Residual Dynamics for Reducing the Simulation-to-Real Gap in Zero-Shot Humanoid Skill Deployment

Thanh Truong[1], Hyeonju Kim[1], JungHwan Lee[1], JaeHong Lee[1], Anh Nguyen[1], Marsha Aqila[1]
Hansol Kang[2] and Hyouk Ryeol Choi[3]

***Abstract*— Bridging the sim-to-real gap remains a key challenge in deploying reinforcement learning (RL) policies on legged and humanoid robots. Many existing methods depend on motion capture, torque sensors, or detailed actuator models, which limit practicality and scalability. We propose a minimal-sensing RL framework that learns residual dynamics between simulation and hardware using only joint state trajectories from the real robot. A policy is trained in simulation to output corrective torque residuals, added to a PD controller, so the simulator matches recorded real trajectories. Unlike mocap-based or torque-sensing approaches, our method relies solely on proprioceptive signals. Preliminary results show that residual dynamics policies reduce sim-to-real mismatch and generate meaningful torque corrections for zero-shot humanoid skill deployment without costly sensing infrastructure.**

I. INTRODUCTION

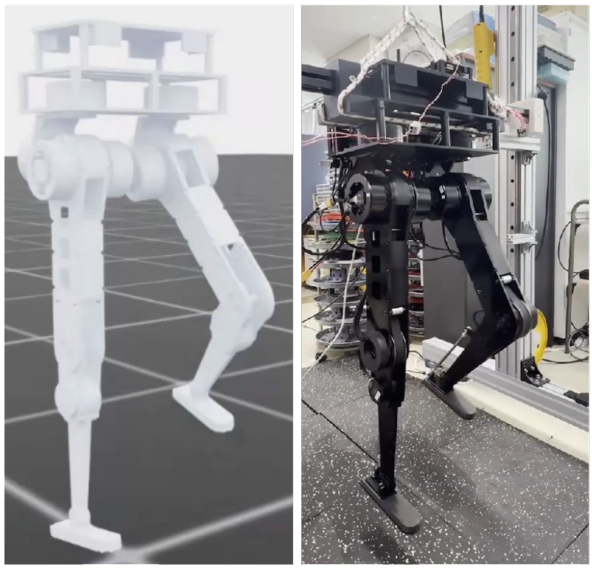
Legged and humanoid robots have recently achieved remarkable progress in agile locomotion with deep reinforcement learning (RL). Yet, transferring policies to real hardware remains challenging due to dynamics mismatches from friction, actuator compliance, sensing delays, and contact uncertainties.

Prior approaches reduce this gap by learning delta actions from motion-capture data (ASAP [1]) or actuator dynamics from torque sensing [2]. While effective, these methods require costly hardware unavailable to many labs.

We propose an alternative: record joint trajectories under PD control, replay them in simulation, and train an RL policy that generates residual torques to correct unmodeled dynamics. Inspired by RL-based system identification [3], our method relies only on proprioceptive signals, making it practical for humanoid platforms without motion capture or torque sensors. Our main contributions could be summarized

- **Residual Dynamics Formulation** – RL-based compensation of sim-to-real mismatch using only joint states, without mocap or torque sensors. Experiments demonstrate that our framework effectively reduces dynamics mismatch, enabling highly agile motions on robot.
- **Open Source** – A codebase built on IsaacLab/Isaac Sim to support further research.

(Corresponding author: Hyouk Ryeol Choi)
[1]Department of Intelligent Robotics, Sungkyunkwan University, Suwon, South Korea {thanhtd9, bornagainljh, fgfg0203, ljh0649, hoanganh2119, marshaqila}@g.skku.edu
[2]AIDIN ROBOTICS Inc., Anyang 14055, South Korea leevsv@gmail.com
[3]Department of Mechanical Engineering, Sungkyunkwan University, Suwon, South Korea choihyoukryeol@gmail.com



a) b)

Fig. 1. (a) Nominal simulated robot compensated by proposed framework. (b) Our fabricated real robot approximated by simulation robot model.

II. RESIDUAL DYNAMIC LEARNING SYSTEM

The proposed framework compensates for dynamics mismatch between simulation and hardware in two stages (Fig. 2). First, joint trajectories are executed on the real robot with a PD controller, and the resulting states are recorded. Second, these data are replayed in simulation to train an RL-based residual dynamics model that applies corrective torques to align simulated and real joint states.

A. Data Acquisition

To capture dynamics above 100 Hz, excitation must span a broad frequency range to avoid oscillations [1]. We use elliptical foot trajectories on an ellipsoidal surface (2), compute joint positions via inverse kinematics, and execute them on the robot under PD control. The resulting joint states are recorded at 500 Hz.

B. Training Residual Dynamic Model

Policy Observations. In the simulation environment, the observation s_t fed into the residual dynamic model are divided into two parts: joint states, $\{q_t, \dot{q}_t, \ddot{q}_t\}$, and joint commands, q_t^{cmd}. These inputs are inspired by the dynamic model (1) and are designed so that the residual torque output,

τ_{res}, can compensate for unmodeled dynamics, including joint friction and mass matrix mismatches $f(q, \dot{q}, \ddot{q})$.

$$M(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + G(q) + f(q,\dot{q},\ddot{q}) = \tau_{pd} + \tau_{res} \quad (1)$$

$$\begin{aligned}
\phi_t &= -\tfrac{A_\phi}{\omega_\phi}\cos(\omega_\phi t) + \tfrac{A_\phi}{\omega_\phi} + \phi_0, \\
\theta_t &= -\tfrac{A_\theta}{\omega_\theta}\cos(\omega_\theta t) + \tfrac{A_\theta}{\omega_\theta} + \theta_0, \\
x_t &= a\cos(\theta_t)\cos(\phi_t) + x_0, \\
y_t &= b\cos(\theta_t)\sin(\phi_t) + y_0, \\
z_t &= c\sin(\theta_t) + z_0
\end{aligned} \quad (2)$$

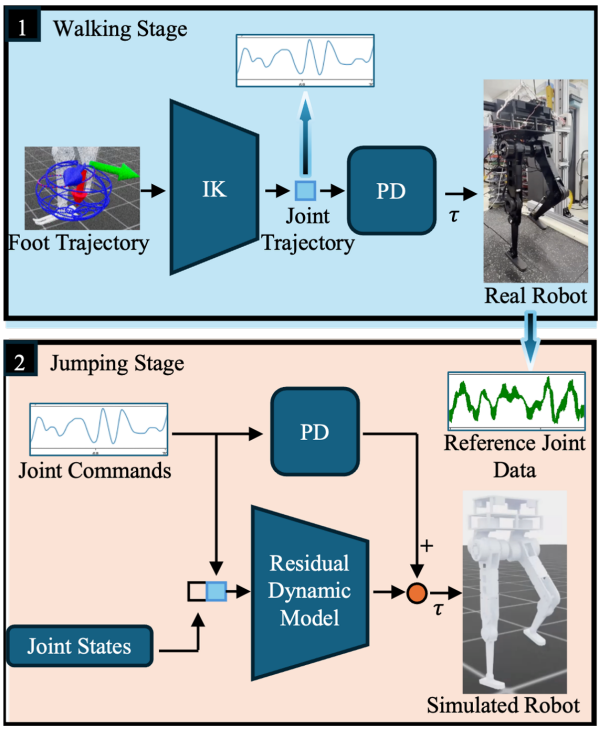


Fig. 2. Training pipeline for residual dynamic model.

Policy Representation. The policy architecture (Fig. 2) employs MLPs with ELU activations to encode joint states and control inputs into torque actions. In addition, to address actuator and sensor delays, could consider a GRU-based memory module can be integrated with the MLPs.

Rewards. As illustrated in Fig. 2, the second training stage forces the simulated robot to imitate recorded real joint states. This is achieved by defining rewards that combine mimic and regularization terms, guiding the agent to produce appropriate commands while respecting hardware limits.

III. PRELIMINARY RESULTS

A. Residual Dynamic Learning

To evaluate the effectiveness of the residual dynamics model, we tested it on a separate trajectory. As shown in Fig. 3, the simulation with only a PD controller failed to capture the real robot's joint behavior. In contrast, with the learned residual dynamics, the simulated model closely reproduced the real trajectories. The error between simulated and real joint states across all links was small, averaging 0.021 ± 0.12 rad.

TABLE I
REWARDS

Description	Weight	Function
Joint position mimic	5.0	$\varphi(q - q_r)$
Flat velocity mimic	3.0	$\varphi(\dot{q} - \dot{q}_r)$
Joint position bonus	10.0	$1_{\{\,\lvert q-\hat{q}_r\rvert<10^{-3}\,\}}$
Joint torque	-0.5e-5	$\lvert\tau\rvert^2$
Joint velocity	-0.5e-3	$\lvert\dot{q}\rvert^2$
Joint acceleration	-0.5e-6	$\lvert\ddot{q}\rvert^2$
Action rate	-0.5e-4	$\lvert a_t - a_{t-1}\rvert^2$
Action acceleration	-0.5e-6	$\lvert a_t - 2a_{t-1} + a_{t-2}\rvert^2$

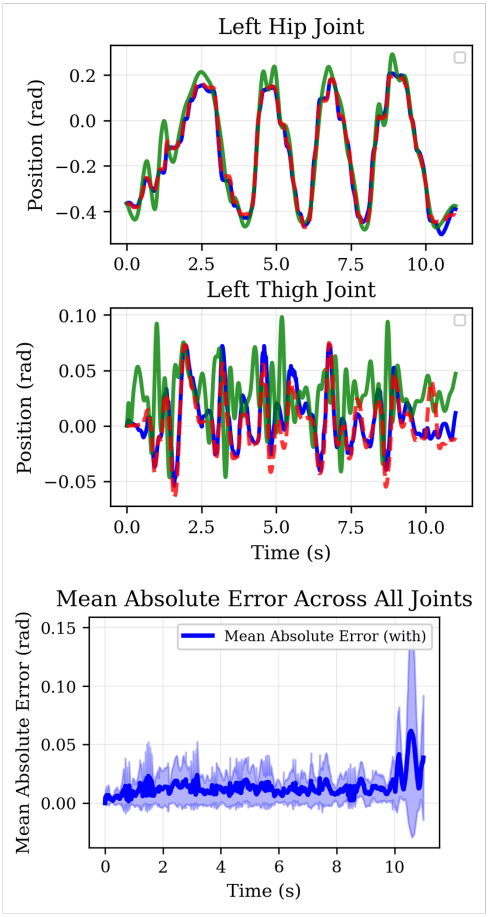


Fig. 3. Comparison of real and simulated joint states with/without residual model (red: real, green: without, blue: with).

B. Effect of Residual Dynamic Model Architecture

In the next experiment, we analyzed the necessity of incorporating a memory mechanism into the residual dynamics model. As hypothesized in the previous section, actuator or sensor delays could motivate the use of GRU+MLPs. The results (Fig. 4) show that both architectures perform similarly: the mean reward was 3.09 for MLPs and 2.854 for GRU+MLPs. However, GRU+MLPs converged more slowly

than plain MLPs. This suggests that delays in our system are negligible, and memory mechanism is not critical.

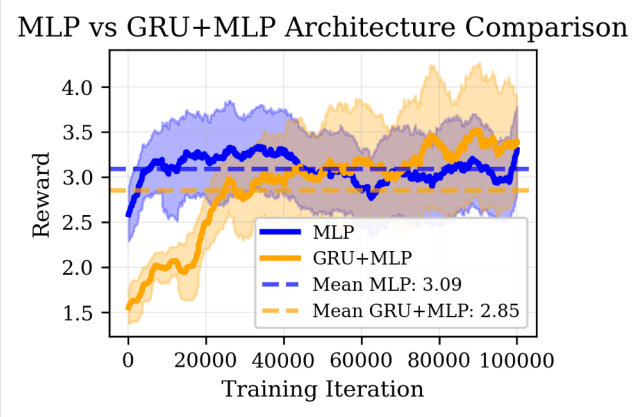


Fig. 4. Joint mimic reward terms over iterations for two network architectures.

IV. CONCLUSIONS

We present a reinforcement learning framework for residual dynamics learning that bridges the sim-to-real gap without costly sensing infrastructure. By leveraging only joint state trajectories, the method aligns simulation with reality and enables zero-shot humanoid skill deployment. Our approach complements existing methods like ASAP and actuator-net modeling, offering a lightweight, accessible alternative for sim-to-real research in humanoid robotics. However, there are limitations to be considered:

- Alignment is trajectory-dependent; unseen skills may still face gaps.
- Residuals learned from limited behaviors may not generalize across tasks.

Future work will extend to online adaptation, where residual dynamics are updated continuously during deployment.

REFERENCES

[1] He et al., "ASAP: Aligning Simulation and Real-World Physics for Learning Agile Humanoid Whole-Body Skills," 2025.
[2] Hwangbo et al., "Learning Agile and Dynamic Motor Skills for Legged Robots," Science Robotics, 2019.
[3] Martin et al., "Reinforcement Learning in System Identification," NeurIPS 2022.