

Impedance Matching: Enabling an RL-Based Running Jump in a Quadruped Robot

Neil Guan¹, Shangqun Yu¹, Shifan Zhu¹, and Donghyun Kim¹

Abstract—Replicating the remarkable athleticism seen in animals has long been a challenge in robotics control. Although Reinforcement Learning (RL) has demonstrated significant progress in dynamic legged locomotion control, the substantial sim-to-real gap often hinders the real-world demonstration of truly dynamic movements. We propose a new framework to mitigate this gap through frequency-domain analysis-based *impedance matching* between simulated and real robots. Our framework offers a structured guideline for parameter selection and the range for dynamics randomization in simulation, thus facilitating a safe sim-to-real transfer. The learned policy using our framework enabled jumps across distances of 55 cm and heights of 38 cm. The results are, to the best of our knowledge, one of the highest and longest running jumps demonstrated by an RL-based control policy in a real quadruped robot. Note that the achieved jumping height is approximately 85% of that obtained from a state-of-the-art trajectory optimization method, which can be seen as the physical limit for the given robot hardware. In addition, our control policy accomplished stable walking at speeds up to 2 m/s in the forward and backward directions, and 1 m/s in the sideway direction.

I. INTRODUCTION

Unlike wheeled systems, legged robots can navigate irregular terrains by leveraging their multiple limbs. This ability allows them to overcome substantial obstacles and wide gaps by running and leaping across diverse terrains. Numerous studies have sought to enhance the dynamic locomotion of legged robots, with notable demonstrations including high-speed running and hurdling [1], and performing backflips [2]. Unofficially, Boston Dynamics’ Atlas robot has shown impressive acrobatic movements across various terrains [3]. Despite these advancements, a common limitation is that the controllers and planners are painstakingly calibrated for specific motions or gaits. Furthermore, model-based methods necessitate model simplification [4], limiting their ability to exploit the full dynamics of the robot.

Reinforcement Learning (RL) emerged as a promising approach for dynamic motion control in legged robots. By effectively exploring the state and action spaces using simulations that encompass full-body dynamics, RL addresses issues tied to limited versatility and model simplification. Advances in RL have led to the development of robust policies capable of navigating challenging terrains in the real world [5]–[8]. Furthermore, a burgeoning interest in honing more advanced skills in bipedal and quadrupedal robots [9]–[13] is opening a new era of RL-based control in legged robotics.

However, there is one fundamental challenge with RL-based dynamic motion control, the so-called sim-to-real issue, and it hampers the deployment of simulation-trained policies in the real world. To address this problem, notable efforts have been made, including dynamics randomization [5]–[8], [10], [14]–[19], meta learning [20], [21], and domain adaptation [11], [19]. Among these, domain randomization is a frequently used method, but it often lacks a formal guideline for selecting both the randomization variables and the range, even though poorly implemented domain randomization can impair policy effectiveness [18].

Indeed, we can substantially reduce the gap between simulation and reality by synchronizing the robot’s output in both realms, given identical command inputs. For instance, if the errors in joint position from the desired joint positions are consistent between the simulated and real robot, the discrepancy in overall locomotion behaviors should be minimal. Time domain system identification was explored by [13] to align the actuation parameters between simulation and reality, albeit briefly. Other research focuses on modeling actuators [15] or learning actuator dynamics [16], but they overlook physics engine implementation details and neglect modeling outside the actuators, such as belt and link contact friction, which can cause an increase in error magnitude when the commanded motion is rapid.

This insight forms the foundation of our method: sim-to-real synchronization based on frequency-domain analysis. Instead of naively copying the feedback gains used in the real robot, we conduct a frequency sweep in both real and simulated robots and select appropriate actuator parameters for simulation by matching their bode plots. We refer to our framework as ‘impedance matching.’ While we recognize that our use of *impedance* – referring here to joint stiffness and damping – diverges from its technical definition in [22], we believe it effectively encapsulates our method’s essence. Our framework offers two critical advantages: 1) It allows us to set up a simulation environment that mirrors a robot performing highly dynamic movements, as the frequency sweep can easily cover beyond 20 Hz, wide enough to encompass running and jumping, and 2) it enables us to reduce the randomization range based on experimental data, as we can approximate the true value and variance from multiple, real-world, hardware frequency analysis experiments.

In our RL training, we implemented function preserving transformations, typically used in supervised learning [23], to maintain normal walking behavior while learning to jump in a curriculum learning setup [14], [16], [17]. Our method centers on developing a jumping behavior derived from

¹ University of Massachusetts Amherst, 140 Governors Dr, U.S. donghyunkim@cs.umass.edu

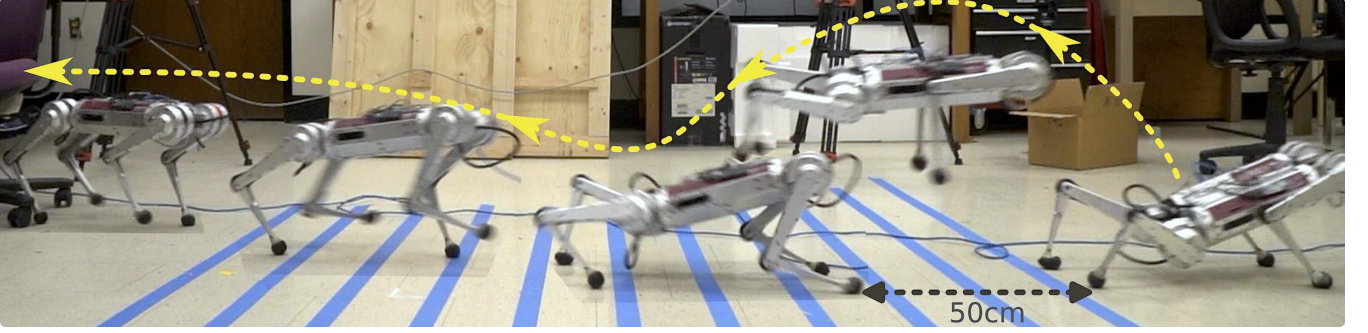


Fig. 1. **Run and jump demonstration of the Mini-Cheetah robot using our trained policy.** Our novel learning framework enables the quadruped robot to execute dynamic behaviors, such as running and jumping, while significantly narrowing the simulation-to-reality gap. The maximum jumping length and height achieved by our policy are comparable to those obtained through trajectory optimization and the state-of-the-art RL-based policies. This suggests that our results are approaching the maximum capability of the given robotic hardware.

an earlier-trained walking behavior. Unlike some research exploring simultaneous learning of multiple behaviors [24], [25] or training an individual policy for each behavior [13], [26], we showed that a jumping behavior can be learned with only minor adjustments to a walking policy.

Our key contributions include: 1) The first introduction of frequency-domain analysis in RL, termed ‘impedance matching’, to reduce the sim-to-real gap; 2) A versatile control policy enabling smooth transitions between dynamic movements such as walking, running, and jumping; 3) The successful real-world deployment of the policy on Mini-Cheetah-Vision, evaluated extensively in dynamic locomotion involving high and long jumping tasks.

II. RELATED WORKS

While there is an abundance of research enabling impressive acrobatic behaviors of robots in simulations [27]–[29], actual demonstrations of jumping with real quadruped robots are relatively rare. Notably, [1] showcased an impressive running jump; however, it relied on a planar model for real-time trajectory optimization, limiting its applicability for 3D omnidirectional running jumps. Works such as [30] successfully pushed the Mini-Cheetah robot’s physical capabilities by combining centroidal dynamics with variational-based optimal control. Yet, the method’s offline optimization requirement constrains its capacity for real-time, spontaneous behavior.

[31] used Depth-based Impulse Control to facilitate the RL-based running jumps over gaps, but at the cost of omnidirectional capabilities. Similarly, [12] managed to train generalist RL policies across multiple environments, but these policies lacked essential motor skills like walking backward, relying exclusively on forward movement and turning for navigation. Several other studies employed an imitation learning component in their RL approaches, where the policy learns by mimicking a reference motion [10], [11], [32]. However, the need for reference trajectories limits the scalability of the methods to address various situations including unseen environmental hazards. In contrast, our approach leverages a unique reward design in lieu of imitation learning. This strategy enables us to apply a simulation-

trained policy directly in the real world without the need for additional motion retargeting [11], [32] or manual creation of reference trajectories [10].

Recently, [13] proposed using policy distillation to combine walking, slope climbing, and jumping policies into a single generalist neural network, but they did not demonstrate on-the-fly policy transitions. Our implementation, however, can easily transition between different behaviors, thereby allowing our robot to modulate its speed freely before executing a jump.

III. SIM-TO-REAL SYNCHRONIZATION BASED ON FREQUENCY-DOMAIN ANALYSIS

To streamline our analysis, we formulate two assumptions. The first one is that the joint dynamics can be closely approximated by a second-order spring-damper system, represented by

$$I\ddot{\theta} + b\dot{\theta} = \tau \left(= K_p(\theta_{\text{des}} - \theta) + K_d(\dot{\theta}_{\text{des}} - \dot{\theta}) \right), \quad (1)$$

$$I\ddot{\theta} + (K_d + b)\dot{\theta} + K_p\theta = K_p\theta_{\text{des}} + K_d\dot{\theta}_{\text{des}}, \quad (2)$$

where I and b refer to the actuator’s inertia and viscous friction, respectively, while K_p and K_d denote the proportional (P) and derivative (D) feedback gains. The second assumption is that the inertia between the actual and simulated robot should be roughly equivalent. To maintain the veracity of this assumption, we disassembled the robot and measured the mass and center of mass (CoM) for each link element by element, and roughly determined our inertia values via CAD models. We also assume that non-viscous and state-dependent friction terms such as static/kinetic/Coulomb friction and load/temperature effects are negligible for high-speed locomotion [33].

Fig. 2 (a) illustrates our experimental setup. We conduct frequency response experiments on each joint individually using a chirp signal, sweeping the joint position command from 0.1 Hz to 25 Hz with an amplitude of 0.25 rad. The frequency range was determined based on the Nyquist frequency, which corresponds to the simulation data’s sampling frequency of 50 Hz, set by our RL policy update frequency. The chosen amplitude is large enough to

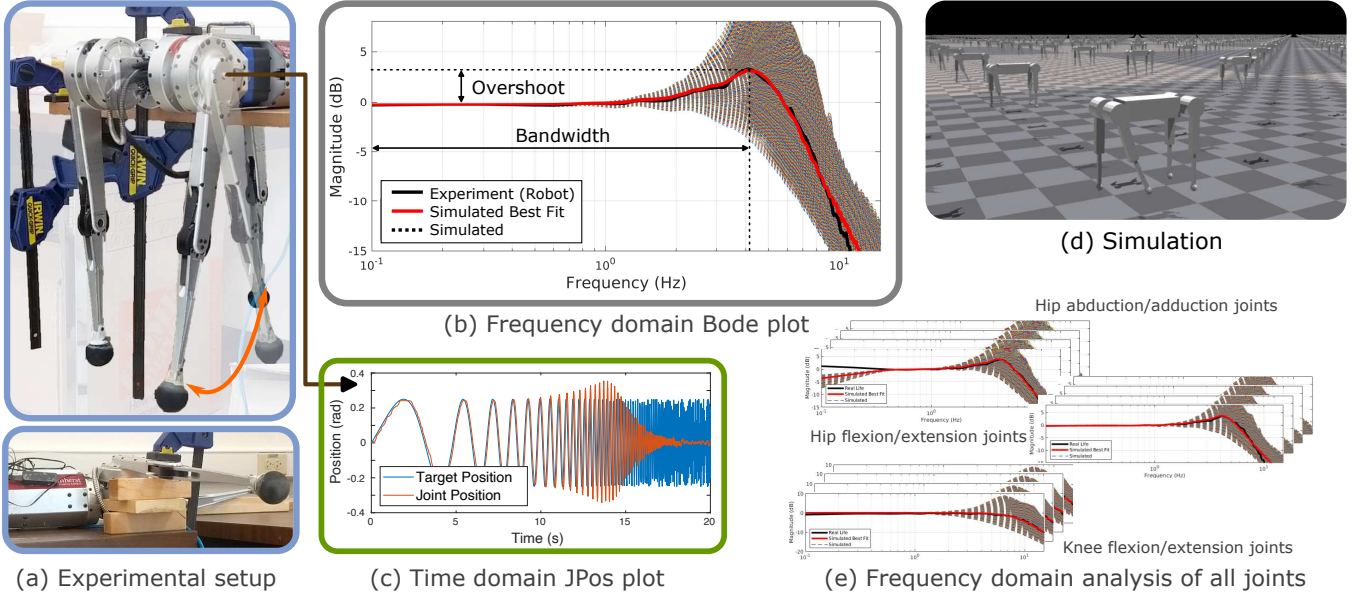


Fig. 2. Chirp signal test and frequency response analysis of a real and simulated robot

yield significant measurements given the motor encoder's resolution while remaining small enough to avoid physical damage from resonance or exceeding the motor's maximum torque. Before performing a sine sweep, we secured the robot's body to a stable platform, employing Proportional-Derivative (PD) feedback control to hold all joint positions at static posture except the joint under examination. When testing knee joints, we secured the thigh link, as depicted in Fig. 2(a), to avoid the coupled behavior of hip and knee flexion/extension joints. This frequency sweep procedure was reiterated for all joints. We executed the parallel frequency sweep experiment in simulation, as depicted in Fig. 2(c), testing various feedback gains for each actuator across a 50×50 grid. The values for the proportional gain, K_p , ranged from $13 - 27 \text{ N m/rad}$, while the derivative gains spanned a range of $0.1 - 0.7 \text{ N m s/rad}$.

We employed MATLAB to estimate the transfer function and convert the measured and commanded joint positions in the simulation and real robots into Bode plots. The phase plot was disregarded for two reasons: 1) the time delay, a significant determinant of the phase plot, behaves differently in simulations than in physical systems and measuring exact delay is also difficult, and 2) the remaining information in the data (bandwidth and overshoot) presented in the magnitude plot provide sufficient information to identify the system characteristics of interest. We identified the appropriate gains for use in the simulation by finding the gain pair that yielded the smallest mean-squared error between the Bode plots of the real and simulated systems over the frequencies near the natural frequency. This error computation in proximity to peak magnitude helps identify matches with similar bandwidth and overshoot. Thus, we employed data ranging from $0.1 - 15 \text{ Hz}$ for the knee joints and $1 - 10 \text{ Hz}$ for the hip abduction and flexion joints to find the best match.

Our choice of feedback gains ($17/0.4$ for K_p/K_d) was

	$K_p(\text{N m/rad})$	$K_d(\text{N m s/rad})$
Hip roll	20.6, 18.1, 18.5, 21.0	0.492, 0.516, 0.431, 0.49
Hip pitch	16.5, 17.7, 16.9, 18.9	0.382, 0.406, 0.382, 0.431
Knee pitch	21.8, 22.0, 22.2, 21.8	0.541, 0.523, 0.553, 0.553

TABLE I
EMPIRICALLY DETERMINED GAINS (FR, FL, HR, HL)

a reimplement of [34]. In theory, the selection can be arbitrary to an extent, as changing the feedback gains will only change the joint target position output from the policy for any given torque output, which is learned via reinforcement learning. We found success in choosing the same gains to get the policy to converge, albeit perhaps choosing a set of critically damped gains would be more stable.

Upon repeating the search across all twelve joints, we successfully established experimental gain pairs, as delineated in Table I. We noted a substantial disparity between the feedback gains used in the robot hardware ($17/0.4$ for K_p/K_d) and those found in the simulated robots that exhibit similar impedance behavior to the physical system. Domain randomization can address changes in a robot's physical properties due to manufacturing tolerances, calibration or wear-and-tear [19], so we used the empirically determined gain variances in our experiments to roughly obtain the smallest domain randomization ranges encompassing our experimental data. As per Table I, we set the feedback gains to be used during simulation training: $20, 17.5, 21.5 \text{ N m/rad}$ for K_p , and $0.45, 0.4, 0.55 \text{ N m s/rad}$ for K_d , corresponding to the hip abduction/adduction, hip flexion/extension, and knee flexion/extension joints, respectively. We executed domain randomization in simulated robots by adding values sampled from $\mathcal{U}^{12}(-2.0, 2.0)$ and $\mathcal{U}^{12}(-0.05, 0.05) \text{ N m s/rad}$.

We demonstrate through impedance matching that high-

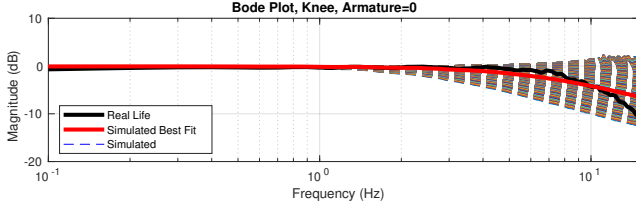


Fig. 3. Knee Bode plot assuming zero rotor inertia.

frequency sim-to-real transfer with robots with geared actuators is very difficult without addressing rotor inertia, which is often not mentioned in various RL locomotion papers. Isaac Gym enables rotor inertia to be approximately simulated via its rigid body armature parameter, which is a constant array added to the diagonals of the mass matrix of the robot, a critical element in fitting Bode plots from simulation to empirical measurements. Assuming rotor inertia is zero, the best-fit simulation parameters do not match real life dynamics, as shown in Fig. 3, as our grid-search across a wide range of gains yields no match to real life. The rotor inertia markedly influences the best fitted gains; thus, careful selection is crucial. We used $0.000072 \times 9.33^2 (\approx 0.0063)$ kg for the reflected rotor inertia of knee joints and $0.000072 \times 6^2 (\approx 0.0026)$ kg for the other joints. Note that we incorporated detailed actuator models [35] into Isaac Gym, which include aspects such as rotor inertia, the back-emf effect, dry friction, and bus (battery) voltage limits, which were experimentally measured using a custom dynamometer. These additions allow us to accurately simulate actuator behavior, particularly at the edge of its operational conditions.

IV. TRAINING FRAMEWORK TO LEARN WALKING, RUNNING, AND JUMPING

In order to acquire multiple skills for a single RL agent, we adopted the idea of Net2Net [23] for multi-task RL. In particular, we use a modified Net2WiderNet implementation performed on the command inputs of a network. This is done by adding a blank neuron by adding a row of zeros to the input layer weight matrix. This blank neuron can be trained to freely signal the transition into and out of the new behavior, trained with conditional rewards. To signal a jump, the blank neuron begins at 0 to walk, transitioning to 1 in the jumping state, and returning to 0 to land. We train in multiple phases. During Phase 1, we train the agent how to walk. During Phase 2, we add a jump command input neuron to the model’s input layer and train both walking and jumping in a 50/50 split to avoid catastrophic forgetting of walking. Notably, the jump can be controlled using velocity commands just as walking could. To our knowledge, this is the first time Net2Net has been used for locomotion.

Our reward terms take inspiration from [12], [14], [17], [34]. Additional rewards introduced starting in Phase 2 are shown in Table II. Of note is the dense jump reward, which shapes the jumping behavior. This simplistic jump reward can train a jump without requiring a reference motion. The sparse jump is intended to further shape the jump to be large

enough to cross gaps. Phase 2 is split into subphases 2a and 2b. The dense jump reward encourages jumping motions, but should be lowered after the robot is capable of jumping. The sparse jump reward encourages consistent jumps based on the desired liftoff velocity, which we set at 2.5m/s.

In multi-task RL, it is often challenging for a single policy to acquire distinct skills due to the presence of opposite gradient directions and nonstationary data. In our case, we note that training both walking and jumping simultaneously from scratch does not converge to distinct behaviors, whereas if we train them sequentially, we obtain distinct behaviors, leading us to believe that the gradients corresponding to one task oppose the gradients corresponding to the other. Segmenting different behaviors into different states not only grants the freedom to engineer different behaviors individually but also grants interpretability in allowing a higher level policy to choose which behaviors are best.

A. System setup:

We use the MIT Mini Cheetah Vision as our experimental platform. The robot stands 30 cm tall and weighs 12 kg. The robot features 12 custom-designed proprioceptive actuators, each with a maximum output torque of 17 N m. Our model has a slightly longer body length compared to the original MIT Mini Cheetah [2]. Our robot uses a minimal sensing suite, allowing our framework to be generalizable to many robots, including relatively inexpensive ones. We use data from the robot’s accelerometer, gyroscope and joint encoders for the robot’s walking, running and jumping capabilities. We use Isaac Gym [14] to train the quadrupedal robot and adapted the code from [34] for the sim-to-real deployment.

B. Estimator network:

We follow [34] in our estimator network implementation on the mini cheetah. Our estimator network takes in the base angular velocity, the projected gravity, the joint position history, the joint velocity history, the action history and the jumping mode. The joint position histories are sampled at t , $t - 0.02$, and $t - 0.04$, and the action history is sampled at $t - 0.02$ and $t - 0.04$. The estimator network predicts foot contact probability and the estimated linear velocity of the robot and is trained via supervised learning. The estimator network is implemented with a $[256 \times 128]$ structure.

C. Control Policy:

The control policy is trained with PPO [36], with both actor and critic networks having a $[1024 \times 512 \times 256]$ structure. The observation space consists of the estimated velocity, the foot contact probabilities, the base angular velocity, the projected gravity, the x , y , yaw velocity commands, the jump command, the joint position, the joint velocity, and the previous actions. The sensing and inference update loop runs at 50 Hz, while the actuator-level PD controller runs at 40 kHz. The action is a set of target joint position vectors that will be assigned to the PD controller. We sampled x velocity commands from $\mathcal{U}^1(-3.0, 3.0)$ m/s, y velocity commands from $\mathcal{U}^1(-1.0, 1.0)$ m/s, and yaw

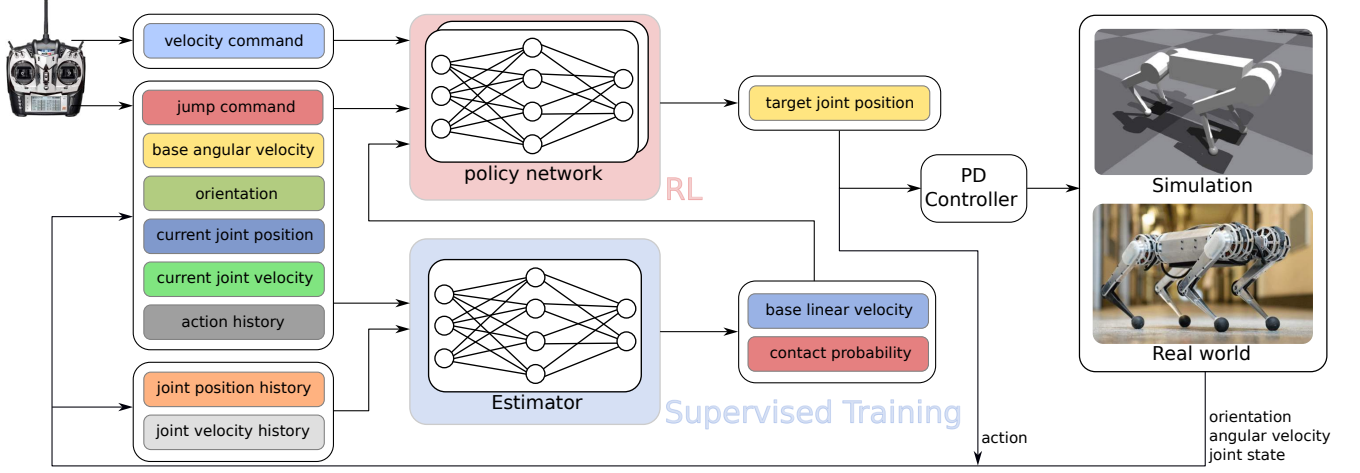


Fig. 4. Main pipeline of the RL agent

TABLE II
JUMP REWARD TERMS

Reward Term	Expression	Scale, (Phase 1)	Scale(Phase 2a)	Scale (Phase 2b)
Dense Jump	$-std(F_{foot})$	0.0	-2.5	-0.25
Sparse Jump	$exp(-(V_{liftoff} - V_{liftoff,desired})^2/2)$	0.0	250	250

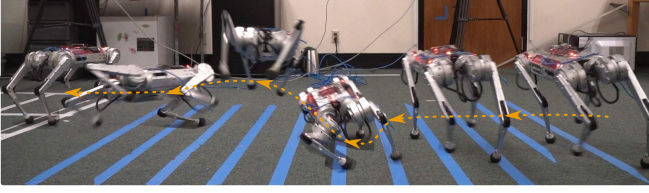


Fig. 5. A sideways running jump using naive policy is not stable.

angular velocity commands from $\mathcal{U}^1(-2.0, 2.0)$ rad/s. Jump commands are randomly sampled between 0 and 1.

D. Domain Randomization:

Domain randomization is employed to prevent overfitting to the simulation dynamics. Decimation is randomized to $\mathcal{U}^1(-2, 2)$ s to mitigate the effect of variable system delay on the real robot. Ground friction is randomized to $\mathcal{U}^1(0.05, 3.0)$ N m/rad. We randomize stiffness and damping by adding a value sampled from $\mathcal{U}^{12}(-2.0, 2.0)$ and $\mathcal{U}^{12}(-0.05, 0.05)$ N m s/rad to our commanded stiffness and damping, respectively, based off our impedance analysis. Shank length is randomized from $\mathcal{U}^4(0.18, 0.20)$ N m to help mitigate the effects of foot deformation. Base mass is randomized from $\mathcal{U}^1(-0.4, 1.6)$ kg. Joint position noise is sampled from $\mathcal{U}^{12}(-0.05, 0.05)$ rad to account for poor encoder measurements and deviations in the zero position of the motor. Joint velocity noise is sampled from $\mathcal{U}^{12}(-0.5, 0.5)$ rad/s. Angular velocity noise is sampled from $\mathcal{U}^3(-0.2, 0.2)$ rad/s. Projected gravity noise is sampled from $\mathcal{U}^3(-0.05, 0.05)$

V. EXPERIMENTAL RESULTS

To prepare a consistent way to test each jump, we start the robot from a standstill and write a short script to accelerate to a specified velocity, jump for a specified time interval, and decelerate to a stop for a total of five jumps for each velocity. We perform this experiment with two trials, one trained with our experimentally determined gains, and another trained with the naive gains. Table III lists the mean and standard deviation jump height and distance of the robot at various speeds for both trials.

The data is obtained via a motion capture rig. We measure the difference between the minimum and maximum height of the tracker when the robot jumps to approximate the jump height, and we find the distance by taking the difference between the robot's X-Y position during the inflection points of the jump. Although the naive method sometimes matches or even exceeds the distance with the matched impedance method, the average jump distance standard deviation is 230% that of the matched impedance policy. Similarly, the jump height average standard deviation is 148% that of the matched impedance policy, further indicating inconsistent behavior in the naively trained policy. Fig 5 illustrates an unstable but long sideways jump from the naive policy, which in our experience, was much more present in the naive policy. We attribute the inconsistency and instability to the poorer sim-to-real transfer in the naive method. A stable sim-to-real transfer is a consistent sim-to-real transfer, and the impedance matched trials' lower variance in jump height and distance indicate a better sim-to-real transfer.

[37] provides a highly optimized upper bound on the jump height of the Mini-Cheetah, achieving jumps onto 34 cm platforms, roughly corresponding to approximately 45 cm

Command	Jump Height (m)				Jump Distance (m)			
	Naive		Matched		Naive		Matched	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Forwards, 1.0 m/s	0.332	0.008	0.318	0.014	0.690	0.076	0.597	0.042
Forwards, 2.0 m/s	0.246	0.022	0.244	0.013	0.944	0.030	0.969	0.026
Sideways, -1.0 m/s	0.304	0.061	0.374	0.016	0.636	0.092	0.587	0.054
Sideways, 1.0 m/s	0.249	0.041	0.366	0.052	0.701	0.183	0.542	0.051
Backwards, -2.0 m/s	0.346	0.031	0.381	0.015	0.944	0.030	0.932	0.025

TABLE III
REAL-LIFE JUMP HEIGHT AND DISTANCE

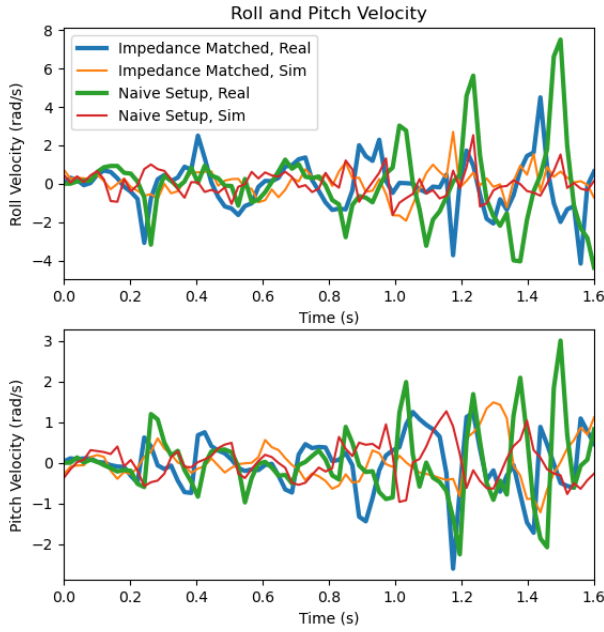


Fig. 6. Angular velocities (roll, pitch) of a robot under the 2.25 m/s forward running command.

jump heights. Treating their work as the robot’s physical limit, we were able to achieve 85% of this height consistently while performing backward running jumps.

We place tape markers on the ground every 20 cm denoting the robot’s distance jumped and go frame-by-frame on the video to determine where the robot entered and exited its airborne phase to roughly estimate the maximum gap each jump can cover. In all trials, the maximum gap the robot can cross is 55 cm going forward at 2 m/s, 50 cm going backward at 2 m/s, and 40 cm going sideways.

We record the roll and pitch velocity readings during the acceleration phase for both the naive and impedance matched policy, as shown in Fig 6.

VI. LIMITATIONS

Our method assumes good inertia estimations, which were determined using CAD models and are difficult to verify. The same actuators performing similar roles have vastly different experimentally determined gains. Although factors such as the simulation implementation, belt friction and link contact friction may account for some of these differences, it is unclear how much these differences are from inaccurate

inertia estimations, and as such, applied loads on the robot such as ground forces cause unmodeled sim-to-real variables. Alternative approaches [15] can be to assume uniform mass distribution for each link, or perform domain randomization on inertia estimations, but even in those contexts, they still depend on an initial inertia estimation, which are often not verified experimentally. Future work should use calibrated mass property measurement systems for empirical inertia verification.

We trained running and jumping with random state transitions leads policies, but they learned conservative behaviors that are always ready to transition between each other. For instance, at higher forward speeds, the walking policy turns into a bounding policy, which can easily transition into jumps, and additional reward turning is necessary to prevent the walking policy from changing. Last, pushing the limits of the hardware is not enough to clear meaningful gaps with modern actuators; another necessity is for the robot to step precisely around gaps, which is not addressed by our method.

VII. CONCLUSION

We develop a new learning framework to enable highly dynamic maneuvering of a quadrupedal robot including stable walking, rapid running, and high/long jumping via RL. RL is a promising approach for training such controllers, but deploying the learned control policy in real life is notoriously difficult due to the gap between simulation and real life. We propose a method to minimize the sim-to-real gap through frequency-domain analysis with little physical risk to the robot. We provide a jumping controller enabling running omnidirectional jumps, trained with a novel, model-agnostic method. We show that our method enables jumps of a distance of 96 cm for the robot’s center of mass which corresponds to a 55 cm gap and heights of 38 cm, which is competitive with state-of-the-art performance in quadrupedal robot jumping in real-world tests in distance and height.

ACKNOWLEDGMENT

We express our gratitude to Naver Labs and MIT Biomimetic Robotics Lab for providing the Mini-cheetah robot as a research platform for conducting dynamic motion studies on legged robots.

REFERENCES

- [1] H.-W. Park, P. M. Wensing, and S. Kim, “High-speed bounding with the mit cheetah 2: Control design and experiments,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017.

- [2] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6295–6301.
- [3] B. Dynamics, "Atlas, partners in parkour," <https://youtu.be/tF4DML7FIWk>, 2021, accessed on 2023/06/06.
- [4] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–9.
- [5] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Robotics: Science and Systems*, 2022.
- [6] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.
- [7] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 403–415. [Online]. Available: <https://proceedings.mlr.press/v205/agarwal23a.html>
- [8] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, jan 2022. [Online]. Available: <https://doi.org/10.1126/2Fscirobotics.abk2822>
- [9] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022. [Online]. Available: <https://doi.org/10.1145/3528223.3530110>
- [10] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through reinforcement learning," 2023.
- [11] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 07 2020.
- [12] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.
- [13] K. Caluwaerts, A. Iscen, J. C. Kew, W. Yu, T. Zhang, D. Freeman, K.-H. Lee, L. Lee, S. Saliceti, V. Zhuang, N. Batchelor, S. Bohez, F. Casarini, J. E. Chen, O. Cortes, E. Coumans, A. Dostmohamed, G. Dulac-Arnold, A. Escontrela, E. Frey, R. Hafner, D. Jain, B. Jyenis, Y. Kuang, E. Lee, L. Luu, O. Nachum, K. Oslund, J. Powell, D. Reyes, F. Romano, F. Sadeghi, R. Sloat, B. Tabanpour, D. Zheng, M. Neunert, R. Hadsell, N. Heess, F. Nori, J. Seto, C. Parada, V. Sindhwani, V. Vanhoucke, and J. Tan, "Barkour: Benchmarking animal-level agility with quadruped robots," 2023.
- [14] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," 2022.
- [15] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.
- [16] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019. [Online]. Available: <https://www.science.org/doi/abs/10.1126/scirobotics.aau5872>
- [17] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021.
- [18] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4955–4961.
- [19] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 737–744.
- [20] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2950–2957, 2020.
- [21] X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, and J. Tan, "Rapidly adaptable legged robots via evolutionary meta-learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3769–3776.
- [22] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American control conference*. IEEE, 1984, pp. 304–313.
- [23] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," 2016.
- [24] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," 2018.
- [25] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," 2022.
- [26] Y. Chen, T. Wu, S. Wang, X. Feng, J. Jiang, S. M. McAleer, Y. Geng, H. Dong, Z. Lu, S.-C. Zhu, and Y. Yang, "Towards human-level bimanual dexterous manipulation with reinforcement learning," 2022.
- [27] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 143:1–143:14, July 2018. [Online]. Available: <http://doi.acm.org/10.1145/3197517.3201311>
- [28] S. Hong, D. Han, K. Cho, J. S. Shin, and J. Noh, "Physics-based full-body soccer motion control for dribbling and shooting," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 1–12, 2019.
- [29] T. Kwon, Y. Lee, and M. V. D. Panne, "Fast and flexible multilegged locomotion using learned centroidal dynamics," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 46:1–46:17, 2020.
- [30] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7693–7699.
- [31] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. bae Kim, and P. Agrawal, "Learning to jump from pixels," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: <https://openreview.net/forum?id=R4E8wTUtdxl>
- [32] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Learning and adapting agile locomotion skills by transferring experience," *preprint arXiv:2304.09834*, 2023.
- [33] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.
- [34] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.
- [35] B. G. Katz, "A low cost modular actuator for dynamic robots," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.
- [36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [37] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," 2021.