-size humanoid (Atlas) and the first video-to-behavior transfer of agile, contact-rich skills, such as box-climbing and dancing, to a physical humanoid (Unitree G1). Related work on ViCap-based sim-to-real transfer such as *VideoMimic* [41] primarily focuses on locomotion-style behaviors, whereas our video-derived demonstrations emphasize complex dynamic multi-contact skills with intermittent whole-body contact events and sustained whole-body contact interactions.

Our policies utilize the next-step reference and current proprioceptive signals as observations, with only the previous action as the history. They output residual joint targets, which are then added to the reference and sent to a joint-level PD (Proportional-Derivative) controller. To ground our simulation in reality, the PD gains are tuned using effective motor armature values derived from an analytical model that approximates closed-chain kinematics in actuators such as knees and ankles. For Spot, we incorporated more accurate models of its power system and actuators in simulation. To handle long-horizon clips and scale beyond single skills, trajectories are split into fixed-duration bins, with difficulty level estimated via an Exponential Moving Average (EMA) of failure scores. A categorical sampler biases toward harder bins, facilitating long-horizon and multi-trajectory learning without catastrophic forgetting. To help stabilize training and avoid long convergence times for highly dynamic behaviors—especially those involving large base rotations—a virtual assistive wrench is computed in a model-based fashion and applied directly to the robot’s base. Its magnitude is adapted based on the per-bin difficulty levels and eventually decays to zero as tracking performance improves, eliminating the need for a handcrafted curriculum. Training is performed entirely in simulation with moderate domain randomization (e.g., impulsive pushes, sensor noise, and variations in friction coefficients and masses). The result of this framework is a pipeline that demonstrates remarkable robustness from data to deployment. It effectively processes reference motions from diverse sources, successfully handling high-fidelity MoCap data, noisy ViCap data with artifacts like pose jitter and foot-skidding, and kinematically clean but often physically infeasible animation data. The framework’s ability to retarget motions from all three sources highlights its resilience to a wide range of data imperfections. This entire training process results in policies that are deployed to the hardware zero-shot via an automated pipeline.

RESULTS

Movie 1 summarizes the methodology and results of the presented work. In this section, we outline our evaluation procedure and present our empirical findings. Unless otherwise indicated, all presented results are obtained on hardware. Each experiment uses a specialized policy trained solely on the target skill. This specialization avoids the extended training required for a purely multi-skill model to sample the target reference as often as in the single-skill setting and, in practice, delivers stronger and more consistent tracking performance within a fixed time budget. Therefore, we do not deploy a multi-skill policy on hardware, but only use it to perform simulation-based analyses. Each policy was



Movie 1. Summary of the ZEST framework and its hardware results.

trained for approximately 10 hours (about 7k iterations) on a single NVIDIA L4 GPU and validated across multiple independent hardware trials with consistent success. Evaluations span multiple embodiments—Boston Dynamics’ electrically actuated Atlas humanoid, Unitree’s G1 humanoid, and Boston Dynamics’ Spot quadruped—and are organized by reference source: MoCap, ViCap, and keyframe animation. Figure 1 shows representative frame snapshots across all skills, per-skill clips are provided in Movie 1, and the corresponding reference motions are included in Movies S1–S3. Table 1 lists all motions, and for each, marks whether it involves multiple body contacts with the ground (i.e., apart from the robot’s feet), and reports tracking metrics for joint angles, base orientation, and base angular velocity. The base orientation is represented by its roll and pitch components. We exclude yaw, as it is not directly observable from the on-board IMU and is subject to significant drift. Next, we evaluate a multi-skill policy in simulation and use it to justify our key design choices via simulation-based ablation studies. Finally, we include a simulation-based comparison with a state-of-the-art whole-body MPC baseline developed at Boston Dynamics.

MoCap-Derived Human-Like Skills on Hardware

MoCap references enable a broad range of behaviors on Atlas (Table 1; Movie S1). As shown in Movie S1, we deploy *walk*, *jog*, *forward roll*, *roll on all fours*, *crawl on all fours*, *army crawl*, *cartwheel*, and a short *breakdance* on hardware, and we also evaluate a MoCap *table-tennis* and a *cartwheel* motion on Unitree’s G1 using the same formulation (with actuator parameters adapted to that platform).

We highlight salient aspects of these MoCap-driven behaviors. For *walk* and *jog*, foot-ground interactions are straightforward, yet several natural-gait hallmarks emerge: a clear heel-toe rollover rather than flat-foot placement; near full-knee extension despite proximity to joint limits; and graceful, quiet transitions—both within gait cycles and when decelerating from a fast jog to stance—without stomping, chatter, or audible impacts. These features are extremely challenging to induce through pure reward shaping alone. We attribute their emergence to high-fidelity MoCap references that preserve human nuance, actuator models tuned to be responsive without excessive stiffness, and regularization that discourages impulsive transients.

The same formulation also accommodates complex multi-contact behaviors without prescribing contact schedules or constraining base height or orientation to upright configurations. All motions begin from a standing configuration and return to standing at completion. *For-*

ward roll throws the body forward from stance, makes initial forearm contact, and completes a full forward roll with back contact. *Roll on all fours* and *crawl on all fours* employ intermittent heel-toe, knee, and wrist contacts; the former executes a lateral roll, while the latter advances forward under low ground clearance. *Army crawl* maintains near-continuous arm-leg-torso contact with deliberate sliding to translate forward. *Cartwheel* alternates hand-foot supports and passes through transient hand-supported inversions that require accurate hand placement and high angular-rate control. The most challenging case is the *breakdance* sequence, which combines extended body-ground contact phases with rapid leg sweeps and includes segments operated near joint limits where a leg—with its knee fully extended—threads beneath the opposite leg, inducing unavoidable self-contact.

These motions are difficult precisely because they invoke contacts beyond the feet (hands, knees, forearms, torso, back), placing them outside the design envelope of standard whole-body control pipelines. For model-based frameworks, the intricacy of modeling contact phenomena—high-impulse events, stick-slip transitions, and extended body-ground interactions over complex geometries—poses a major obstacle to designing stabilizing controllers for multi-contact behaviors. Accordingly, these methods typically predefine the active end-effectors and avoid motions with whole-body contacts. Even for RL policies trained in simulation, these regimes magnify the sim-to-real gap, as small mismatches in contact, friction, compliance, and geometry compound over extended contact-rich phases. Consistent with Table 1, simpler foot-only gaits (*walk, jog*) exhibit lower tracking errors across $\text{MAE}(q)$ and $\text{MAD}(R)$, whereas more dynamic, multi-contact skills (*forward roll, roll on all fours, crawl on all fours, army crawl, cartwheel, breakdance*) show higher orientation and angular-velocity discrepancies.

Finally, we also imitate a *cartwheel* motion and a *table-tennis* on Unitree’s G1. While our G1 evaluations are predominantly ViCap-derived motions (next section), this result provides an additional check of humanoid-to-humanoid cross-morphology transfer while holding the formulation constant across varying data sources. The G1’s *Cartwheel* reference, although approximately 2s longer than the version for Atlas, demonstrates a comparable motion quality. However, it has a higher orientation error, which we attribute to differences in the quality of the IMUs between the two platforms. Another challenging long-horizon task is the *table-tennis*, which requires coordinating upper-body swings with dynamic base maneuvers. Its extended duration makes it difficult to learn, and our adaptive sampling strategy was crucial in achieving a successful outcome for this behavior.

ViCap-Derived Human-Like Skills on Hardware

Our experiments with ViCap references highlight robustness to noisy, video-reconstructed motions while maintaining fluid, temporally coherent motion over long sequences (Movie S2). We employ video demonstrations captured in the wild with a handheld phone camera, yielding a fast and practical data generation pipeline: *record* → *reconstruct* → *retarget* → *train* → *execute*, where *reconstruct* denotes 3D pose estimation with temporal smoothing. Converting a recorded video into a retargeted reference motion takes minutes, so in practice, we can capture a new clip in the morning, train during the day, and run it on hardware that evening.

On Atlas, we execute a *soccer kick* motion and three *dance* snippets (A–C). These sequences include extended single-foot support phases and multiple consecutive hops, requiring sustained balance with precise foot placement and timing. Despite visible artifacts in the references (pose jitter and foot skidding; Movie S2), we still achieve zero-shot sim-to-real transfer while preserving expressivity, without heavy motion post-processing (e.g., physics-aware trajectory optimization for high-fidelity dynamic retargeting); as reported in Table 1, tracking errors

remain surprisingly low (e.g., $\text{MAE}(q)$ and $\text{MAD}(R)$ near 0.05 rad). We attribute this to a reward formulation that does not force tight adherence to the reference root trajectory: the policy is encouraged to track the global motion in a relaxed fashion, rejecting obvious artifacts while still solving the task. Moreover, our MDP and rewards do not require contact labels, so the policy is agnostic to noisy contact timings/schedules extracted from the ViCap reference.

We further demonstrate cross-embodiment transfer on Unitree’s G1 with an athletic *ballet sequence* featuring a sustained aerial phase with both legs abducted in mid-air and extended single-foot support near the edge of stability, stressing balance, orientation regulation, and precise foot placement. We also evaluate scene-interactive *box* skills on G1: *jump onto box, climb up box, and climb down box*. These experiments involve whole-body contacts and push the system near its operational limits. Instead of providing the policy with the box’s precise location through perception or MoCap, we test its ability to execute these skills robustly using a minimalist, proprioceptive-only interface. To achieve this, we train the policy with slight randomizations in the initial relative pose between the robot and the box. At test time, the policy reliably handles this variability without additional sensing. Furthermore, these scene-interactive skills reflect a combination of challenges, including (i) compounded modeling mismatches arising from multi-contact interactions, (ii) discrepancies between the simulated box and the physical platform (notably compliance and friction), (iii) lack of box-robot relative-positioning feedback, and (iv) artifacts in the reconstructed references. Despite these factors, hardware executions are completed reliably, and the characteristic style of the source motions remains evident. As highlighted in Table 1, errors remain reasonable overall and are comparable to those observed for the more challenging behaviors presented in the MoCap section. To further characterize repeatability and robustness on these challenging behaviors, we conduct additional variability testing. In Movie S6, we execute box climb up and box climb down five times consecutively, achieving five out of five successful runs under conditions kept as close as possible to the nominal training setup (i.e., the box height set to its nominal value of 0.75 m and the robot initialized near the nominal pose with respect to the box). In Movie S7, we stress-test the policy’s robustness by varying the initial robot-box relative pose around the nominal configuration. Since these policies are inherently invariant to lateral offsets relative to the box center, we randomize the robot’s initial offset in *x* and *yaw* while keeping *y* close to the nominal value. All trials initialized within the training distribution succeed (± 10 cm in *x* and ± 0.3 rad in *yaw*), as highlighted in Figure S5. Beyond this range, we observe occasional successes, while failures occur for initial states outside the training distribution. We additionally test robustness to mass variability by attaching a 2 kg payload to the torso and still obtain successful executions for both box climb up and box climb down. Finally, we vary the box height beyond the randomized training range. Although training uses a nominal height of 0.75 m with randomization over [0.70, 0.80] m, box climb up executes successfully at 0.55 m, while box climb down succeeds at both 0.55 m and 0.95 m.

Keyframe-Animated Skills for Humanoids and Quadrupeds on Hardware

Keyframe animation broadens our motion library beyond natural human movement. For humanoids such as Atlas, it enables us to exploit robot-specific degrees of freedom not represented in human motion capture, such as continuous joints in the arms, legs, or spine. These joints enable configurations that human anatomy cannot achieve. For instance, we achieve a *handstand invert* motion by rotating the robot’s back-yaw joint while maintaining stable hand support (Movie S3).

Keyframe animation is equally important for platforms that lack a human form, where collecting human-like demonstrations is inherently

Table 1. Evaluation across motions, grouped by reference source. A skill is labeled as *multi-contact* if any body part other than the feet interacts with the ground. Metrics are reported as *mean absolute error (MAE)*, *mean angular distance (MAD)*, and *mean L2 error (ML2)* over successful trials; lower is better. $MAE(q)$ is for joint-position error; $MAD(R)$ is for base orientation error; $ML2(\omega)$ is for base angular velocity errors, respectively. We also include data for the maximum errors in joint position and orientation, denoted as $\max(q)$ and $\max(R)$ respectively.

Motion	Robot	Multi-Contact Skill (X/✓)	Duration (s)	MAE (q) (rad)	MAD (R) (rad)	ML2 (ω) (rad/s)	$\max(q)$ (rad)	$\max(R)$ (rad)
MoCap-Derived Motions								
Walk	Atlas	X	8.47	0.057	0.030	0.364	0.487	0.078
Jog	Atlas	X	5.89	0.041	0.059	0.699	0.643	0.165
Forward roll	Atlas	✓	7.11	0.062	0.137	1.654	0.907	0.528
Roll on all fours	Atlas	✓	7.80	0.064	0.080	1.604	0.643	0.203
Crawl on all fours	Atlas	✓	14.74	0.056	0.097	0.922	0.764	0.240
Army crawl	Atlas	✓	16.66	0.103	0.075	1.011	0.638	0.276
Cartwheel	Atlas	✓	5.98	0.084	0.064	1.734	0.673	0.235
Breakdance	Atlas	✓	6.56	0.079	0.133	1.911	0.825	0.365
Cartwheel	G1	✓	8.00	0.078	0.331	0.886	0.660	0.892
Table-tennis	G1	X	29.98	0.108	0.402	1.827	2.483	0.637
ViCap-Derived Motions								
Soccer kick	Atlas	X	6.00	0.049	0.050	1.253	0.653	0.141
Dance snippet A	Atlas	X	12.70	0.051	0.039	0.414	0.769	0.080
Dance snippet B	Atlas	X	9.40	0.054	0.066	1.463	0.762	0.239
Dance snippet C	Atlas	X	11.59	0.055	0.046	1.248	0.759	0.125
Ballet sequence	G1	X	7.00	0.061	0.186	1.395	1.101	0.588
Jump onto box	G1	X	5.00	0.041	0.259	0.936	0.883	0.914
Climb up box	G1	✓	9.18	0.073	0.385	0.855	1.103	0.798
Climb down box	G1	✓	10.62	0.108	0.394	0.836	0.957	0.867
Animation-Derived Motions								
Handstand invert	Atlas	✓	5.18	0.074	0.082	0.465	0.950	0.450
Handstand balance	Spot	X	4.30	0.075	0.064	0.409	0.532	0.235
Continuous backflip	Spot	X	6.00	0.192	0.150	0.732	2.381	0.528
Barrel roll*	Spot	X	2.45	0.146	0.840*	1.451	1.068	1.571
Happy dog	Spot	X	12.00	0.107	0.069	1.008	1.180	0.349

* During barrel roll, the Spot IMU saturated mid-flight; therefore, IMU data were excluded from that segment.

difficult. To assess generality under a larger morphology gap, we apply the same framework on Boston Dynamics’ Spot, where we demonstrate a *handstand balance* (forelegs), a *continuous backflip*, a *barrel roll*, and a playful *happy dog* motion.

Although animation references are kinematically clean and temporally coherent, several sequences are dynamically aggressive or partially infeasible for the physical system. In these cases, the learned policy intentionally relaxes adherence to the reference—especially for base orientation and angular-rate profiles—to remain feasible, which increases reported means while preserving overall execution quality. This pattern is visible in Table 1 when contrasting $MAE(q)$ and $MAD(R)$ with the corresponding maxima: for Atlas *handstand invert*, the means are moderate while $\max(q)$ and $\max(R)$ spike during transient inversions; on Spot, *backflip* and *barrel roll* show larger orientation and angular-velocity errors (see $MAD(R)$ and $ML2(\omega)$) and high maxima, reflecting brief deviations from the reference for the sake of feasibility and stability. Notably, the policy’s robustness is highlighted on the Spot robot for the *barrel roll* motion, where it completes the motion even after the on-board IMU sensor saturates mid-maneuver.

Simulation-Based Evaluation and Ablation Studies

We train a single multi-skill policy for Atlas over 15 motions with diverse lengths and difficulties drawn from MoCap, ViCap, and keyframe animations, ranging from simple gaits (walking, squatting) to multi-contact and acrobatic sequences (army crawl, breakdancing, cartwheel-to-backflip). To inspect how the method operates during learning, Figure 2(a-d) shows a representative mid-training snapshot of our adaptive sampling signals: failure levels per trajectory bin, the resulting sampling probabilities, the number of times each bin has been

visited, and the assistive-wrench scaling applied when starting from that segment. Higher failure rates increase sampling probability and, over time, visitation counts rise where failures are frequent. The easiest bin is the animated squat (trajectory index 8), while the cartwheel-to-backflip sequence (trajectory index 13) is the hardest; the sampler thus allocates more training to the latter and less to the former, while modulating the assistive wrench accordingly.

Evaluation proceeds by initializing each episode at the start of a randomly sampled reference trajectory and rolling it to completion. For each reference, we run 10000 rollouts under full domain randomization, sampling external pushes, observation noise, robot link masses, friction coefficients, and perturbations of the initial state. We quantify performance and robustness via ablations presented in Figure 2(e-f), which report success rates after 10 h ($\approx 7k$ iterations) and 20 h ($\approx 14k$ iterations) of training. At 10 h, our method achieves the highest mean success and markedly better lower-tail performance across behaviors. By 20 h, several ablations narrow the mean gap, yet the baseline remains more robust, with higher p10 and minimum success values that lie closer to the mean.

We ablate, one at a time, curriculum, adaptive sampling, reference and observation windows, action space, and actor–critic observation design. Removing curriculum still converges, but its 20 h performance roughly matches our method at 10 h, indicating that a minimal variant without curriculum is feasible but less sample-efficient, particularly on motions with large base-orientation variability (e.g., trajectory index 13). Removing adaptive sampling decreases performance because the policy undersamples hard bins. The heatmaps support this mechanism, as sampling probabilities track failure levels and concentrate updates where failure is high, which the ablation cannot reproduce.

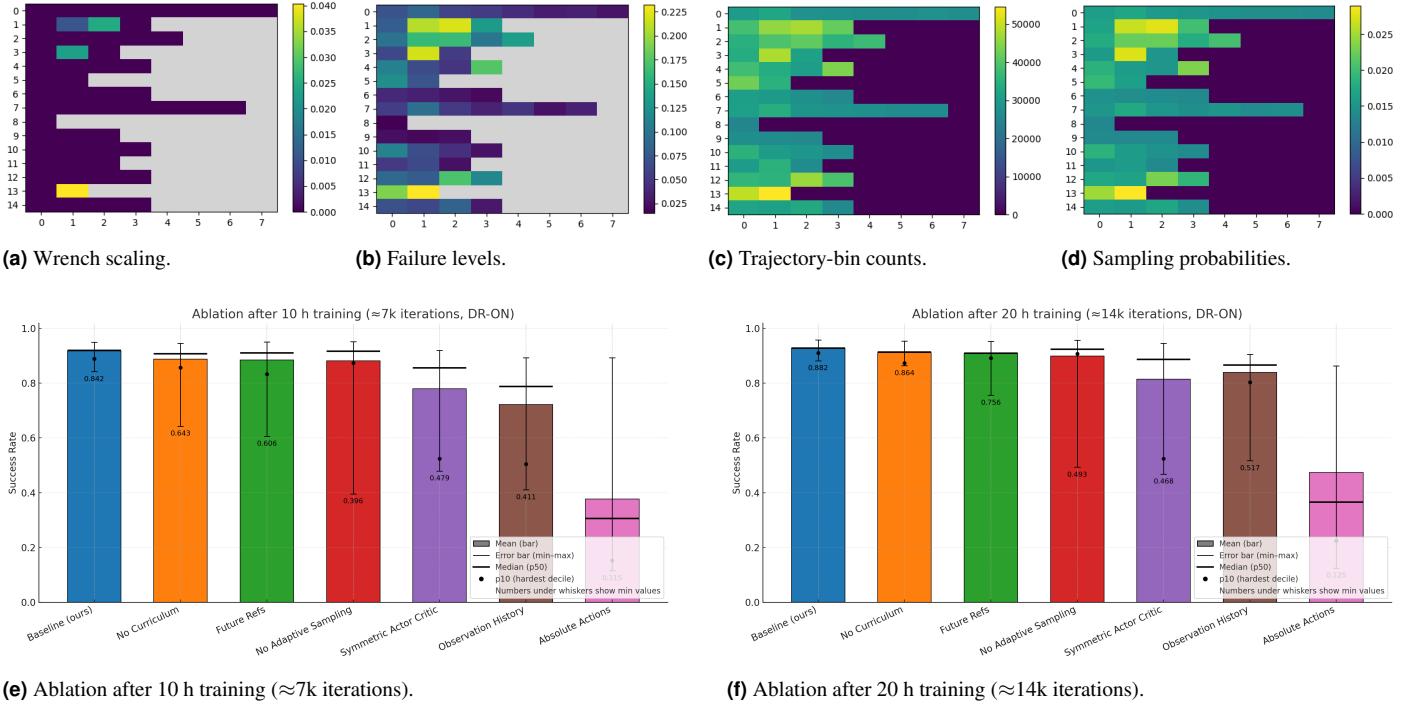


Fig. 2. Simulation-based evaluation and ablation studies. (a–d) A mid-training snapshot of the signals driving our adaptive curriculum: (a) assistive wrench scaling, which is modulated by failure, (b) per-bin failure levels, (c) bin visitation counts, and (d) sampling probabilities. Higher failure rates increase the sampling probability, and over time, visitation counts rise where failures are frequent. (e–f) Success rates for the baseline and ablations after 10 h and 20 h of training under complete domain randomization. Bars show mean success; whiskers show min–max; internal lines mark median; dots denote p10; numbers under lower whiskers report the minimum. The plots show that the assistive curriculum and adaptive sampling are critical for performance and sample efficiency. Removing privileged information from the critic or using absolute actions significantly degrades robustness, while adding observation/reference windows hinders learning.

To better contextualize the ablations over the wrench-assist curriculum and the adaptive sampling strategy, we summarize the roles played by each component. The assistive wrench primarily serves as an exploration and stabilization aid for highly dynamic motions (e.g., the cartwheel-to-backflip sequence) that frequently terminate early in training; in these regimes, it reduces catastrophic failures, stabilizes learning, and typically improves sample efficiency when reset-based exploration (i.e., Reference State Initialization) alone is insufficient. For simpler motions with fewer early failures, we find the wrench can often be reduced or removed with limited impact. Adaptive sampling is most beneficial when training on long-horizon trajectories or large motion libraries, and when the goal is to track each phase of each motion well rather than optimizing average performance. Without adaptive sampling, training can under-emphasize rare or difficult trajectory segments, leading to persistent weaknesses on those segments and degraded lower-tail performance.

Using longer reference or observation windows—either 20 steps of history or 20 steps of future references (0.4 s each)—increases input dimensionality and redundancy, complicates credit assignment, and slows or prevents convergence under the same budget. This negative result may be specific to our hyperparameters and architecture; sequence-to-sequence models or different tuning could yield different conclusions, but we did not pursue such changes to keep the setup minimal and focused. We provide a finer-grained sweep over future-reference horizons and observation-history lengths (5, 10, 15, and 20 steps) in the Supplementary Materials (Figure S3), which reveals a gradual degradation in performance as the window length increases.

Matching actor and critic inputs by stripping privileged observations from the critic degrades value estimation. The critic benefits from

privileged signals such as the full root state, contact forces, key-body positions, and the assistive-wrench signal, which reflects curriculum-induced dynamics. Without these, value estimates become noisier, leading to weaker policy updates. Finally, absolute actions (as opposed to residual actions) yield the worst performance. Because episodes are initialized on the reference throughout training, small policy outputs—which typically occur during early stages of training—can induce large PD tracking errors and hence large joint torques, destabilizing rollouts. In addition, absolute actions impose a larger exploration burden: the policy must synthesize the full joint commands, whereas residual actions leverage the feedforward reference and learn only the required offsets, leading to faster and more stable learning.

Additional details on the reference motions used for ablation are provided in the Supplementary Materials (Table S1), along with a video showcasing evaluation rollouts from the multi-skill policy (Movie S4). We further report simulation-based robustness sweeps across domain-randomization factors in the Supplementary Materials (Figure S4).

Simulation-Based Benchmarking Against MPC

We compare a multi-skill RL policy trained jointly on all Atlas motions presented in Table 1 against a whole-body MPC baseline developed at Boston Dynamics, which is representative of state-of-the-art model-based controllers [2]. Evaluations use a subset of motions drawn from all three reference sources (MoCap, ViCap, and animation). For MPC, we extract contact schedules from each reference using a simple heuristic that considers the distance between the foot and the ground, as well as the foot’s velocity. This heuristic is relatively straightforward for animated sequences (e.g., *handstand invert*), but it becomes more error-prone for MoCap and particularly ViCap motions, where unrealistic

Table 2. Simulation-based comparison of MPC vs. a multi-skill

RL policy. The RL policy is trained on all Atlas motions in Table 1, but evaluations are performed over a subset of these motions. A dash (–) indicates failure to execute the behavior.

Motion	MAE(q) (rad)		MAD(R) (rad)	
	MPC	RL	MPC	RL
Walk	0.047	0.055	0.020	0.028
Jog	0.117	0.076	0.110	0.062
Handstand invert	–	0.147	–	0.125
Cartwheel	0.237	0.088	0.060	0.063
Roll on all fours	–	0.086	–	0.091
Dance snippet B	–	0.096	–	0.150

foot sliding and noisy contact timings lead to mislabelled contacts and necessitate careful manual tuning. The MPC stack natively supports interaction with feet and hands, but not contacts beyond those end-effectors. As a result, motions that involve knee, torso, or forearm contacts (e.g., *army crawl*, *forward roll*, *crawl on all fours*) fall outside its default capability and were excluded from the MPC comparison. In contrast, the learned policy does not require explicit contact labeling, is more tolerant of model mismatch and contact-timing errors, and readily accommodates interactions with arbitrary body parts.

For a quantitative comparison, Table 2 reports MAE(q) and MAD(R) for MPC and RL. Empirically, for *walk*, MPC and RL perform nearly the same: the contact schedule is clean and easy to label, and the trajectory never pushes the robot beyond its operational bounds, which suits MPC. On the other hand, RL outperforms MPC on the more dynamic behaviors, namely *cartwheel* and *jog*. In particular, for *jog*, we observe MPC degradation tied to inconsistent contact annotation at the start of the reference despite extensive manual refinements. Several sequences fail under MPC (reported as “–” in the MPC columns): *dance B* (inaccurate contact sequence and sliding), *handstand invert* (controller saturation on arm torque under a point-contact constraint), and *roll on all fours* (difficulty regulating extended toe-contact phases without knee collisions). The videos of the references augmented with contact labels, as well as rollouts from MPC and the RL policy, are provided in Movie S5.

DISCUSSION

ZEST bridges motion datasets and robust whole-body control on legged robots. In doing so, it delivers three core contributions. First, it establishes a generic and streamlined motion-imitation recipe that trains RL policies in a single stage from heterogeneous reference sources with various imperfections (high-fidelity MoCap, noisy ViCap, and non-physics-constrained animation) and deploys it on hardware zero-shot. *ZEST* avoids complexities such as contact labels, reference or observation windows, state-estimators, extensive reward shaping, and multi-stage training. Second, our work achieves a new state of the art in demonstrated hardware capabilities across multiple embodiments. On Atlas, we demonstrate, to our knowledge, the first dynamic multi-contact behaviors on a full-size humanoid. Concurrently, on the Unitree G1, we present the first physical motion transfer of highly dynamic skills, such as dancing and box climbing, directly from video. Third, while focused on Atlas, the same policy structure and training recipe carry over to distinct morphologies such as Boston Dynamics’ Spot and Unitree’s G1. Compared to a state-of-the-art whole-body MPC baseline, which prescribes contacts only at predefined end-effectors, the learned controller eliminates the need for contact schedules and motion-specific costs, is more robust to modeling and contact-timing mismatches, and accommodates interactions with arbitrary body parts

(e.g., knees, torso, forearms).

On the other hand, several limitations remain and motivate our next steps. First, we do not evaluate the generalization of our multi-skill policy to motions outside the training set – unseen skills that are close to, but not identical with, the training distribution. Such tests would probe whether the policy is able to learn transferable control primitives as opposed to purely overfitting to specific references. We defer this generalization study to future work. Second, the current formulation is proprioceptive and assumes flat and non-slipping terrain; explicit perception of uneven or compliant environments is left for future work. Finally, sim-to-real hinges on reasonable modeling; while we provide a practical procedure involving PLA modeling and a principled selection of armature-dependent PD gains, fully automated system identification remains an open problem. Particularly challenging are complex phenomena that are difficult to capture with first-principles models, and are likely to necessitate data-driven methodologies.

Furthermore, we outline several directions that extend the scope of this work. First, we will pursue general tracking of previously unseen motions via compact motion embeddings or with zero-/few-shot adaptations, paired with continual-learning strategies to expand libraries without catastrophic forgetting. In parallel, we plan to move beyond full-reference control toward partially conditioned interfaces. One approach is to train a teacher to track a large corpus with full references and then distill the knowledge to a student that accepts sparse, human-interpretable inputs. These inputs could include keyframes, short pose snippets, object and scene keyframes, or language commands, following the masked-conditioning paradigm explored in the literature on physics-based character control [55]. Additionally, we will explore high-level generative planners that map user commands to motion plans, which a general-purpose RL-based tracker can execute, thereby closing the loop between behavior synthesis and control, as demonstrated in recent character-control systems [36, 37].

MATERIALS AND METHODS

We now present the technical details of *ZEST*. This section details its architecture, as illustrated in Figure 3, with a primary focus on the goal-conditioned MDP formulation. We describe the core components of this framework, including modeling, curation of the reference motion dataset, the training setup, and the policy deployment process on hardware.

Systems and Modeling

We evaluated *ZEST* using three robots with different morphologies: the full-scale Boston Dynamics Atlas humanoid (30 DoF, 1.8m, 100kg), the smaller Unitree G1 humanoid (29 DoF, 1.2m, 35kg), and the Boston Dynamics Spot quadruped (12 DoF, 33kg). Our RL approach uses the Isaac Lab simulator [56]. However, a primary challenge for humanoids is to efficiently model the Parallel-Linkage Actuator (PLA), which is used in ankles, knees, and waists. While a crucial design for dynamic performance, simulating their closed-chain mechanisms results in stiff dynamics that are computationally expensive to calculate. To resolve the trade-off between simulation fidelity and computational efficiency, we developed a series of progressive approximations for the PLAs. A brief overview of these models is provided below, with the approach illustrated in Figure 4. Detailed mathematical derivations are available in the Supplementary Materials (Section S2).

1. *Locally Projected Model (Massless-Links Approx.):* Our first approximation is motivated by the lightweight design of the PLA’s support links. We assume these links are massless while retaining the inertia of the motor armatures and the main kinematic chain of PLA. This yields a simulator-compatible model where the effective motor armature is configuration-dependent.

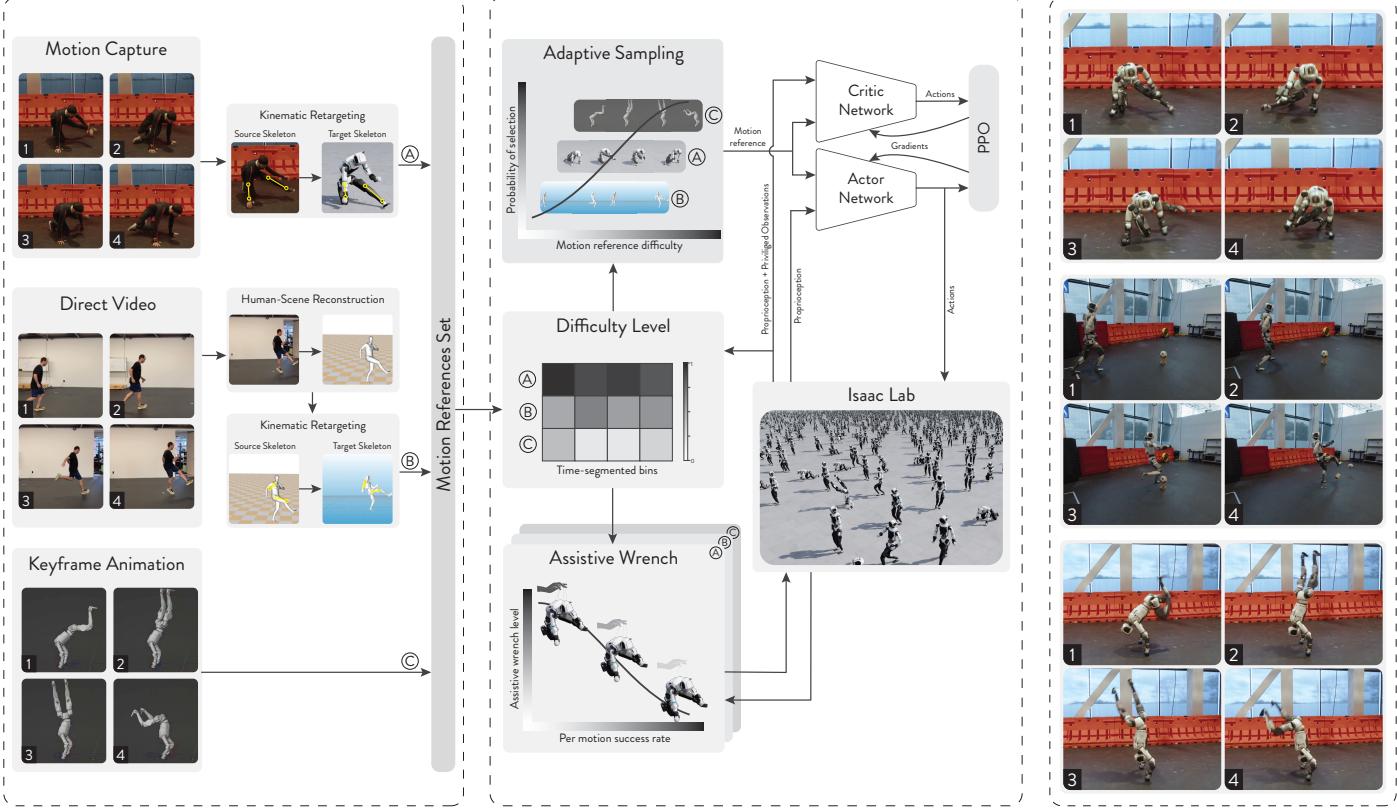


Fig. 3. Overview of ZEST, which consists of three main stages. (1) **Reference data:** A diverse set of motions from MoCap, ViCap, and keyframe animation is processed; MoCap/ViCap references are kinematically retargeted to the target robot. (2) **Training setup and MDP formulation:** In simulation, the policy is trained using only on-robot signals and the next target state from the reference, while a separate critic receives privileged information (e.g., true base velocity, contact forces) to accelerate learning. To handle long-horizon clips and scale beyond single skills, an adaptive sampling scheme is proposed: trajectories are segmented into fixed-duration bins; a per-bin difficulty level metric is updated via an EMA of failure scores; and a categorical sampler biases reset-state selection toward harder bins while avoiding catastrophic forgetting of easier behaviors. A model-based virtual assistive wrench is applied at the base to stabilize training for highly dynamic behaviors; the current bin’s difficulty level modulates its magnitude and is automatically annealed to zero as tracking improves. (3) **Zero-shot deployment:** The trained policy is deployed directly to the physical robot without any fine-tuning.

2. *Dynamic Armature Model (Diagonal Approx.):* To handle coupled PLAs (e.g., a humanoid’s ankles), which produce non-diagonal armature matrices that standard simulators cannot handle, we use a Jacobi approximation. This method approximates the off-diagonal dynamic effects, introducing only a minor, transient error.
3. *Nominal Armature Model (Fixed-Configuration Approx.):* Finally, to eliminate the computational overhead of updating armatures at every timestep ($\approx 20\%$ slowdown), we compute the armature values once at a single, nominal configuration and fix them. This provides a computationally cheap model and a principled basis for designing the robot’s PD controller gains.

While the Spot model proposed in [57] was sufficient for generating most behaviors, it proved insufficient for the more dynamic triple backflip behavior. To bridge the sim-to-real gap, we developed a more detailed actuator model that incorporates the robot’s power-limiting algorithm and simulates nonlinear effects, such as motor magnet saturation and torque losses resulting from transmission inefficiencies and friction. Static parameters of the model (e.g., rotor inertia, friction) were obtained from manufacturer specifications. The positive and negative work efficiencies were identified by optimizing the model against real-world data logged from the robot and randomly selected during

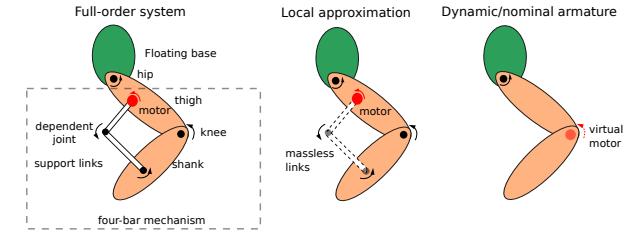


Fig. 4. Progressive simplification of proposed Parallel-Linkage Actuator models. It illustrates our modeling approach using a representative humanoid leg, where a motor on the thigh actuates the knee via a four-bar linkage. We begin with (1) **the Exact Model**, which fully resolves the closed-loop dynamics. We then introduce a series of progressively simplified models. (2) **The Locally Projected Model** assumes massless support links. (3) **The Dynamic/Nominal Armature Model** uses a Jacobi approximation for coupled joints, and finally, the most efficient model calculates armature values at a single fixed configuration.

training within ranges. See Supplementary Materials (Section S3) for derivations.

Reference Motion Pipeline

Our framework learns from a diverse library of reference motions, derived from MoCap, ViCap, and keyframe animation. All human-centric data sourced through MoCap or ViCap is mapped to the target robot's skeleton through kinematic retargeting. Crucially, we assume no contact labeling for any data source, relying solely on kinematic information. The following subsections provide a brief overview of our data sources and kinematic retargeting pipeline; comprehensive implementation details are deferred to the Supplementary Materials (Section S4).

Data Sources

MoCap data, captured using Xsens [58] and Vicon [59] systems, serves as our highest-fidelity source for natural human movements, requiring only minimal post-processing. To leverage the vast amount of motion available in casual videos, our ViCap pipeline reconstructs 3D human motion from a single camera. This process combines *MegaSaM* [60] for robust camera motion and scene estimation with *TRAM* [61] for human pose estimation, a pairing that we found significantly reduces artifacts, such as pose jitter and foot-skidding. Finally, we use Keyframe Animation to create motions that are not humanly possible or are intended for non-humanoid morphologies, such as the Spot quadruped.

Kinematic Retargeting

To transfer MoCap and ViCap data to a target robot, we solve a space-time optimization [62]. The objective function comprises several weighted costs: tracking the root position and orientation of the source motion, aligning corresponding bones via alignment frames, and regularizing the output by penalizing joint velocities. During this process, we also jointly optimize for a uniform scale and time resampling of the source data to ensure any ballistic motions are physically consistent with real-world gravity. The optimization is subject to the robot's forward kinematics and non-penetration constraints with itself and the ground.

Training Setup and MDP Formulation

We formulate *ZEST* as a goal-conditioned MDP, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, P, r, \gamma)$. The state $s_t \in \mathcal{S}$ represents the robot and environment states, and $a_t \in \mathcal{A}$ is the continuous action corresponding to motor action. At time t , the agent receives the observation, $o_t = \psi(s_t, g_t)$ and the target, $g_t \in \mathcal{G}$, selects actions $a_t \sim \pi_\theta(\cdot | o_t, g_t)$, obtains a reward $r_t = r(s_t, a_t, g_t)$, and the system evolves to a new state $s_{t+1} \sim P(\cdot | s_t, a_t)$. The goal variable g_t encodes the reference target to be tracked and can be instantiated in several forms: (i) a time-indexed reference snippet $\hat{s}_{t:t+H}$ with $H \geq 0$ (e.g., a single step or a short window); (ii) a latent embedding $z_t = f(\hat{s}_{t:t+H})$ of such a snippet; or (iii) a phase variable $\phi_t \in [0, 1]$ plus a trajectory identifier in the case of multiple reference motions. The policy parameters θ are optimized to maximize the expected discounted return over a horizon T

$$J(\theta) = \mathbb{E} \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t, g_t) \right],$$

where the expectation is over rollouts induced by P and π_θ , randomized initial states, and the goal distribution.

Training is conducted in Isaac Lab using Proximal Policy Optimization (PPO) [63] as the RL algorithm. To stabilize optimization and improve convergence, all observations are normalized using an empirical running-average scheme. The actor and critic networks are modeled as Multi-Layer Perceptrons (MLPs) with three hidden layers and ELU activations. To accelerate training while preserving the hardware deployability, we utilize an asymmetric actor-critic architecture [64]. The critic is trained using extra, privileged information available

only in simulations, which enables it to estimate the value function more accurately. In contrast, the policy is limited to the observations available during hardware deployment, ensuring it can be transferred without any state estimator. This approach prevents lag and bias from estimator fusion, sidesteps the fragile contact and terrain assumptions often required by state estimators, and eliminates the need for added complexity that is not captured during training. With the learning algorithm established, we now define the specifics of the MDP.

Actor and Critic Observations

The Actor, otherwise known as policy, receives proprioception information along with target references

$$\mathbf{o}_t = (\mathbf{o}_{\text{prop}}, \mathbf{o}_{\text{ref}}),$$

the proprioceptive part $\mathbf{o}_{\text{prop}} = ({}_{\text{T}}\omega_{\text{IT}}, {}_{\text{T}}g_I, q_j, \dot{q}_j, a_{t-1})$ contains only instantaneous onboard measurements. We intentionally exclude quantities that require a state estimator, such as global base pose or linear velocity. Here, ${}_{\text{T}}\omega_{\text{IT}}$ is the absolute angular velocity of the torso-attached IMU frame $\{\mathcal{T}\}$, ${}_{\text{T}}g_I$ is the normalized gravity vector expressed in $\{\mathcal{T}\}$, and $(q_j, \dot{q}_j, a_{t-1})$ denote joint positions, joint velocities, and the previous action. We hypothesize that including a_{t-1} is sufficient to infer essential information, such as contact states and base linear velocity, under nominal terrain conditions without resorting to long observation histories; it also provides the minimal temporal context needed to enforce action smoothness.

The reference observations $\mathbf{o}_{\text{ref}} = ({}_{\text{I}}\hat{r}_{IB}^z, {}_B\hat{v}_{IB}, {}_B\hat{\omega}_{IB}, {}_B\hat{g}_I, \hat{q}_j)$ encode the next target state extracted from the demonstration: ${}_{\text{I}}\hat{r}_{IB}^z$ is the base height; ${}_B\hat{v}_{IB}$ and ${}_B\hat{\omega}_{IB}$ are the absolute linear and angular base velocities expressed in the base frame $\{\mathcal{B}\}$; ${}_B\hat{g}_I$ is the gravity direction in $\{\mathcal{B}\}$; and \hat{q}_j are the reference joint positions. We deliberately avoid future-reference windows beyond this immediate next target as our ablations showed no benefit for performance or convergence speed.

The Critic receives the policy observations in addition to extra signals referred to as privileged information. This privileged input includes the base linear velocity, base height, end-effector positions, velocities, and contact forces, as well as curriculum-related signals. Additional details are provided in the Supplementary (Table S3).

Actions and Joint-Level Control

The policy outputs residual actions a_t , which are added to the reference joint positions before sending them as position targets to the joint-level PD controllers:

$$q_j^{\text{cmd}} = \hat{q}_j + \Sigma a_t,$$

where Σ is a positive-definite diagonal matrix of per-DoF action scales. These scales set the amplitude of residual corrections and exploration on each axis and reflect whether additional feedforward torque is needed for reference tracking: actuators that require more assistance (e.g., hips and knees) receive larger scales, whereas those adequately handled by the PD control loop alone (e.g., the robot's head and wrists) use smaller scales. The PD gains are chosen by modeling each joint as an independent second-order system:

$$I \theta''(t) + K_d \theta'(t) + K_p \theta(t) = 0,$$

where I is the nominal armature value (refer to the Modeling section) about that axis. The gains K_p and K_d are heuristically tuned to achieve a critically damped response with a desired natural frequency $\omega_n > 0$,

$$K_p = I \omega_n^2, \quad K_d = 2 I \omega_n.$$

In practice, we set ω_n to balance between responsiveness and robustness. It is set high enough for fast tracking, but not so high that the loop becomes brittle during deployment or that simulation requires very small time steps to resolve stiff dynamics with fast transients, which would in turn increase training times. The same PD gains are used in simulation and on hardware to minimize sim-to-real mismatch.

Reward Terms

The total reward combines three main components: a tracking reward, a regularization reward, and a survival reward,

$$r_{\text{total}} = r_{\text{track}} + r_{\text{reg}} + r_{\text{survival}}. \quad (1)$$

All terms are deliberately kept generic: nothing is task- or motion-specific, and the reward setup does not rely on reference contact labels from the demonstrations, making it broadly applicable across behaviors. The tracking reward measures how well the policy follows the reference motion. It can be expressed in different forms, such as a quadratic penalty or a decaying exponential kernel; in our case, we adopt the latter, which yields a dense, bounded shaping signal that emphasizes accuracy near the reference and varies smoothly with the tracking error:

$$r_{\text{track}} = \sum_i c_{t_i} \exp\left(-\kappa \frac{\|\mathbf{e}_i\|^2}{\sigma_i^2}\right), \quad (2)$$

where c_{t_i} is the reward weight for term i , and \mathbf{e}_i denotes the error between the current and reference values for quantities such as base linear velocity, base angular velocity, base pose, joint positions, and keybody poses. The regularization component, denoted r_{reg} , aggregates penalties that encourage smooth and physically feasible behavior. Concretely, we penalize rapid changes in actions across time steps, large joint accelerations, and violations of joint position and torque limits. The survival reward is a constant positive term granted at each step, $r_{\text{survival}} = c_{\text{survival}}$, which discourages premature termination by rewarding longer episodes. Full functional forms and hyperparameters for all reward terms are provided in the Supplementary Materials (Table S4).

Early Terminations

To steer exploration toward promising states and avoid wasting samples on highly undesirable trajectories, we trigger early termination under conditions such as contact forces exceeding a predefined threshold or a large deviation of the agent from the reference.

Domain Randomization

To ensure robust zero-shot sim-to-real transfer, we employ domain randomization to prevent the policy from overfitting to the nominal simulation model. Our domain randomization strategy includes injecting Gaussian noise into observations, applying random impulsive pushes to the robot at random times, varying surface friction coefficient, and slightly randomizing link masses around their nominal values. We used these randomizations carefully, as excessive variability can degrade performance by making policies overly conservative or aggressive. The details of the domain randomization's hyperparameters can be found in the Supplementary (Table S5).

Adaptive Reference State Initialization

To initialize our environment during training, we first adopt the Reference State Initialization (RSI) strategy introduced in [29]. At each reset, a random phase $\phi_{\text{init}} \sim \mathcal{U}(0, 1)$ is sampled. Then the environment is initialized at the corresponding reference state $\hat{s}(\phi_{\text{init}})$. RSI exposes the policy to relevant states along the trajectory from the outset, decouples episode length from demonstration length, and improves sample efficiency. Additionally, when a rollout reaches the end of a demonstration, we insert a brief dwell and then reset by timeout (not failure), which further increases the number of informative transitions.

While uniform RSI is generally effective, with long-horizon trajectories or large sets of trajectories, it can be inefficient because it blindly samples phases, oversampling already-mastered regions and undersampling difficult segments. To bias sampling toward where learning is most needed, we use an adaptive RSI scheme: all demonstrations

are partitioned into fixed N -step bins, and each bin b maintains a failure level f_b derived from recent tracking performance (based on a similarity score metric) and updated with an EMA filter. At reset, a bin index k is drawn from a categorical distribution whose logits are proportional to $\{f_b\}$, and a phase is then sampled uniformly within the selected bin to obtain ϕ_{init} ; a small floor probability is retained for every bin to prevent forgetting previously acquired skills. This procedure increases sampling density on difficult motions and difficult phases while preserving global coverage; additional implementation details are provided in the Supplementary (Section S5).

Assistive Wrench Automatic Curriculum

While the framework presented so far is sufficient to track most motions in our library, skills with highly dynamic base rotations and large angular rates (e.g., handstands, cartwheels, backflips) tend to trigger immediate early terminations under RSI alone and converge significantly slower compared to simpler behaviors such as walking and running. To address this, we employ an automatic assistive-wrench curriculum, serving as a continuation strategy that gradually increases task difficulty: a virtual spatial wrench is applied at the base and annealed as tracking improves, analogous to a gymnastics coach providing diminishing support to their student. Similar assistive-force curricula have been explored in previous related works [65–68]. The assistive wrench is computed in a model-based fashion using a PD term on the base pose tracking error, together with a feedforward component that compensates nominal torso dynamics. To keep assistance partially supportive while still compelling the agent to learn the skill for itself, we scale the assistive wrench by a gain $\beta \in (0, \beta_{\max})$ with $\beta_{\max} < 1.0$. The annealing schedule is driven automatically by the per-bin failure levels introduced in the adaptive RSI procedure: bins with higher failure receive stronger assistance initially and are relaxed more aggressively as tracking improves. Assistance decays quickly and ultimately vanishes once a target tracking performance is reached. Additional details are provided in the Supplementary (Section S6).

Hardware Deployment Pipeline

An important yet often overlooked aspect of embodied intelligence is the hardware deployment pipeline. Since our research involves frequent testing on physical robots, we created an efficient, automated, framework- and hardware-agnostic pipeline to ensure safe and consistent releases across multiple platforms. This pipeline follows a three-stage workflow. First, during training, policy checkpoints are exported in ONNX format and logged to a registry, along with detailed metadata that captures all relevant configurations. For validation, these artifacts are retrieved, and the metadata is used to automatically configure the controller in our *evaluation simulator* [69], which mirrors the hardware's software stack. Once a policy performs as expected, it is deployed to the physical robot using the exact same automated process, guaranteeing consistency and eliminating manual setup errors. It is worth noting that while some deployment workflow is necessary for hardware testing, the learned policy is not tied to this specific pipeline; we include it primarily to accelerate iteration and reduce incidental sources of error during sim-to-sim evaluation and hardware testing.

ACKNOWLEDGMENTS

We would like to thank Merritt Moore for the MoCap data collection. Moreover, we note that while all ideas, methods, and results are original to the authors, AI-based tools were used solely to refine the clarity and readability of the manuscript text. **Author contributions:** The *Core Contributors* designed and implemented the ZEST pipeline; trained most of the policies, and oversaw the hardware experiments on robots. They also contributed to the writing of the manuscript. The *Additional*

Contributors assisted with engineering tasks to support the project, including deployment code on hardware, data collection and curation, as well as supporting hardware experiments. The *Project Leads* set the overarching vision by defining the core scientific goals. They provided technical guidance and coordinated the team’s efforts. Finally, they helped shape the overall narrative of the paper. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper or the Supplementary Materials.

SUPPLEMENTARY MATERIALS

- Section S1. Nomenclature
- Section S2. Implementation Details: Actuator Modeling
- Section S3. Implementation Details: Spot Modeling
- Section S4. Implementation Details: Motion Dataset
- Section S5. Implementation Details: Adaptive RSI Sampling
- Section S6. Implementation Details: Assistive Wrench Curriculum
- Figure S1. Illustration of Main Chain and Support Chain
- Figure S2. Atlas Ankle Torque Limits Profile
- Figure S3. Effect of Reference and Observation Windows on Success.
- Figure S4. Robustness to Model Uncertainty and Disturbances.
- Figure S5. Robustness Maps from Varied Initial Conditions.
- Table S1. Library of Reference Motions for Multi-Skill Policy
- Table S2. Normalized MSE of Simplified PLA Models.
- Table S3. Observation Terms Summary
- Table S4. Reward Terms Summary
- Table S5. Domain Randomization Summary
- Table S6. MDP Hyperparameters
- Table S7. PPO Hyperparameters

Other Supplementary Material for this manuscript includes:

- Movie S1. Retargeted MoCap-derived References
- Movie S2. Retargeted ViCap-derived References
- Movie S3. Keyframe Animation References
- Movie S4. Simulation-based Multi-Skill Policy Evaluation
- Movie S5. Simulation-based Benchmarking against MPC
- Movie S6. Box-Climbing: Repeatability Testing on Hardware.
- Movie S7. Box-Climbing: Robustness Testing on Hardware.
- Movie S8. Impact of PLA Modeling on Sim-to-Real Transfer.

REFERENCES

1. Scott Kuindersma, Robin Deits, Maurice F. Fallon, Andres Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Auton. Robots*, 40(3):429–455, 2016. doi: 10.1007/s10514-015-9479-3.
2. Boston Dynamics. Picking up momentum. Boston Dynamics Blog, 2024. URL <https://bostondynamics.com/blog/picking-up-momentum/>. Blog post on Model Predictive Control for Atlas; accessed 2025-08-12.
3. Scott Kuindersma. Recent progress on atlas, the world’s most dynamic humanoid robot. YouTube video, 2020. URL <https://www.youtube.com/watch?v=EGABAx52GKI>.
4. Boston Dynamics. Atlas gets a grip. YouTube video, 2023. URL https://www.youtube.com/watch?v=e1_QhJ1EhQ.
5. Boston Dynamics. Hd atlas manipulates | boston dynamics. YouTube video, 2024. URL <https://www.youtube.com/watch?v=LeeiN9smjY>.
6. Manuel Kudruss, Maximilien Naveau, Olivier Stasse, Nicolas Mansard, Christian Kirches, Philippe Souères, and Katja D. Mombaur. Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations. In *IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, pages 684–689, 2015. doi: 10.1109/HUMANOIDS.2015.7363428.
7. He Li and Patrick M. Wensing. Cafe-mpc: A cascaded-fidelity model predictive control framework with tuning-free whole-body control. *IEEE Trans. Robotics*, 41:837–856, 2025. doi: 10.1109/TRO.2024.3504132.
8. Ruben Grandia, Farbod Farshidian, Espen Knoop, Christian Schumacher, Marco Hutter, and Moritz Bächer. DOC: Differentiable optimal control for retargeting motions onto legged robots. *ACM Trans. Graph.*, 42(4):96:1–96:14, 2023. doi: 10.1145/3592454.
9. Jean-Pierre Sleiman, Farbod Farshidian, and Marco Hutter. Versatile multi-contact planning and control for legged loco-manipulation. *Science Robotics*, 8(81):eadg5014, 2023. doi: 10.1126/scirobotics.adg5014.
10. Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 2021.
11. Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020. doi: 10.1109/SSCI47803.2020.9308468.
12. Sehoon Ha, Joonho Lee, Michiel van de Panne, Zhaoming Xie, Wenhao Yu, and Majid Khadiv. Learning-based legged locomotion: State of the art and future perspectives. *Int. J. Robot. Res.*, 44(8):1396–1427, 2025. doi: 10.1177/02783649241312698.
13. Zhaoyuan Gu, Junheng Li, Wenlan Shen, Wenhao Yu, Zhaoming Xie, Stephen McCrorry, Xianyi Cheng, Abdulaziz Shamsah, Robert Griffin, C Karen Liu, et al. Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning. *IEEE/ASME Trans. Mechatronics*, 2025.
14. Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986.
15. Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022. doi: 10.1126/scirobotics.abk2822.
16. Wandong Sun, Baoshi Cao, Long Chen, Yongbo Su, Yang Liu, Zongwu Xie, and Hong Liu. Learning perceptive humanoid locomotion over challenging terrain. *arXiv preprint arXiv:2503.00692*, 2025.
17. A.J. Miller, Fangzhou Yu, Michael Brauckmann, and Farbod Farshidian. High-performance reinforcement learning on spot: Optimizing simulation parameters with distributional measures. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 9981–9988, 2025. doi: 10.1109/ICRA55743.2025.11128575.
18. Guillaume Bellegarda, Yiyu Chen, Zhuochen Liu, and Quan Nguyen. Robust high-speed running for quadruped robots via deep reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10364–10370, 2022. doi: 10.1109/IROS47612.2022.9982132.
19. Xuxin Cheng, Kexin Shi, Ananya Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450, 2024. doi: 10.1109/ICRA57147.2024.10610200.
20. David Hoeller, Nikita Rudin, Dhionis V Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024. doi: 10.1126/scirobotics.adl7566.
21. Nikita Rudin, Junzhe He, Joshua Aurand, and Marco Hutter. Parkour in the wild: Learning a general and extensible agile locomotion policy using multi-expert distillation and rl fine-tuning. *arXiv preprint arXiv:2505.11164*, 2025.
22. OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Jack Hessel, Rishabh Agarwal, Mikael Henaff, Lerrel Pinto, Pieter Abbeel, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. doi: 10.1177/0278364919887447.
23. Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviychuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. DeXtreme: Transfer

- of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969, 2023. doi: 10.1109/ICRA48891.2023.10160216.
24. Toru Lin, Kartik Sachdev, Linxi Fan, Jitendra Malik, and Yuke Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. *arXiv preprint arXiv:2502.20396*, 2025.
 25. Yuntao Ma, Andrei Cramariuc, Farbod Farshidian, and Marco Hutter. Learning coordinated badminton skills for legged manipulators. *Science Robotics*, 10(102):eaud3922, 2025. doi: 10.1126/scirobotics.adu3922.
 26. Yuanhang Zhang, Yifu Yuan, Prajwal Gurunath, Tairan He, Shayegan Omidshafiei, Ali-akbar Agha-mohammadi, Marcell Vazquez-Chanlatte, Liam Pedersen, and Guanya Shi. Falcon: Learning force-adaptive humanoid loco-manipulation. *arXiv preprint arXiv:2505.06776*, 2025.
 27. Ioannis Dadiotis, Mayank Mittal, Nikos Tsagarakis, and Marco Hutter. Dynamic object goal pushing with mobile manipulators through model-free constrained reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13363–13369, 2025. doi: 10.1109/ICRA55743.2025.11128166.
 28. Clemens Schwarke, Victor Klemm, Matthijs van der Boon, Marko Bjelonic, and Marco Hutter. Curiosity-driven learning of joint locomotion and manipulation tasks. In *Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pages 2594–2610. PMLR, 2023.
 29. Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, 2018. doi: 10.1145/3197517.3201311.
 30. Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. DReCon: Data-driven responsive control of physics-based characters. *ACM Trans. Graph.*, 38(6):206:1–206:11, 2019. doi: 10.1145/3355089.3356536.
 31. Ye Yuan and Kris M. Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33*, pages 21763–21774, 2020. doi: 10.5555/3495724.3497550.
 32. Zhengyi Luo, Jinkun Cao, Alexander W. Winkler, Kris M. Kitani, and Weipeng Xu. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10895–10904, 2023. doi: 10.1109/ICCV51070.2023.01000.
 33. Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, 40(4):144:1–144:20, 2021. doi: 10.1145/3450626.3459670.
 34. Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Trans. Graph.*, 41(4):94:1–94:17, 2022. doi: 10.1145/3528223.3530110.
 35. Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. CALM: Conditional adversarial latent models for direable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. doi: 10.1145/3588432.3591541.
 36. Guy Tevet, Yoni Kasten, Chen Tessler, Moshe Gabel, Amit H. Bermano, and Daniel Cohen-Or. CLoSD: Closing the loop between simulation and diffusion for multi-task character control. In *International Conference on Learning Representations (ICLR)*, 2025.
 37. Michael Xu, Yi Shi, KangKang Yin, and Xue Bin Peng. Parc: Physics-based augmentation with reinforcement learning for character controllers. In *SIGGRAPH 2025 Conference Papers*, 2025. doi: 10.1145/3721238.3730616.
 38. Zhengyi Xu, Hung Yu Ling, Yu-Xiong Wang, and Liang-Yan Gui. Interimic: Towards universal whole-body control for physics-based human-object interactions. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.
 39. Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems (RSS)*, 2020.
 40. Ruben Grandia, Espen Knoop, Michael A. Hopkins, Georg Wiedebach, Jared Bishop, Steven Pickles, David Müller, and Moritz Bächer. Design and control of a bipedal robotic character. In *Robotics: Science and Systems (RSS)*, 2024.
 41. Arthur Allshire, Hongsuk Choi, Junyi Zhang, David McAllister, Anthony Zhang, Chung Min Kim, Trevor Darrell, Pieter Abbeel, Jitendra Malik, and Angjoo Kanazawa. Visual imitation enables contextual humanoid control. *arXiv preprint arXiv:2505.03729*, 2025.
 42. Tairan He, Jiawei Gao, Wenli Xiao, Yuanhang Zhang, Zi Wang, Jiashun Wang, Zhengyi Luo, Guanqi He, Nikhil Sobh, Chaoyi Pan, Zeji Yi, Guannan Qu, Kris M. Kitani, Jessica Hodgins, Linxi Jim Fan, Yuke Zhu, Changliu Liu, and Guanya Shi. ASAP: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. In *Robotics: Science and Systems (RSS)*, 2025. to appear.
 43. Weiji Xie, Jinrui Han, Jiakun Zheng, Huanyu Li, Xinzhe Liu, Jiyuan Shi, Weinan Zhang, Chenjia Bai, and Xuelong Li. Kungfubot: Physics-based humanoid whole-body control for learning highly-dynamic skills. *arXiv preprint arXiv:2506.12851*, 2025.
 44. Tong Zhang, Boyuan Zheng, Ruiqian Nai, Yingdong Hu, Yen-Jen Wang, Geng Chen, Fanqi Lin, Jiongye Li, Chuye Hong, Koushil Sreenath, et al. Hub: Learning extreme humanoid balance. *arXiv preprint arXiv:2505.07294*, 2025.
 45. Yukiyasu Fuchioka, Tetsuya Ogata, and Shuji Kajita. Opt-Mimic: Imitation of optimized trajectories for dynamic quadruped behaviors. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 6726–6732, 2023. doi: 10.1109/ICRA48891.2023.10161217.
 46. Fabian Jeneltén, Junzhe He, Farbod Farshidian, and Marco Hutter. Dtc: Deep tracking control. *Sci. Robotics*, 9(86):eadh5401, 2024.
 47. Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization. *Front. Robot. AI*, 9:854212, 2022. doi: 10.3389/frobt.2022.854212.
 48. Jean-Pierre Sleiman, Mayank K. Mittal, and Marco Hutter. Guided reinforcement learning for robust multi-contact loco-manipulation. In *Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pages 2210–2247. PMLR, 2024.
 49. Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris M. Kitani, Changliu Liu, and Guanya Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. In *Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pages 1516–1540. PMLR, 2024.
 50. Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Jan Kautz, Changliu Liu, Guanya Shi, Xiaolong Wang, Linxi Jim Fan, and Yuke Zhu. HOVER: Versatile neural whole-body controller for humanoid robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2025. to appear.
 51. Zixuan Chen, Mazeyu Ji, Xuxin Cheng, Xuanbin Peng, Xue Bin Peng, and Xiaolong Wang. Gmt: General motion tracking for humanoid whole-body control. *arXiv preprint arXiv:2506.14770*, 2025.
 52. Qiayuan Liao, Takara E. Truong, Xiaoyu Huang, Guy Tevet, Koushil Sreenath, and C. Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *CoRR*, abs/2508.08241, 2025. doi: 10.48550/ARXIV.2508.08241.
 53. Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C. Karen Liu, Rocky Duan, and Guanya Shi. Omnidretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. *CoRR*, abs/2509.26633, 2025. doi: 10.48550/ARXIV.2509.26633.
 54. Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Sirui Chen, Fernando Castañeda, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, Xingye Da, Runyu Ding, Cyrus Hogg, Lina Song, Edy Lim, Eugene Jeong, Tairan He, Haoru Xue, Wenli Xiao, Zi Wang, Simon Yuen, Jan Kautz, Yan Chang, Umar Iqbal, Linxi Jim Fan, and Yuke Zhu. SONIC: supersizing motion tracking for natural humanoid whole-body control. *CoRR*, abs/2511.07820, 2025. doi: 10.48550/ARXIV.2511.07820.

55. Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. Maskedmimic: Unified physics-based character control through masked motion inpainting. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 43(6), 2024. doi: 10.1145/3687951.
56. Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.
57. AJ Miller, Fangzhou Yu, Michael Brauckmann, and Farbod Farshidian. High-performance reinforcement learning on spot: Optimizing simulation parameters with distributional measures. *arXiv preprint arXiv:2504.17857*, 2025.
58. Movella Inc. Xsens link motion capture system. <https://www.movella.com/motion-capture/xsens-mvn-link>, 2025. Inertial motion capture hardware and software system.
59. Vicon Motion Systems Ltd. Vicon motion capture system. <https://www.vicon.com/>, 2025. Optical motion capture hardware and software system.
60. Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasm: Accurate, fast and robust structure and motion from casual dynamic videos. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10486–10496, 2025.
61. Yufu Wang, Ziyun Wang, Lingjie Liu, and Kostas Daniilidis. Tram: Global trajectory and motion of 3d humans from in-the-wild videos. In *European Conference on Computer Vision*, pages 467–487. Springer, 2024.
62. Michael Gleicher. Retargeting motion to new characters. In *Proc. of ACM SIGGRAPH 98, Annual Conference Series*, pages 33–42. ACM SIGGRAPH, jul 1998. doi: <http://dx.doi.org/10.1145/280814.280820>.
63. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
64. Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.
65. Andrej Karpathy and Michiel van de Panne. Curriculum learning for motor skills. In *Advances in Artificial Intelligence (Canadian AI 2012)*, volume 7310 of *Lecture Notes in Computer Science*, pages 325–330. Springer, 2012. doi: 10.1007/978-3-642-30353-1_31.
66. Wenhao Yu, Greg Turk, and C. Karen Liu. Learning symmetric and low-energy locomotion. *ACM Trans. Graph.*, 37(4):144:1–144:12, 2018. doi: 10.1145/3197517.3201397.
67. Tao Huang, Junli Ren, Huayi Wang, Zirui Wang, Qingwei Ben, Muning Wen, Xiao Chen, Jianan Li, and Jiangmiao Pang. Learning humanoid standing-up control across diverse postures. In *Robotics: Science and Systems (RSS)*, 2025. to appear.
68. Zhanxiang Cao, Yang Zhang, Buqing Nie, Huangxuan Lin, Haoyang Li, and Yue Gao. Learning motion skills with adaptive assistive curriculum force in humanoid robots. *arXiv preprint arXiv:2506.23125*, 2025.
69. Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Supplementary Material

Section S1. Nomenclature

I	Inertial (world) frame
B	Base frame
T	Torso frame
j	Joint index
kb	Keybodies
\mathcal{J}	Set of joint indices
n_j	Number of joints
n_{kb}	Number of keybodies
n_a	Number of actions
\mathbf{q}	Joint positions vector
$\dot{\mathbf{q}}$	Joint velocities vector
$\ddot{\mathbf{q}}$	Joint accelerations vector
τ	Joint torques vector
\mathbf{M}	Mass matrix
\mathbf{h}	Nonlinear forces vector
\mathbf{J}	Jacobian matrix
\mathbf{F}	Force vector
\mathbf{W}	Wrench vector
\mathbf{I}	(Base) inertia matrix
\mathcal{M}	Markov Decision Process
s_t	State at time t
a_t	Action at time t
o_t	Observation at time t
π_θ	Policy parameterized by θ
θ	Policy parameters
T	Finite horizon length
N	Number of reference trajectories
$i \in \{1, \dots, N\}$	Trajectory index
D_i	Continuous duration of trajectory i
L_i	Discrete length of trajectory i
Δ	Temporal bin width
B	Total number of bins
$b \in \{0, \dots, B-1\}$	Bin index
$M_{i,b}$	Validity mask for trajectory i in bin b
Ω	Feasible set of valid trajectory-bin pairs
$ \Omega $	Cardinality of feasible set of bins
$f_{i,b}$	Failure level for trajectory i in bin b
α	EMA smoothing factor
τ	Sampling temperature
τ_{base}	Base temperature
ε	Uniform floor weight (per-bin lower bound $\geq \varepsilon / \Omega $)
t_{init}	Initial time for episode
ϕ	Normalized phase $\in [0, 1]$

ϕ_{init}	Initial phase for episode reset
$\ell_{i,b}$	Logits for sampling
$p_{i,b}$	Sampling probabilities
$\beta_{i,b}$	Assistance scale for trajectory i in bin b
β_{\max}	Maximum assistance scale
β_e	Environment-specific assistance scale
$\hat{S}_{i,b}$	Smoothed similarity metric
η	Target similarity threshold
\mathbf{F}_b	Assistive force vector
\mathbf{M}_b	Assistive moment vector
\mathbf{w}_e	Applied assistive wrench
k_p^v, k_d^v	Virtual force PD gains
k_p^ω, k_d^ω	Virtual torque PD gains
$\mathbf{r}_{\text{b,com}}$	CoM position w.r.t. base
s_k	Instantaneous similarity metric at step k
\bar{s}_e	Length-normalized similarity metric for episode e
L_{real}	Realized episode length
L_{\max}	Maximum possible episode length
\bar{L}_{episode}	Episode length limit
r_{total}	Total reward
r_{track}	Tracking reward component
r_{reg}	Regularization reward component
r_{survival}	Survival reward component
c_{t_i}	Tracking reward weight for term i
\mathbf{e}_i	Error vector for tracking term i
σ	Standard deviation for tracking rewards
κ	Stiffness parameter for tracking rewards
dt	Control timestep
\mathbf{p}, \mathbf{v}	Position and linear velocity
Φ, ω	Orientation and angular velocity
\mathbf{g}	Gravity vector
\mathbf{r}_{XY}	Position of frame Y w.r.t. frame X
\mathbf{v}_{XY}	Velocity of frame Y w.r.t. frame X
ω_{XY}	Angular velocity of frame Y w.r.t. frame X
$x(\cdot)$	Quantity expressed in frame X
$(\hat{\cdot})$	Reference (desired) value
$(\cdot)^*$	Reference (desired) value (alternative notation)
$\mathbb{E}[\cdot]$	Expectation operator
$\ \cdot\ $	Euclidean norm
$\exp(\cdot)$	Exponential function
$\text{clip}(\cdot)$	Clipping function
$\mathcal{U}(a, b)$	Uniform distribution between a and b
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution
$\mathbf{1}\{\cdot\}$	Indicator function
$\lceil \cdot \rceil$	Ceiling function
\boxminus	Lie-group difference on $SO(3)$
\times	Cross product operator

Section S2. Implementation Details: Actuator Modeling

The PLA systems in humanoid robots are essential hardware components that help reduce limb inertia by positioning the actuators closer to the body, which enhances dynamic performance. Nonetheless, enforcing their loop-closure constraints leads to large internal forces, making the simulation of dynamics stiff and substantially increasing the computational cost of the iterative Temporal Gauss-Seidel solver used for numerical integration. Since an entirely accurate model is too slow for large-scale RL training, we tackle this trade-off between fidelity and efficiency by developing a series of progressive approximations to model the PLAs. This section details the formulation of our progressive PLA approximation models and provides an experimental validation of their accuracy. It concludes the section by showcasing the importance of considering configuration-dependent torque limits of the PLA actuators.

Projected Model

Here, we formulate an analytically exact *Projected Model* to circumvent the numerical constraint enforcement used in standard simulators. We begin by deriving the general equation of motion for a class of systems that contain a closed-kinematic chain. An instance of such a mechanism is the four-bar linkage, illustrated in Figure 4, where the primary kinematic chain consists of the floating base, thigh, shank, and their connecting joints. The PLA's actuation is provided by a motor mounted on the thigh, which drives the knee joint through a four-bar linkage. Our objective is to project the dynamics of the PLA's support links onto the main kinematic chain. To analyze the dynamics of PLA, we conceptually decompose its closed-loop structure into two overlapping open kinematic chains: a *main chain*, \mathcal{M} , and a *support chain*, \mathcal{S} , as illustrated in Figure S1.

- The *main chain*, \mathcal{M} , is comprised of the robot's primary kinematic tree, the PLA's parent link, and its unactuated branch.
- The *support chain*, \mathcal{S} , consists of the PLA's support links, which also originate from the same parent link.

A central assumption of this decomposition is that while the parent link is fully modeled with its physical mass in chain \mathcal{M} , it is considered massless in the support chain, \mathcal{S} , to avoid double-counting inertial properties. We will further assume that the velocity of the PLA joints in the support kinematic chain, including the actuated joints \mathbf{q}_d and dependent joints, \mathbf{q}_o , can be calculated based on the velocity of the PLA joints in the main kinematic chain, \mathbf{q}_o ,

$$\begin{bmatrix} \dot{\mathbf{q}}_d \\ \dot{\mathbf{q}}_i \end{bmatrix} = \begin{bmatrix} \Gamma_d(\mathbf{q}_o) \\ \Gamma_i(\mathbf{q}_o) \end{bmatrix} \dot{\mathbf{q}}_o, \quad (3)$$

where Γ_d and Γ_i denote the kinematic mappings from the main chain joint velocities to the dependent and actuated joints, respectively.

To simplify our derivation, we focus on a scenario where the parent joints are unactuated and the entire system is free from external forces, with actuation occurring only through the PLA. While extending this analysis to a more general case is straightforward, we will not cover it here to keep the discussion concise. The dynamics of the main chain and auxiliary chain can then be expressed as follows:

$$\mathbf{M}_{\mathcal{M}} \begin{bmatrix} \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_o \end{bmatrix} + \mathbf{h}_{\mathcal{M}} = \mathbf{J}_{\mathcal{M}}^T \mathbf{F} \quad (4)$$

$$\mathbf{M}_{\mathcal{S}} \begin{bmatrix} \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_d \\ \ddot{\mathbf{q}}_i \end{bmatrix} + \mathbf{h}_{\mathcal{S}} = \begin{bmatrix} 0 \\ 0 \\ \tau_i \end{bmatrix} - \mathbf{J}_{\mathcal{S}}^T \mathbf{F}, \quad (5)$$

where $\mathbf{M}_{\mathcal{M}}$ and $\mathbf{h}_{\mathcal{M}}$ are the mass matrix and the nonlinear forces of the main chain, and $\mathbf{M}_{\mathcal{S}}, \mathbf{h}_{\mathcal{S}}$ denote those of the support chain. The variable \mathbf{q}_p corresponds to the parent joints, and τ_i represents the torque applied at the PLA's actuated joint. \mathbf{F} is the generalized interaction force exchanged between the main and support chains at their connection point, and $\mathbf{J}_{\mathcal{M}}$ and $\mathbf{J}_{\mathcal{S}}$ are the corresponding Jacobians at that point. Given that the velocity of the point should be equal, we have

$$\mathbf{J}_{\mathcal{S}} \begin{bmatrix} \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_d \\ \dot{\mathbf{q}}_i \end{bmatrix} = \mathbf{J}_{\mathcal{M}} \begin{bmatrix} \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_o \end{bmatrix}.$$

Applying the mapping in 3 yields the corresponding relationship between the Jacobians

$$\mathbf{J}_{\mathcal{M}} = \mathbf{J}_{\mathcal{S}} \underbrace{\begin{bmatrix} \mathbf{I} & 0 \\ 0 & \Gamma_d(\mathbf{q}_o) \\ 0 & \Gamma_i(\mathbf{q}_o) \end{bmatrix}}_{\mathbf{G}(\mathbf{q}_o)}. \quad (6)$$

Based on 5, 4, and 6, the dynamics of the *Projected Model* can be derived as

$$(\mathbf{M}_{\mathcal{M}} + \mathbf{G}^T \mathbf{M}_{\mathcal{S}} \mathbf{G}) \begin{bmatrix} \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_o \end{bmatrix} + \mathbf{h}_{\mathcal{M}} + \mathbf{G}^T \mathbf{h}_{\mathcal{S}} + \mathbf{G}^T \mathbf{M}_{\mathcal{S}} \dot{\mathbf{G}} \begin{bmatrix} 0 \\ \dot{\mathbf{q}}_o \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_o \end{bmatrix}, \quad (7)$$

where τ_o denotes the output torque of PLA. The transmission Jacobian Γ_i defines the relationship between the input torque and the output torque of the PLA, expressed as:

$$\tau_o = \Gamma_i^\top(\mathbf{q}_o) \tau_i. \quad (8)$$

Although this model is precise, its governing equations do not adhere to the standard rigid body structure required by off-the-shelf physics engines. This structural incompatibility prevents its direct application, leading us to create subsequent approximate models that are tailored to be compatible with simulators.

Locally Projected Model: Massless-Links Approximation

Our first approximation is motivated by the lightweight design of the support links in the parallel kinematic chain of PLAs. We assume that these support links are massless while still accounting for the full inertia of the motor armatures and the main kinematic chain of PLA. By treating the support links as massless and considering the Coriolis forces, \mathbf{h}_S , to be negligible, we can derive:

$$\left(\mathbf{M}_{\mathcal{M}} + \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{M}_o \end{bmatrix} \right) \begin{bmatrix} \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_o \end{bmatrix} + \mathbf{h}_{\mathcal{M}} + \begin{bmatrix} 0 \\ \mathbf{h}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_o \end{bmatrix}, \quad (9)$$

where $\mathbf{M}_o(\mathbf{q}_o) = \Gamma_i^\top \mathbf{I}_i \Gamma_i$ is the locally projected mass matrix, under the assumption that $\mathbf{M}_S = \text{Diag}(0, \mathbf{I}_i)$ is diagonal and only encodes the PLA actuator armature values, with \mathbf{I}_i representing the motor armatures of the PLA. The term $\mathbf{h}_0(\mathbf{q}_o, \dot{\mathbf{q}}_o) = \Gamma_i^\top \mathbf{I}_i \dot{\Gamma}_i \dot{\mathbf{q}}_o$ represents the vector of nonlinear forces arising from the PLA dynamics, which acts locally on the PLA's output joints.

This model results in a valid rigid body model that is compatible with standard simulators. In this model, the armature values (inertia terms added to the diagonal of the mass matrix) depend on the configuration. In a simulator like Isaac Lab, this can be implemented by updating the armature values at each simulation step using a lookup table.

Dynamic Armature Model: Diagonal Approximation

Although the *Locally Projected Model* is local, meaning that the dynamics of the PLA actuators only affect themselves and are solely a function of the PLA's output joint, it should be noted that \mathbf{M}_o is not necessarily diagonal even though \mathbf{I}_i is diagonal. Since standard simulators cannot accommodate off-diagonal terms, this model faces limitations when applied to coupled PLAs, such as those found in the ankles of Atlas and G1, where two actuators simultaneously drive the ankle pitch and roll joints. To resolve this, we propose our second approximation, which we refer to as the *Dynamic Armature Model*. To derive this model, we decompose \mathbf{M}_o into two components: D_o , which is a diagonal matrix, and O_o , which represents the off-diagonal elements. We then define the model as follows:

$$\left(\mathbf{M}_{\mathcal{M}} + \begin{bmatrix} 0 & 0 \\ 0 & D_o \end{bmatrix} \right) \begin{bmatrix} \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_o \end{bmatrix} + \mathbf{h}_{\mathcal{M}} + \begin{bmatrix} 0 \\ \mathbf{h}_0 + O_o \ddot{\mathbf{q}}_o \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_o \end{bmatrix}. \quad (10)$$

In the next step, instead of using $\ddot{\mathbf{q}}_o$, we will use the previous joint acceleration of the PLA, denoted as $\ddot{\mathbf{q}}_o'$.

$$\left(\mathbf{M}_{\mathcal{M}} + \begin{bmatrix} 0 & 0 \\ 0 & D_o \end{bmatrix} \right) \begin{bmatrix} \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_o \end{bmatrix} + \mathbf{h}_{\mathcal{M}} + \begin{bmatrix} 0 \\ \mathbf{h}_0 + O_o \ddot{\mathbf{q}}_o' \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_o \end{bmatrix}. \quad (11)$$

The *Dynamic Armature Model* provides a Jacobi approximation of the *Locally Projected Model*, representing the PLA dynamics as two components: a configuration-dependent diagonal armature and a fictitious torque. This fictitious torque, which accounts for the off-diagonal dynamic effects, is calculated using the PLA joint acceleration from the previous timestep. This approximation introduces only a transient error, provided the system's mass matrix is diagonally dominant. We have empirically verified that this condition holds for the Atlas ankle by sweeping its whole workspace.

Nominal Armature Model: Fixed-Configuration Approximation

Thus far, our approximations have produced a compatible model with minimal error. However, the suggested configuration-dependent armature matrix creates a practical bottleneck. Updating armature values at each simulation step incurs a significant computational overhead (approximately 20% in Isaac Lab). To address this issue, we introduce our final and strongest approximation, the *Nominal Armature Model*. This model computes the joint armatures once at a single, well-chosen nominal configuration and then fixes these values throughout training, denoted by \bar{D}_o and \bar{O}_o . The dynamics of the *Nominal Armature Model* are as follows:

$$\left(\mathbf{M}_{\mathcal{M}} + \begin{bmatrix} 0 & 0 \\ 0 & \bar{D}_o \end{bmatrix} \right) \begin{bmatrix} \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_o \end{bmatrix} + \mathbf{h}_{\mathcal{M}} + \begin{bmatrix} 0 \\ \mathbf{h}_0 + \bar{O}_o \ddot{\mathbf{q}}_o' \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_o \end{bmatrix}, \quad (12)$$

This model enables adding the projected armature (of the motors on the closed kinematic chain) to the main kinematic tree when training is initialized. Crucially, this single set of nominal armature values serves a dual purpose: it simplifies the simulation dynamics and it provides a principled basis for designing the fixed PD gains of our low-level controller.

PLA Models Evaluation

To quantify the accuracy of our approximations, we compare them to the actual model. In addition to the models mentioned above, we also evaluate a baseline known as the *Simplest Model*, which excludes all fictitious torque from the *Nominal Armature* model.

The tests are conducted using the Atlas ankle, which is commanded to follow a 5 Hz sinusoidal trajectory in both pitch and roll, with an amplitude of half the joint position limit. To measure performance, we calculate the joint acceleration of each approximate model and determine the error relative to the actual model. This error is normalized at each timestep, and we report the final Mean Squared Error (MSE) for each model in Table S2.

The *Locally Projected, Dynamic Armature* and *Nominal Armature* models all demonstrate behaviors that closely match those of the exact model, indicating that the assumption of massless support links and the Jacobi approximation are valid. Although the *Simplest Model* shows relatively larger errors, these errors primarily manifest as a shift in the motion, and the resulting motion is quite similar to that of the actual model. Given that the motion of the *Simplest Model* is qualitatively similar, its simplicity makes it an attractive option, especially since the RL policy could learn to compensate for these model discrepancies. As a result, in some of the training runs, we opted to use this model.

Impact of PLA Modeling on Sim-To-Real Transfer

To better isolate the effect of PLA modeling on sim-to-real transfer, we performed a controlled comparison that holds the remainder of the pipeline fixed while varying only the armature model, which is also used to compute the joint-level PD gains. Specifically, we compare our *nominal armature model* for the Unitree G1 robot, which accounts for the parallel linkages at the waist and ankle, against an *unprojected* variant that ignores these linkages. For both models, we use the same PD gain design recipe and hyperparameters (i.e., the same natural frequency and damping ratio), and we keep all other training components identical. In simulation, the two setups exhibit similar training curves and comparable success during rollout evaluation. On hardware, however, the difference becomes clear: across three representative behaviors—namely the cartwheel motion, the ping-pong sequence, and the box climb down—the policy using the nominal armature model transfers reliably, whereas the policy trained with unprojected armatures consistently fails during execution. Qualitative comparisons are provided in Movie S8.

PLA Torque Limits

The equation Eq. (8) defines the relationship between the input torque and the output torque of the PLA. This mapping is dependent on the configuration, meaning that the output torque limits are not simply box limits, even though the input space does have box limits. An approximation approach is to use box limits. However, this can lead to either overly conservative or overly optimistic results. Figure S2 illustrates the configuration-dependent torque limits on the Atlas ankle at specific joint roll and pitch angles, compared to the fixed boundary approximation. We sweep the ankle pitch-roll joint space over its position limits. At each pitch-roll joint pair, we calculate the transmission Jacobian Γ_i (eq.Eq. (3)) and apply this linear transformation to the box limit in the motor space, resulting in a sequence of parallelograms. As shown, the fixed limits significantly underestimate the possible torque output.

Section S3. Implementation Details: Spot Modeling

To bridge the sim-to-real gap for the more challenging continuous backflip behavior, we incorporated a more accurate model of Spot’s power system and actuators in simulation. First, we modeled Spot’s power system. Spot employs a power-limiting algorithm that sums the requested mechanical power and resistive losses across all actuators and selectively saturates the commanded torque to maintain the total system power within defined limits. We replicated this algorithm in simulation so that policies would experience the same constraints during training¹. Second, we modeled the motor and transmission efficiencies. At high currents, the motor constant of an electric motor decreases. We modeled this magnet saturation effect using the following equation:

$$\tau_{out} = \frac{\tau_{in}}{1 + k|\tau_{in}|},$$

with τ_{out} denoting output torque, τ_{in} input torque, and k a derating constant. Torque losses from transmission inefficiencies and friction were modeled using the following equation:

$$\tau_{out} = \eta(\tau_{in} - I\alpha) - K_c \tanh(s\omega) - K_v\omega,$$

where I is the rotor inertia, α the joint acceleration, K_c the Coulomb friction constant, s a smoothing factor, ω the joint velocity, K_v the viscous friction constant, and η the transmission efficiency. Two values of η are used, depending on whether the motor performs positive or negative work. The complete actuator model is thus expressed as:

$$\tau_{out} = \eta \left(\frac{\tau_{in}}{1 + k|\tau_{in}|} - I\alpha \right) - K_c \tanh(s\omega) - K_v\omega.$$

The actuator model output was low-pass filtered for simulation stability. Parameters k , I , K_c , s , and K_v were sourced directly from the manufacturer’s specifications. To define the randomization range for η , we recorded joint torques during dynamic maneuvers (such as a continuous backflip) and estimated the parameters through optimization. The minimum and maximum values set the domain randomization ranges for each actuator type.

Section S4. Implementation Details: Motion Dataset

In this section, we briefly present the three data sources used in this project.

¹Further details of this algorithm cannot be provided, as they involve proprietary industrial information.

MoCap Data

MoCap was used to gather high-quality motion references of a human using a combination of Xsens and Vicon systems. Human skeletons from different data sources can vary in animation bone layout, naming, and proportions due to differences in software conventions and variations in morphology among actors. Retargeting is performed in two stages to convert these into robot trajectories. In the first stage, animation of skeletons is exported to *.bvh* format and retargeted to a target skeleton with human-like kinematics and spherical joints, but proportions similar to the target robot. Alignment frames are placed on the source skeleton bones corresponding to the head, torso, upper arms, lower arms, hands, pelvis, knees, and feet. Optimization is used to minimize spacetime constraint violations between those frames and their counterparts on the target skeleton by solving for the target skeleton’s joint poses at all timesteps. We also jointly optimize a uniform scaling of the source data, resampling time inversely with the scale to preserve a gravitational constant of 9.81 m/s^2 in ballistic motion. The second stage applies kinematic retargeting, mapping the resulting motions to the target robot.

ViCap Data

To leverage the vast amount of motion data available in online videos, we use a custom 3D motion extraction pipeline to reconstruct global human motion, including full-body poses and joint angles, from monocular video recordings (e.g., from a cellphone). While this process yields detailed, high-fidelity motion, the resulting data is prone to more artifacts than MoCap. The process consists of two main stages. In the first stage, we utilize *MegaSaM* to recover the camera trajectory in the world frame, as well as the underlying scene structure and the global metric scale. In the second stage, we estimate human motion as an *SMPL* model in the camera frame using *TRAM*, and then map these poses to world coordinates using the transformation derived from *MegaSaM*. While the original *TRAM* implementation integrates these two steps, we found that substituting the first stage with *MegaSaM* significantly reduces artifacts such as character misorientation, sliding, and floating. Finally, the extracted *SMPL* motion is retargeted to either the Atlas or G1 through kinematic retargeting. Before this process, the *SMPL* skeleton is uniformly scaled to match the target robot, with the scaling factor chosen by matching the *SMPL* thigh bone length to that of the robot.

Keyframe Animation

We utilize keyframe animation for two primary purposes: first, to create movements that are not achievable by humans, such as utilizing the continuous rotation joint of the robot; and second, to generate motions for non-humanoid morphologies, like the Spot quadruped, where collecting real-world animal data is challenging. As these motions are authored with the target robot’s kinematics in mind, they typically do not require any retargeting.

Section S5. Implementation Details: Adaptive RSI Sampler

Given a library containing N reference trajectories indexed by $i \in \{1, \dots, N\}$ with continuous durations D_i and discrete lengths L_i . We fix a temporal bin width $\Delta > 0$ and partition time into bins $b \in \{0, \dots, B-1\}$ where

$$B = \left\lceil \max_i \frac{D_i}{\Delta} \right\rceil.$$

A validity mask

$$M_{i,b} = \mathbf{1}\{b \cdot \Delta < D_i\}$$

marks bins that intersect the support of trajectory i . We define the feasible set $\Omega = \{(i, b) : M_{i,b} = 1\}$ and maintain a per-bin *failure level* matrix $f \in \mathbb{R}^{N \times B}$ on Ω . For numerical convenience, we set $f_{i,b} = -\infty$ for $(i, b) \notin \Omega$ so that invalid bins receive zero probability under a softmax. For each rollout e that starts in bin (i, b) , we compute a length-normalized similarity metric that captures overall tracking performance

$$\bar{s}_e = \frac{1}{L_{\max}} \sum_{k=1}^{L_{\text{real}}} s_k, \quad s_k \in [0, 1],$$

where s_k is given by the joint tracking reward. L_{real} is the realized episode length and $L_{\max} = \min(\bar{L}_{\text{episode}}, L_i)$ is the total number of steps that could have been realized when early terminations are not triggered (so early terminations are automatically penalized by missing terms). We then update the failure level with an exponential moving average (EMA):

$$f_{i,b} \leftarrow (1 - \alpha) f_{i,b} + \alpha (1 - \bar{s}_e), \quad \alpha \in (0, 1).$$

To sample the next initialization bin, we form logits on valid bins,

$$\ell_{i,b} = \frac{f_{i,b}}{\tau}, \quad \tau > 0,$$

and draw from the floor-smoothed categorical distribution

$$p_{i,b} = (1 - \varepsilon) \frac{e^{\ell_{i,b}}}{\sum_{(u,v) \in \Omega} e^{\ell_{u,v}}} + \varepsilon \frac{1}{|\Omega|}, \quad \varepsilon \in (0, 1),$$

with $p_{i,b} = 0$ for $(i, b) \notin \Omega$. The temperature τ controls concentration around hard bins; the uniform floor ε preserves global coverage and mitigates forgetting. Finally, given $(i, b) \sim p$, we sample a start time for trajectory i

$$t_{\text{init}} \sim \mathcal{U}(b \cdot \Delta, \min\{(b+1)\Delta, D_i\}),$$

and convert to a normalized phase $\phi = t_{\text{init}} / D_i \in [0, 1]$.

Section S6. Implementation Details: Assistive-Wrench Curriculum

We convert per-bin failure levels $f_{i,b}$ back to a smoothed similarity metric $\hat{S}_{i,b} = 1 - f_{i,b}$ on Ω and map them to a non-negative assistance scale via a monotone schedule that vanishes at a target similarity η :

$$\beta_{i,b} = \text{clip}\left(1 - \frac{\hat{S}_{i,b}}{\eta}, 0, \beta_{\max}\right), \quad \eta > 0, \beta_{\max} \in [0, 1]$$

At initialization, we set the environment's assistive gain based on its sampled trajectory-bin pair (i, b) (i.e., $\beta_e = \beta_{i,b}$) and we hold β_e constant during that episode.

Virtual wrench computation. Let $(\mathbf{p}, \mathbf{v}, \Phi, \omega)$ denote the base position, linear velocity, orientation, and angular velocity; a hat on top of each variable denotes its corresponding reference. We compute a nominal spatial wrench acting on the robot's torso as follows:

$$\mathbf{F}_b = M\left(\hat{\mathbf{v}} + k_p^v(\hat{\mathbf{p}} - \mathbf{p}) + k_d^v(\hat{\mathbf{v}} - \mathbf{v}) - \mathbf{g}\right), \quad (13a)$$

$$\mathbf{M}_b = \mathbf{I}\hat{\boldsymbol{\omega}} + k_p^\omega \mathbf{I}(\hat{\Phi} \boxminus \Phi) + k_d^\omega \mathbf{I}(\hat{\boldsymbol{\omega}} - \boldsymbol{\omega}) + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) - \mathbf{r}_{b,\text{com}} \times M\mathbf{g}, \quad (13b)$$

where M and \mathbf{I} are the whole-body mass and nominal base inertia at a default configuration, \mathbf{g} is gravity, $\mathbf{r}_{b,\text{com}}$ is the position of the whole-body CoM with respect to the base, and \boxminus denotes the Lie-group difference on $SO(3)$ (implemented via the rotation-log map for quaternions). The applied assistive wrench is

$$\mathbf{w}_e = \beta_e \begin{bmatrix} \mathbf{F}_b \\ \mathbf{M}_b \end{bmatrix},$$

expressed in the world frame and applied at the torso link. The cap $\beta_{\max} < 1$ keeps assistance partial to encourage meaningful exploration and avoid overfitting to the modified physics.

Coupling to the adaptive RSI sampler. Because $w_{i,b}$ is a deterministic function of $\hat{S}_{i,b}$, bins with persistent errors (large $f_{i,b}$, small $\hat{S}_{i,b}$) receive stronger assistance initially. As tracking improves, the EMA drives $f_{i,b}$ down, which in turn reduces $w_{i,b}$ and eventually clamps it to zero once $\hat{S}_{i,b} \geq \eta$. Simultaneously, the adaptive RSI scheme increases the sampling frequency of challenging bins until their failure levels subside while preserving coverage via the sampler's floor probability, yielding a simple, performance-coupled continuation scheme that is guaranteed to decay assistance to zero at the desired similarity threshold.

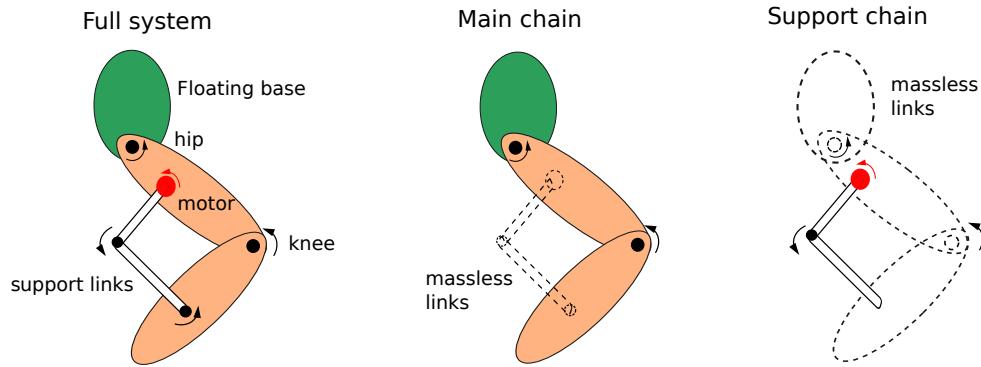


Fig. S1. Illustration of main chain and support chain. Dashed links are assumed to be massless.

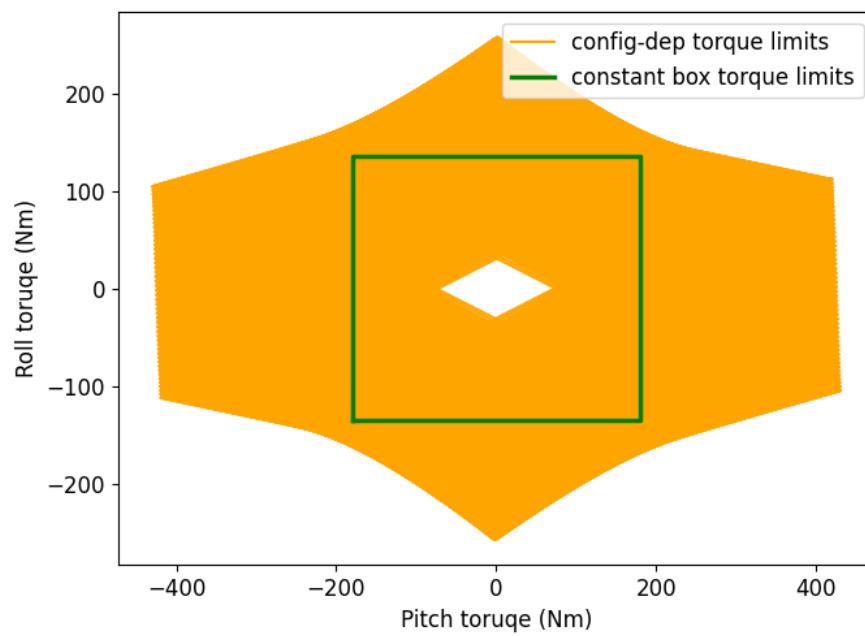


Fig. S2. Configuration-dependent torque limit versus box torque limit for Atlas ankle.

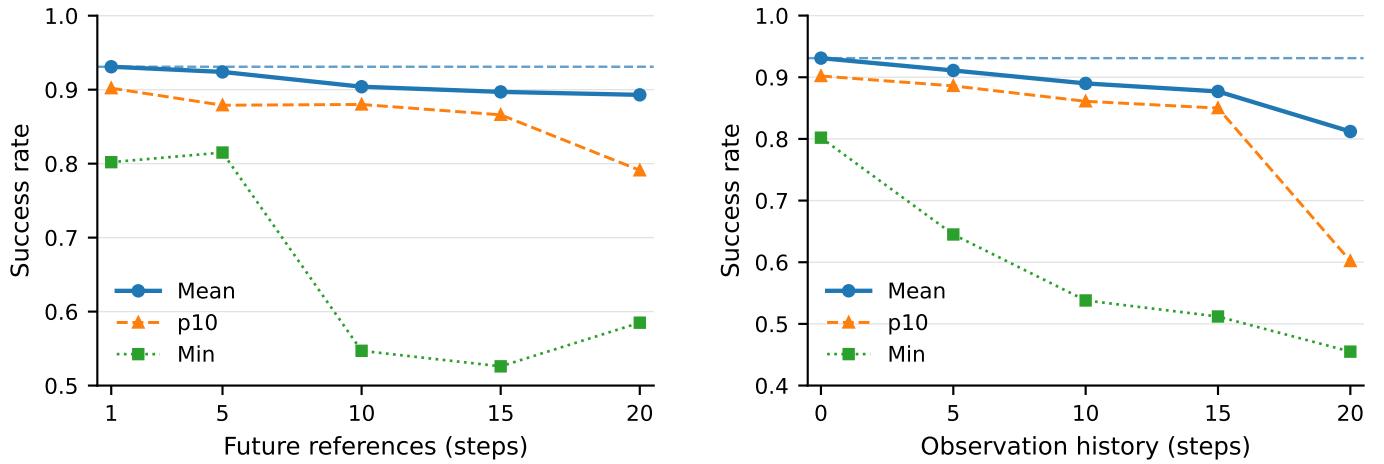


Fig. S3. Effect of future references and observation history on success. We report success statistics as a function of (left) the number of future reference steps and (right) the observation history length. Curves show the mean success rate, the 10th percentile (p10), and the minimum across multiple trajectories. Overall, increasing the future-reference horizon tends to reduce performance, with degraded lower-tail behavior (p10/min) at longer horizons. Increasing observation history is even more detrimental: performance drops more sharply, especially in the lower tail, as history length grows, indicating that stacking past observations is harmful in this setting.

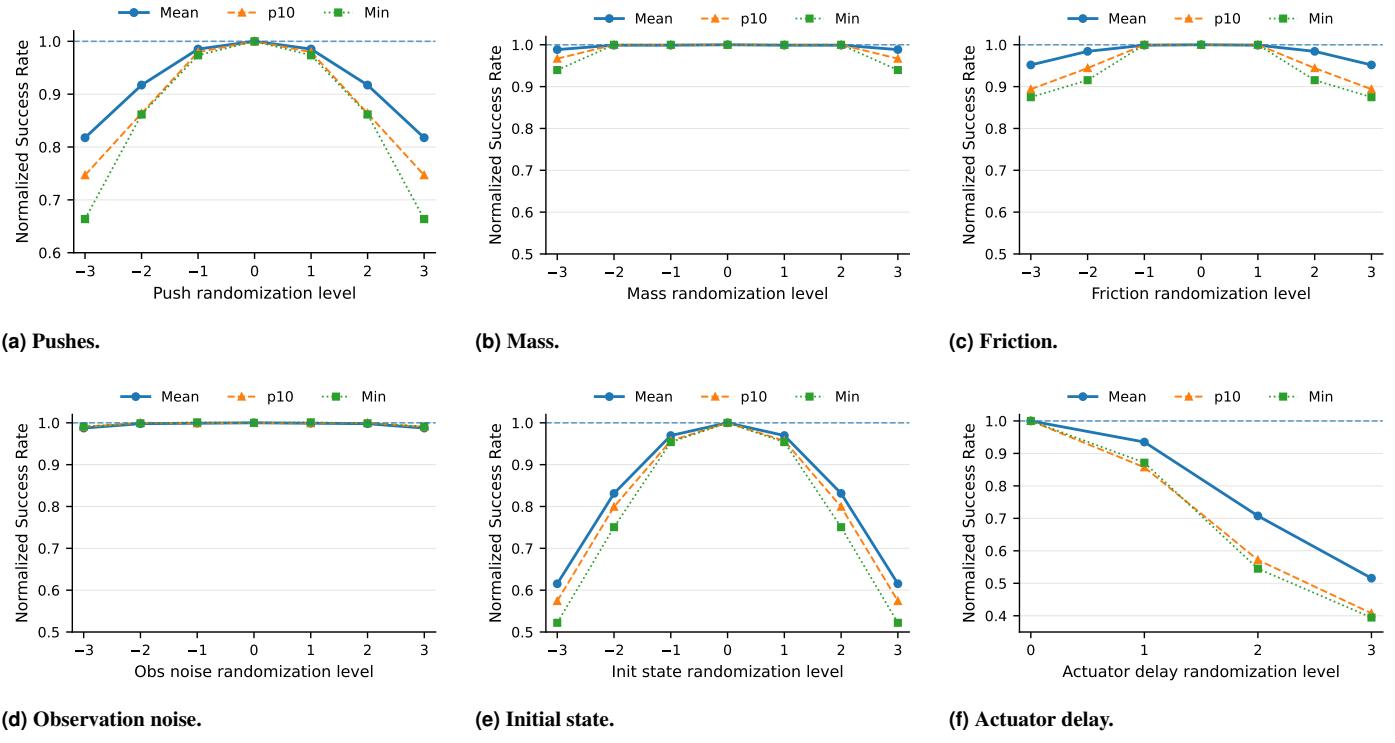


Fig. S4. Robustness to model uncertainty and disturbances. We evaluate a single policy under controlled robustness stress tests and report normalized success statistics (mean, p10, and minimum over multiple trajectories), divided by the nominal (no-randomization) performance so that the baseline at randomization level 0 equals 1.0. Unless noted otherwise, the policy was trained with randomization level 1, and levels 2–3 extrapolate beyond the training distribution. **Actuator delay** has the strongest effect in this suite; importantly, delay randomization was *not* included during training, yet performance remains acceptable for modest delays (randomization levels 1–3, corresponding to uniformly sampled delays of 0–1, 0–2, and 0–3 control steps), suggesting that other training randomizations (e.g., observation/action noise and dynamics randomization) provide indirect robustness to latency. Among the randomizations used during training, **initial-state perturbations** have the largest impact, producing the steepest drop in both mean and lower-tail performance. We attribute this primarily to the fact that aggressive initial-state perturbations can spawn the robot in self-colliding configurations (e.g., arm–torso collisions) that are effectively irreversible under the controller, leading to a higher probability of early failure. **Push disturbances** and **friction variation** exhibit the next strongest degradation, particularly in the tail (p10/min), reflecting reduced robustness under aggressive external forces and contact parameter shifts. In contrast, **mass variation** and **observation noise** have comparatively mild effects over the tested range, suggesting the learned controller is largely invariant to these sources.

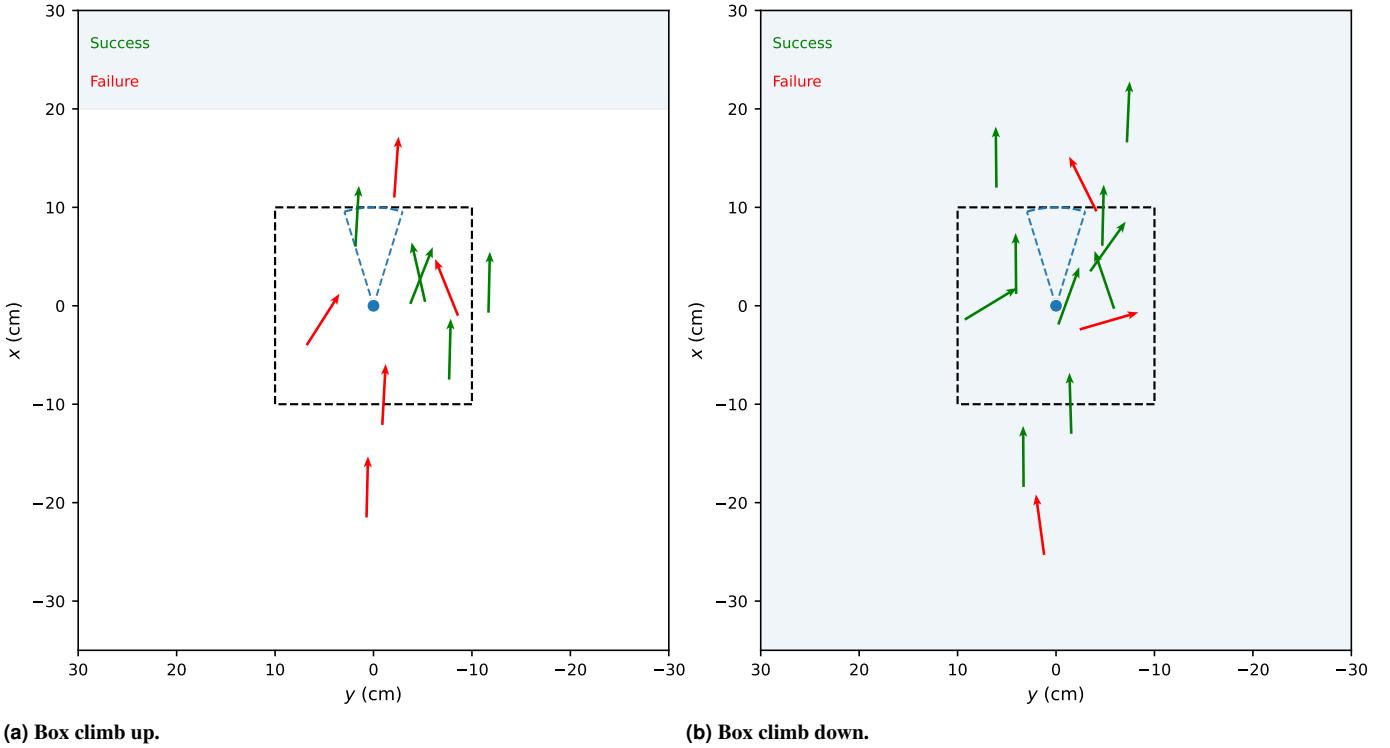


Fig. S5. Robustness maps for box-climbing policies from varied initial conditions (top-down view). Each trial is drawn at its initial planar offset ($\Delta x, \Delta y$) relative to the box, with heading indicated by a short arrow oriented by the initial yaw $\Delta\psi$; green indicates success and red indicates failure. Axes follow the manuscript convention: x points upward and y points left. The box footprint is shown as a light rectangle (80 cm along x and 120 cm along y); in this zoomed view, only the portion of the footprint within the plotted window is visible. The dashed black square indicates the training randomization range $\Delta x, \Delta y \in [-10, 10]$ cm, and the dashed blue arc indicates yaw randomization $\Delta\psi \in [-0.3, 0.3]$ rad. **Left:** nominal robot start is defined at $(0, 0)$ with the box center located 60 cm ahead along $+x$. **Right:** nominal robot start is aligned with the box center at $(0, 0)$. For both box climb up and box climb down, all trials initialized within the training distribution succeed. We also observe successes outside this range (mainly for box climb down), while failures occur only for initial conditions beyond the training distribution. *It is important to note that for visualization purposes, the y -positions were scattered by a small offset within ± 8 cm to reduce overlap (policy is inherently invariant to y).*

Table S1. Library of reference motions used for multi-skill policy evaluation and ablation studies.

Trajectory Index	Motion / Description	Data Source	Duration (s)
0	Army Crawl	MoCap	16.633
1	Dance	ViCap	9.333
2	Stylish Walk	ViCap	11.533
3	Soccer Kick	ViCap	6.633
4	Breakdance	MoCap	8.300
5	Cartwheel	MoCap	5.967
6	Crouch Walk	MoCap	8.633
7	Crawl on all Fours	MoCap	15.050
8	Deep Squat	Animation	2.992
9	Animated Walk	Animation	7.483
10	Kneeling	MoCap	8.383
11	Run	MoCap	6.475
12	Lightsaber Routine	Animation	9.400
13	Cartwheel–Backflip	MoCap	5.867
14	Roll on all Fours	MoCap	8.800

Table S2. Normalized mean-squared errors of simplified PLA models. Errors are calculated as the differences of the joint accelerations between simplified models and the exact model, and are normalized over the joint accelerations of the exact model.

	Ankle Pitch	Ankle Roll
<i>Locally Projected Model</i>	$9.527e^{-4}$	$3.418e^{-4}$
<i>Dynamic Armature Model</i>	$1.296e^{-3}$	$7.902e^{-4}$
<i>Nominal Armature Model</i>	$1.220e^{-3}$	$9.391e^{-4}$
<i>Simplest Model</i>	0.0776	0.8673

Table S3. Observation Terms Summary. We do not perform any scaling or clipping on individual observation terms. All noise models are zero-mean Gaussian and additive in nature. Privileged information is used by the critic only.

Term Name	Definition	Noise	Dim.
Proprioceptive Observations			
Torso angular velocity	$T\omega_{IT}$	$\mathcal{N}(0, 0.10^2)$	3
Gravity vector in torso frame	Tg_I	$\mathcal{N}(0, 0.015^2)$	3
Joint positions	q_j	$\mathcal{N}(0, 0.005^2)$	n_j
Joint velocities	\dot{q}_j	$\mathcal{N}(0, 0.25^2)$	n_j
Previous action	a_{t-1}	—	n_a
Reference Observations			
Base height	$I\hat{r}_{IB}^z$	—	1
Base linear velocity	$B\hat{v}_{IB}$	—	3
Base angular velocity	$B\hat{\omega}_{IB}$	—	3
Gravity vector in base frame	$B\hat{g}_I$	—	3
Joint positions	\hat{q}_j	—	n_j
Privileged Information (critic only)			
Base linear velocity	Bv_{IB}	—	3
Base height	Ir_{IB}^z	—	1
Base contact force	Bf_{contact}	—	3
Keybodies contact forces	$\{f_i\}_{i=1}^{n_{kb}}$	—	$3 \times n_{kb}$
Keybody positions w.r.t. base	Br_{BK}	—	$3 \times n_{kb}$
Keybody linear velocities	Bv_{IK}	—	$3 \times n_{kb}$
Fictitious force (assistive wrench)	f_{assist}	—	3
Fictitious torque (assistive wrench)	τ_{assist}	—	3
Wrench scale	β	—	1
Tracking rewards	r_{track}	—	7
Task phase	ϕ	—	1

Table S4. Reward Terms Summary. The environment scales the reward weights with the time-step dt . For brevity, the time index t is omitted unless needed. Keybody positions and orientations are expressed w.r.t. the base frame. A common stiffness parameter of $\kappa = 1/4$ is set for all tracking reward terms.

Term Name	Definition	Weight	σ_i
Tracking Reward			
Base position tracking	$\exp(-\kappa \ \mathbf{r}_{IB} - \mathbf{r}_{IB}^*\ ^2 / \sigma_1^2)$	1	0.4
Base orientation	$\exp(-\kappa \ \Phi_{IB} \boxminus \Phi_{IB}^*\ ^2 / \sigma_2^2)$	1	0.5
Base angular velocity	$\exp(-\kappa \ \boldsymbol{\omega}_{IB} - \boldsymbol{\omega}_{IB}^*\ ^2 / \sigma_3^2)$	1	1.5
Base linear velocity	$\exp(-\kappa \ \mathbf{v}_{IB} - \mathbf{v}_{IB}^*\ ^2 / \sigma_4^2)$	1	0.6
Joint position	$\exp(-\kappa \ q_j - q_j^*\ ^2 / \sigma_5^2)$	1	$0.3 \cdot \sqrt{n_j}$
Keybodies position	$\exp(-\kappa \ r_{kb} - r_{kb}^*\ ^2 / \sigma_6^2)$	1	$0.2 \cdot \sqrt{n_{kb}}$
Keybodies orientation	$\exp(-\kappa \ \Phi_{kb} \boxminus \Phi_{kb}^*\ ^2 / \sigma_7^2)$	1	$0.4 \cdot \sqrt{n_{kb}}$
Regularization Penalty			
Action smoothness	$-\ \mathbf{a}_t - \mathbf{a}_{t-1}\ $	0.15	–
Joint acceleration	$-\ \ddot{q}_j\ $	1×10^{-5}	–
Joint position limit	$-\sum_{i \in \mathcal{J}} [\max(0, q_i^{\min} - q_i) + \max(0, q_i - q_i^{\max})]$	1.0	–
Joint torque limit	$-\sum_{i \in \mathcal{J}} [\max(0, \tau_i^{\min} - \tau_i) + \max(0, \tau_i - \tau_i^{\max})]$	0.1	–
Survival Reward			
Survival	1	1	–

Table S5. Domain randomization terms including dynamics randomization and external perturbations.

Term	Value
Static friction	$\mathcal{U}(0.6, 1.0)$
Dynamic friction	$\mathcal{U}(0.5, 0.9)$
Restitution	$\mathcal{U}(0.0, 0.2)$
Link masses	$\mathcal{U}(0.9, 1.1) \times \text{default masses}$
External disturbance (impulsive push applied to base)	Interval = $\mathcal{U}(0.000 \text{ s}, 10.000 \text{ s})$, $v_{xy} = 0.500 \text{ m s}^{-1}$

Table S6. MDP Hyperparameters.

Hyperparameter	Value
Episode length (\bar{L}_{episode})	10.000 s
Simulation time-step (dt)	0.004 s
Control decimation	5
Action scale (per joint)	{0.05, 0.10, 0.20}
Bin width (Δ)	$\min(4.000 \text{ s}, \min_i D_i)$
EMA alpha (α)	0.005
Base temperature (τ_{base})	1.0
Adaptive sampling temperature (τ)	$\frac{\tau_{\text{base}}}{\log(1 + \Omega)}$
Uniform floor weight (ε)	0.15
Max wrench scale (β_{\max})	0.60
Similarity threshold (η)	0.80
Virtual torque PD gains (k_p^ω, k_d^ω)	(200.0, 10.0)
Virtual force PD gains (k_p^v, k_d^v)	(0.0, 10.0)

Table S7. PPO Hyperparameters

Hyperparameter	Value
Actor Network	MLP(512, 256, 128) with ELU activation
Critic Network	MLP(512, 512, 256) with ELU activation
Empirical Normalization	True
Learning Rate (start of training)	1e-3
Learning Rate Schedule	“adaptive” (based on KL-divergence)
Discount Factor	0.99
GAE Discount Factor	0.95
Desired KL-divergence	0.01
Clip Range	0.2
Entropy Coefficient	0.001
Value Function Loss Coefficient	0.5
Number of Epochs	5
Number of Environments	4096
Batch Size	245,760 (4096 × 24)
Mini-Batch Size	61,440 (4096 × 6)