

Watch Your Step: Anticipated Perception for Humanoid Locomotion

Matthias Heyrman^{1,*}, Chenhao Li^{1,2}, Ioannis Dadiotis¹, Marco Hutter¹

Abstract—Perceptive locomotion policies typically observe a dense robot-centric elevation map, although only the terrain beneath and around the next footsteps informs the control decision. Learned encoders compress the whole visual input, leaving the policy to work out which inputs matter while learning the baseline locomotion task. Learning both at once slows training and, without computationally expensive attention mechanisms, makes generalization to new terrain harder. We instead remove redundant inputs from the elevation map before the policy sees it. A lightweight foothold prediction model (FPM) predicts future footholds from proprioception and the velocity command alone, and we sample the elevation map only in small patches around these predicted footholds. We train the prediction model alongside the locomotion policy, periodically retraining it on new policy rollouts as the robot’s gait matures. This allows the perception to anticipate where the policy will step, while the policy learns to leverage terrain information around those steps. With the proposed lightweight preprocessing, the policy reaches a given terrain difficulty in roughly a third of the training iterations required with the complete map. It outperforms policies that observe the complete map where footholds are sparse, including in narrow walled corridors, and on unseen terrain, and it matches an attention-based encoder trained at several times the cost. The policy transfers zero-shot to a Unitree G1 humanoid traversing various obstacles, stairs, gaps, and step-downs beyond the training distribution in both indoor and outdoor environments.

I. INTRODUCTION

Vision is an indispensable tool when traversing unstructured terrain. People constantly survey the ground ahead of them to plan their next steps and to adjust the trajectory of a foot in mid-air. Deep reinforcement learning (RL) has driven significant progress in legged robot locomotion [1], [2]. Using high-fidelity physics simulators, modern RL algorithms with parallelized environments, and effective techniques to reduce the sim-to-real gap, humanoids can be conditioned on a robot-centric elevation map [3] to traverse rough terrain.

However, this reliance on dense environmental maps introduces fundamental limitations related to sample efficiency, redundancy, and distraction. Only the small regions of terrain where the feet will next make contact should inform control decisions, yet policies typically sample and process the full map uniformly, even though most of it describes terrain the robot will never step on, such as cells far to the side or directly opposite the commanded velocity direction. These irrelevant inputs make it hard to learn which regions of

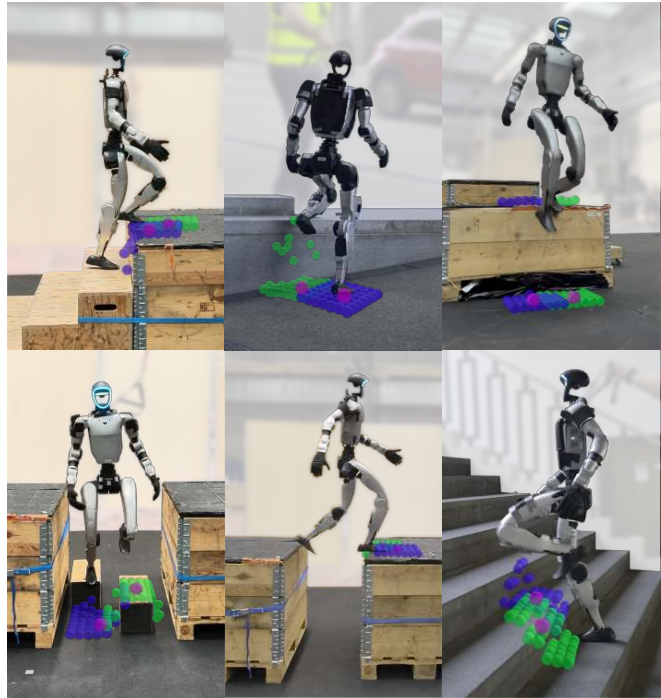


Fig. 1. Zero-shot multi-terrain locomotion, including stairs, drops, gaps, unstable terrain, and crowded elevation-mapped environments using a Foothold Prediction Model (FPM) filtered elevation map. Predicted future footholds are shown in magenta, and the sampled elevation map points are shown in green and blue.

the map matter and can distract the policy with terrain features that have no bearing on the next step, while the larger state space makes generalization to different terrains harder and increases the computational cost of the policy. By contrast, people tend to look at the ground approximately 1.5 seconds, i.e., 2 to 3 steps, ahead of them, focusing on their planned footholds instead of the entire surrounding scene [4]. On more challenging terrains, this look-ahead lets them intuitively adapt a step in mid-air: if a foothold seems insecure or they notice a better spot near the original target, they shift their focus and change the step trajectory.

Existing work primarily aims to reduce redundant information from the full elevation map through learned embeddings that feed the control policy. Typical encoders compress the full input into a latent vector [2], scale well to complex terrains [5], [6], and transfer to real-world locomotion [7], [8]. However, encoders such as CNNs still have to learn to extract the relevant features from the full map, and thus inherit the redundancy and sample inefficiency of the visual input rather than addressing it.

Attention-based encoders, conditioned on proprioception

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

¹Robotic Systems Laboratory, ETH Zürich, Switzerland.

²ETH AI Center, ETH Zürich, Switzerland.

*Corresponding author: mheyрман@ethz.ch

Project page: leggedrobotics.github.io/watch-your-step

and the elevation map, weigh terrain features to focus on possible future steps [9], and later work extends them with local and global feature streams [10], [11] and uncertainty-aware terrain estimates [10]. Using egocentric depth images alongside a height scan enables an attention-based anticipatory gaze that better respects onboard compute limitations [12]. These methods learn where to look, but nothing in the encoder structure ensures that the attended features are future footholds; the encoder has to discover the focus implicitly from the task reward. Other depth-based methods encode the images directly [13], [14]. All of these methods process the entire visual input when deployed and leave it to the policy to ignore what does not matter. Terrain outside of its next steps can therefore still distract it.

Another approach analyzes the terrain and plans where the robot’s feet should land, a long-standing body of work. Classical methods select candidate contact regions based on kinematic reachability and terrain geometry [15]–[18], and learning-based approaches evaluate candidate footholds based on the height map around the nominal landing position [19] or across the local map under kinematic and collision constraints [20]. These foothold selection methods run as standalone modules that hand a target to a downstream tracking controller. This requires strongly defined hand-crafted selection criteria for each platform, such as thresholds on the slope, roughness and curvature of a foot-sized patch, its distance to edges, and robot-specific checks of kinematic reachability and leg collision. These criteria encode the designer’s notion of a safe foothold and must be manually retuned for a new foot geometry or gait. The module is also not necessarily conditioned on how the downstream controller actually moves, which leaves its output liable to be inaccurate or suboptimal for the gait that executes it.

The mismatch between planner and controller is especially pertinent for modern legged locomotion controllers, which are RL policies trained in simulation to maximize a reward. RL-based approaches can learn precise foot placement in simulation through a terrain difficulty curriculum and crafted rewards [21], [22] and can learn to track foothold planners [23], [24], use them to shape rewards [25], or track a decoupled learned planning policy [26]. In these cases, however, the foothold serves as a tracking target or a reward that shapes the policy’s actions, not as a filter that shapes the policy’s observations. Prior work has used predicted footholds to reward safe locomotion using privileged information acquired in simulation during training [14], but the predictions never reach the hardware.

Filtering the elevation map around future footholds before the policy sees it raises two challenges. First, the filter cannot observe the terrain to decide where to look, as it would inherit the overhead of the full elevation map that it is meant to remove. It must therefore predict the footholds entirely from the robot’s own state and command, while the policy, like a person, can still adjust its trajectory to choose footholds based on the terrain it sees around each prediction. Second, where the feet will land is a property of the gait, which the policy is still learning. A predictor fit to an early, unsteady

gait would misplace its patches once the policy improves, and a policy trained on misplaced patches may learn to distrust them.

In this paper, we propose a human-inspired, state-based sampling module that filters the elevation map before the locomotion policy receives any terrain information. To filter the elevation map without observing it, a lightweight *foothold prediction model* (FPM) predicts the robot’s next two footholds from its current proprioceptive state and velocity command alone. We treat only the terrain around these predicted footholds as relevant for locomotion and sample the elevation map in patches surrounding them. These patches act as the exteroceptive observation of the locomotion policy, creating a state-conditioned representation of the terrain.

To ensure that the predicted footholds, and with them the sampled patches, stay up-to-date as the locomotion policy learns more difficult terrains, we co-train the FPM and the locomotion policy in an iterative training loop. We train the locomotion policy with RL in simulation and, after a fixed number of iterations, refit the FPM in a self-supervised fashion on the footholds the current policy produces, allowing the perceptual filter to adapt as the policy converges. We demonstrate that this approach reaches a given terrain difficulty in roughly a third of the iterations required by other methods using the complete elevation map, at a lower perception cost to the policy and across a variety of rough and sparse terrain styles. We also show indoor and outdoor hardware experiments, where the resulting policy successfully transfers to a Unitree G1 humanoid (Fig. 1).

In summary, our contributions are as follows:

- 1) A self-supervised foothold prediction model (FPM) that accurately predicts the next two footholds from the current proprioceptive state and velocity command, co-trained on policy rollouts during RL training.
- 2) A human-inspired perceptive locomotion pipeline that samples the elevation map only around the footholds predicted by the FPM, with insights into its key design choices, such as the prediction horizon and the encoding of the sampled patches.
- 3) Simulation and hardware experiments, including comparisons against state-of-the-art elevation-map encoders, that demonstrate the advantages of restricting the policy’s exteroceptive input to the terrain around its next predicted footholds over the complete map.

II. PRELIMINARIES

We formulate the locomotion problem as an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where $\mathcal{S}$ denotes the robot’s state space, $\mathcal{A}$ the robot’s possible action space, $P(s_{t+1}|s_t, a_t)$ represents the environment transition dynamics, $R(s_t, a_t)$ is the reward function, and γ the discount factor. We train the policy with PPO [27] across 8192 parallel environments in IsaacLab [28] on a single NVIDIA RTX 3090 GPU, controlling a 29-DoF Unitree G1 humanoid. Physics is integrated at 200 Hz and the policy control rate is 50 Hz.

We define the system state $\mathbf{x}_t$ as the robot’s estimated

Algorithm 1 Policy and FPM co-training

```

1: Initialize locomotion policy  $\pi_\phi$  and FPM  $f_\theta$ 
2: Pin  $\hat{\mathbf{p}}_t^1, \hat{\mathbf{p}}_t^2$  to  $(0, \pm 0.1)$  m beneath the base for bootstrapping
3: Train  $\pi_\phi$  for  $n_0 = 500$  iterations
4: while training do
5:    $K \leftarrow \lceil N_s / N_{\text{env}} \rceil$   $\triangleright N_s = 5 \times 10^5$  samples
6:   for  $K$  steps do
7:     Record  $\{\mathbf{x}_t, c_t^L, c_t^R, \mathbf{r}_t\}$  using policy  $\pi_\phi$ 
8:   end for
9:   Label strictly future footholds  $\mathbf{p}_t^1, \mathbf{p}_t^2$  to form  $\mathcal{D}$ 
10:  Mask windows that span a termination (base-pose jump  $> 0.5$  m)
11:  Retrain  $\theta$  from scratch on  $\mathcal{D}$ , 25 epochs, 80/20 split
12:  Export  $f_\theta$ ; hot-swap into the observation pipeline
13:  for  $n_r = 500$  policy iterations do
14:    Predict footholds  $\hat{\mathbf{p}}_t = f_\theta(\mathbf{x}_t)$ 
15:    Sample  $k \times k$  height patches at  $\hat{\mathbf{p}}_t^1, \hat{\mathbf{p}}_t^2$ 
16:    Update  $\pi_\phi$  with PPO
17:    Advance terrain and assistive-force curricula
18:  end for
19: end while

```

proprioceptive state:

$$\mathbf{x}_t = [\mathbf{v}_t, \boldsymbol{\omega}_t, \mathbf{g}_t, \mathbf{v}_{\text{cmd}}, \mathbf{q}_t, \dot{\mathbf{q}}_t] \in \mathbb{R}^{70}, \quad (1)$$

composed of the base linear and angular velocities $\mathbf{v}_t$ and $\boldsymbol{\omega}_t$, respectively, the projected gravity vector $\mathbf{g}_t$, the velocity command $\mathbf{v}_{\text{cmd}}$, and the joint positions and velocities $\mathbf{q}_t$ and $\dot{\mathbf{q}}_t$ relative to the nominal joint configuration. The velocity command $\mathbf{v}_{\text{cmd}} = (v_x, v_y, \omega_{\text{yaw}}) \in [-1, 1] \text{ m s}^{-1} \times [-0.5, 0.5] \text{ m s}^{-1} \times [-1, 1] \text{ rad s}^{-1}$ is resampled during training, with 2% of the environments receiving a turn-in-place command and another 2% a zero command. We do not include the exteroceptive input in the robot state, treating it as a separate observation.

III. METHOD

Our framework, shown in Fig. 2, consists of two learned components: the FPM f_θ , with parameters θ , that predicts the next two footholds from the current system state $\mathbf{x}_t$, and a locomotion policy π_ϕ , with parameters ϕ , whose exteroceptive input is the elevation map sampled around only those predicted footholds. We co-train the two components as summarized in Alg. 1, using PPO to continuously optimize the locomotion policy and refitting, in a self-supervised fashion, the FPM at fixed intervals to rollouts of the current policy.

A. Foothold Prediction Model (FPM)

The FPM models the closed-loop gait of the current locomotion policy,

$$f_\theta : \mathbb{R}^{70} \rightarrow \mathbb{R}^4, \quad \hat{\mathbf{p}}_t = [\hat{\mathbf{p}}_t^1, \hat{\mathbf{p}}_t^2] = f_\theta(\mathbf{x}_t),$$

mapping the unscaled, noise-free system state $\mathbf{x}_t$ of Eq. (1) to the planar positions $\hat{\mathbf{p}}_t^1, \hat{\mathbf{p}}_t^2 \in \mathbb{R}^2$ of the next two footholds in the current base frame. It is deliberately blind: it observes neither the elevation map nor its own previous output.

1) *Prediction target*: As the robot moves during the policy rollouts of the co-training loop, we record its state trajectory together with every new foothold after a foot is airborne, detected from the binary contact indicators c_t^L, c_t^R . Once the rollout is complete, we look backward in time and pair each past state $\mathbf{x}_t$ with the planar positions of the next two footholds in this sequence, irrespective of which foot made them, expressed in the base frame at time t as the label $\mathbf{p}_t = [\mathbf{p}_t^1, \mathbf{p}_t^2]$. The first foothold is the landing location of the foot in the air or about to lift off, and the second is the first contact of the other foot *after* its upcoming lift-off. Falls and terminations reinitialize the robot, so we mask a window only when its label would fall beyond a termination. Pre-fall states whose next foothold still occurred are kept, which keeps the FPM accurate for unstable, near-failure gaits.

2) *Loss*: We train the FPM by supervised regression on the dataset $\mathcal{D}$ of recorded system states and labeled footholds, minimizing the Huber loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{x}_t, \mathbf{p}_t) \sim \mathcal{D}} [\mathcal{L}_\delta(f_\theta(\mathbf{x}_t) - \mathbf{p}_t)]$$

with threshold $\delta = 0.01$, whose bounded gradient keeps outliers near episode boundaries from dominating the fit.

3) *Network*: The FPM is an MLP with two hidden layers with widths $[128, 128]$ followed by layer normalization and LeakyReLU(0.1), with orthogonal weight initialization. It normalizes its input with the per-dimension mean and standard deviation of $\mathbf{x}_t$ over the training split of each refit, stored in the exported model, so deployment needs no external normalization. Table I lists the optimization settings.

B. Foothold-Gated Height Sampling

The locomotion policy’s exteroceptive input is sampled from a robot-centric elevation map $\mathcal{H}_t$ [1], [3], [29], a 2.5D grid of $1.6 \text{ m} \times 1.0 \text{ m}$ at 0.05 m resolution ($33 \times 21 = 693$ heights) in a yaw-aligned frame attached to the base, with heights relative to the base height. We transform each predicted foothold into the map frame, round it to the nearest cell, and gather the $k \times k$ cells around it into a patch $\mathbf{h}_t^1, \mathbf{h}_t^2 \in \mathbb{R}^{k^2}$, with $k = 7$ and clamping at the map boundary; at 0.05 m this is a 0.35 m square, comparable to the 0.208 m length of the Unitree G1 foot. The two patches, each preceded by its predicted foothold, form the height scan

$$\mathbf{h}_t = [\hat{\mathbf{p}}_t^1, \mathbf{h}_t^1, \hat{\mathbf{p}}_t^2, \mathbf{h}_t^2] \in \mathbb{R}^{102},$$

which lets the policy interpret each patch relative to its location and reduces the exteroceptive input from 693 cells to 102 values. The actor observation $\mathbf{s}_t = [\mathbf{x}_t, \mathbf{a}_{t-1}, \mathbf{h}_t] \in \mathbb{R}^{201}$ concatenates the system state, with angular and joint velocities scaled by 0.2 and 0.05, the previous action, and the height scan; the action $\mathbf{a}_t \in \mathbb{R}^{29}$ is a joint-position offset from the nominal configuration, scaled by 0.25 and tracked by a joint-level PD controller.

To keep the policy from over-trusting the query point, we add two forms of randomization to the sampled elevation map: per-episode planar and vertical offsets in $[-0.05, 0.05] \text{ m}$ displace the sampled patches uniformly

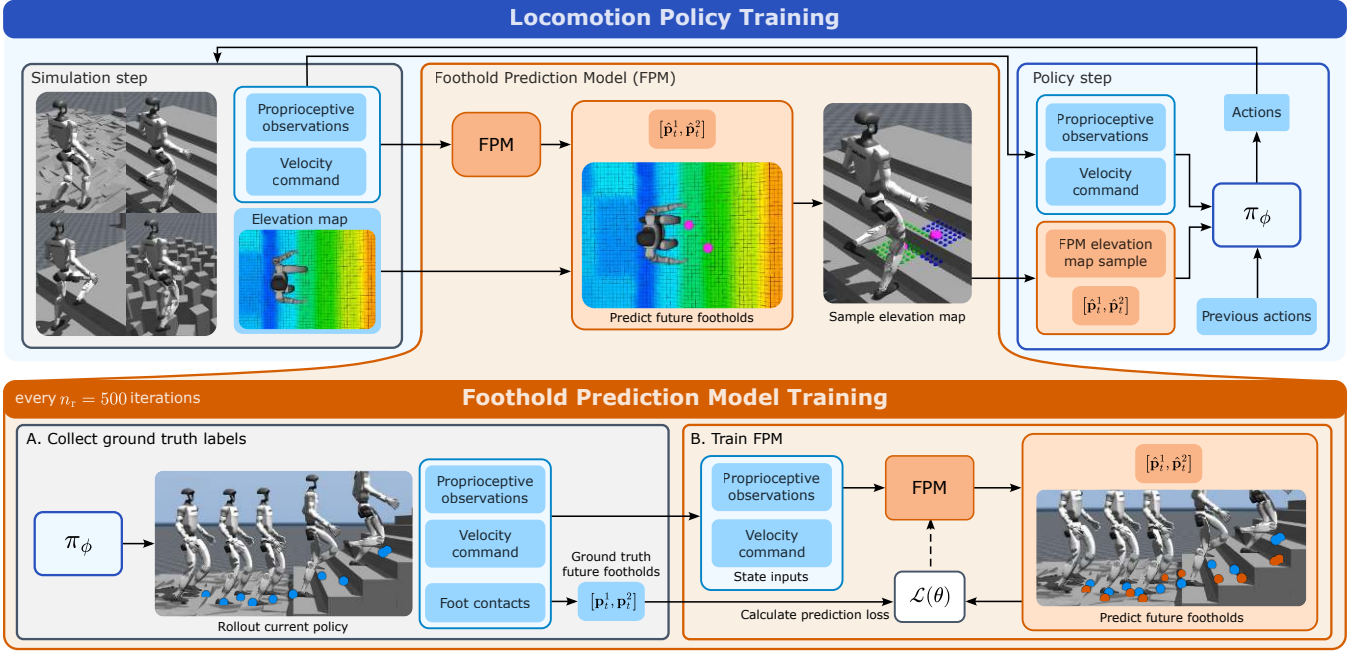


Fig. 2. Method overview. The policy learns to track velocity commands over varied terrain from an elevation map gated around the FPM’s predicted footholds. Every $n_t = 500$ iterations, A. rollouts of the current policy yield ground-truth future footholds and B. the FPM is retrained from scratch to predict them.

while the reported coordinates stay undrifted, emulating misaligned predictions and odometry drift, and noise $\mathcal{U}(-0.1, 0.1)$ m is added to each sampled point. The exteroceptive input is then clipped to $[-1, 1]$. We train with an asymmetric actor-critic: the critic additionally receives the complete 693-point map $\mathcal{H}_t$ and $\mathbf{h}_t$, so the full map informs value estimation during training, but is not needed at deployment.

C. Co-Training Schedule

As described in Alg. 1, we co-train the locomotion policy and the FPM in an alternating loop in which we refit each component to the latest version of the other. The schedule runs in units of PPO iterations, each comprising 24 environment steps for $N_{\text{env}} = 8192$ parallel environments.

1) *Bootstrap*: The policy has not learned to stand or walk at the beginning of training, so no FPM is initially fit. We instead bootstrap the locomotion policy to learn to walk on easy terrain and fix the foothold predictions directly beneath the base so that the policy observes a height scan under its own body. We use this bootstrapping mode for the first $n_0 = 500$ iterations of RL training, at which point the policy walks adequately and its footholds carry information worth learning.

2) *FPM refit*: Every $n_t = 500$ PPO iterations of locomotion policy learning, we roll out the policy and gather $N_s = 5 \times 10^5$ transitions, which takes $\lceil N_s / N_{\text{env}} \rceil = 62$ environment steps. For every recorded step we log the system state $\mathbf{x}_t$, the per-foot contact indicators c_t^L, c_t^R , and the world-frame base and foot poses $\mathbf{r}_t$, from which we label the future footholds $\mathbf{p}_t$ as described in Sec. III-A.1 to form the dataset $\mathcal{D}$. We then retrain the FPM from scratch for 25 epochs

on a random 80/20 train-validation split with shuffled mini-batches, rather than fine-tuning it, so that it cannot inherit a stale gait from earlier in training. The refit FPM is loaded into the observation pipeline, where it gates the policy’s elevation map input for the next n_t iterations before the cycle repeats.

D. Training Curriculum and Implementation

The locomotion policy is trained on procedurally generated terrains with 10 difficulty levels and a progress curriculum similar to previous methods [1]. The terrain suite is dominated by stepped and sparse geometry, composed of: stepping stones (30%; stone width 0.3 m to 0.5 m, gaps 0.1 m to 0.2 m), upright and inverted pyramid stairs (15% each; step height 0.1 m to 0.25 m), and boxes (15%; 0.45 m grid, height 0.05 m to 0.3 m). The remainder comprises upright and inverted pyramid slopes (7.5% and 5%, gradient up to 0.55), random rough ground (5%, noise up to 0.1 m), pits (5%) and flat ground (2.5%).

To facilitate initial terrain traversal learning, we apply an assistive upward force to the torso equal to a fraction $\alpha(l)$ of the robot’s weight, where

$$\alpha(l) = (1 - \beta) \left(1 - \frac{l}{l_{\text{cut}}} \right), \quad \beta = 0.5, \quad l_{\text{cut}} = 4,$$

is clamped to $[0, 1 - \beta]$ and driven by the mean terrain level l across environments, the force decaying to zero at cutoff level l_{cut} . The robot therefore carries half its weight at level 0 and its full weight from level 4 onward, at which point more challenging terrains dominate. The assistance is particularly helpful on sparse terrains given the sparse exteroceptive input, as it gives the policy room to learn to adjust a step mid-air to land on a valid foothold.

TABLE I

Learning parameters for the self-supervised Foothold Prediction Model and parameters and rewards for the RL locomotion policy.

Foothold Prediction Model (FPM)	
Hidden Layers / Activation	[128, 128] / LeakyReLU(0.1)
Input Normalization	z-score, train split
Optimizer / Learning Rate	Adam / 5×10^{-3}
Loss	Huber ($\delta = 0.01$)
Batch Size / Epochs per Refit	1024 / 25
Train-Validation Split	80/20, random
Samples per Refit N_s	5×10^5
Bootstrap n_0 / Refit Interval n_r	500 / 500 iterations
Locomotion Policy (PPO)	
Actor / Critic Hidden Layers	[512, 256, 128], ELU
Parallel Environments N_{env}	8192
Steps per Environment per Iteration	24
Learning Epochs / Mini-Batches	5 / 4
Learning Rate / Desired KL	1×10^{-3} (adaptive) / 0.01
Discount γ / GAE λ	0.99 / 0.95
Clip Range / Entropy Coefficient	0.2 / 0.005
Max Gradient Norm	1.0
Initial Action Noise Std	1.0
Locomotion Reward Terms	
xy Linear Velocity Tracking (exp)	+1.0
Yaw Angular Velocity Tracking (exp)	+1.0
Stand Still at Nominal Pose	+2.5
z Linear Velocity (L2)	-0.5
xy Angular Velocity (L2)	-0.05
Joint Torques (L2)	-10^{-5}
Joint Accelerations (L2)	$-2.5 \cdot 10^{-8}$
Action Rate (L2)	-0.01
Undesired Contacts	-1.0
Joint Deviation Legs / Waist / Arms (L1)	-1.0 / -1.0 / -0.1
Termination (Base Tilt > 0.7 rad)	-20

Training implementation details are presented in Table I, including PPO training hyperparameters and the training reward terms used. The reward set consists of task rewards for successfully tracking the velocity command, including proper stand-still behavior, and regularization terms that minimize undesirable behavior. Leveraging the robot’s symmetry, we augment each batch with the mirrored transition instead of adding a symmetry loss [30].

We use standard domain randomization parameters during training [31] to modify environmental and robot model parameters or apply external disturbances to add robustness to inaccurate modeling, non-ideal terrain conditions, or potential events that may occur to the robot. In addition, we add uniform noise to all proprioceptive observations and inject occasional large spikes into the base velocity observations to emulate state-estimator faults.

IV. EXPERIMENTS AND RESULTS

In this section, we present simulation and hardware experiments alongside comparative results with state-of-the-art baselines and ablations regarding key choices of our method. We leave the broader interpretation of the results to Sec. V.

We compare our FPM-based locomotion policy against five alternative methods. We evaluate five trained seeds per method over 1000 episodes per terrain on the training terrain

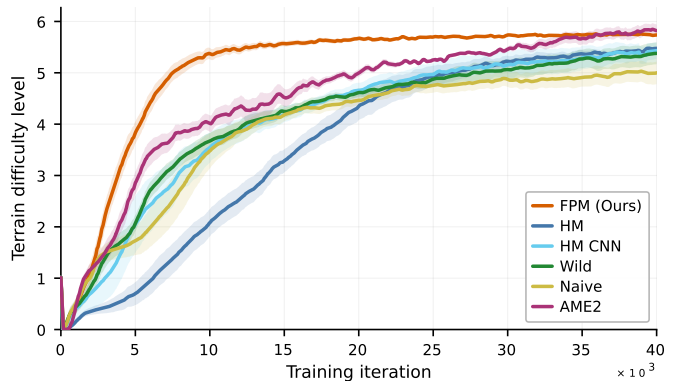


Fig. 3. Mean terrain-curriculum level during training over five seeds; shaded bands show one standard deviation. FPM reaches level 5 in 7.5k iterations, $3.5\times$ faster than HM and $2.7\times$ faster than AME2.

set and five previously unseen, out-of-distribution (OOD) terrains: undulating waves with amplitude 0.05 m to 0.25 m and wavelength 7.5 m, discrete rectangular obstacles 0.5 m to 1.0 m tall and 0.2 m wide, high-amplitude random terrain of 0.05 m to 0.15 m, concentric 0.35 m gaps, and sparse rectangular pallets. All cases share the policy architecture and PPO hyperparameters of Table I, the same reward set, the terrain and assistive-force curricula, the same task inputs, and asymmetric critics.

The five alternative methods are as follows. *HM* provides the complete elevation map alongside the proprioceptive and task inputs. Two alternative sampling strategies test whether a partial map around the current feet or along the commanded direction suffices: *Wild* is the cylindrical height scan of Miki et al. [2], which queries the map around the current position of each foot, and *Naive* crops a 10×20 patch of 200 values from the map, centered at a point below the torso and shifted 0.25 m per m s^{-1} of the commanded velocity, a velocity-proportional nominal placement akin to the Raibert heuristic [32]. Finally, we compare against methods that train an encoder alongside the locomotion policy to extract features from the full elevation map. *HM CNN* passes the map through a CNN encoder, and *AME2* is inspired by the attention-based encoder from Zhang et al. [10] conditioned on velocity commands.

A. Learning speed

Fig. 3 shows the average terrain curriculum level during training. FPM converges the fastest, after $\sim 15\text{k}$ iterations, and reaches level 5 after 7.5k iterations, HM after 26.1k, HM CNN after 25.2k, Wild after 27.6k, Naive after 37.9k, and AME2 after 20.0k. This is a speedup of 3.4 to $3.5\times$ over the full-map methods and $2.7\times$ over the attention encoder of AME2. AME2 ends marginally above FPM at 40k iterations, but at a much higher computational cost, requiring four GPUs to train and taking 6.3s per policy update against 0.55s for FPM, with equal rollout collection time. FPM also shows less variance across seeds throughout training, with one seventh of the spread at 40k iterations.

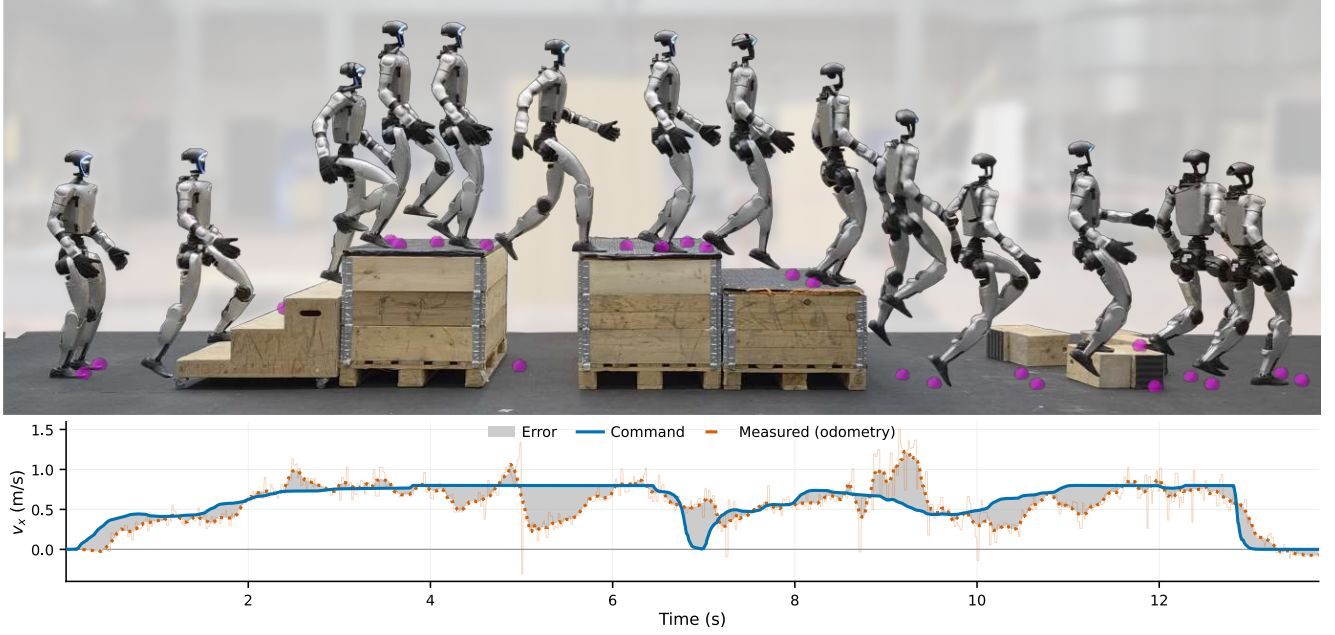


Fig. 4. Sim-to-real transfer to a Unitree G1 demonstrating locomotion over challenging obstacles, including stairs and rough terrain expected within the training distribution, and a 0.55 m gap and 0.6 m high drop from outside the training distribution. Predicted footholds are in magenta for each step. The commanded x-velocity and the smoothed measured x-velocity are shown for the length of the course.

TABLE II

Command tracking success and fall rate for locomotion and corridor benchmarks. Stones is a training terrain, Gaps and Pallets are OOD. Best result in each column marked in **bold**, second best in *italics*.

Method	Success (%)		Sparse terrain (%)			Falls (%)		Corridor (%)			
	Train	OOD	Stones	Gaps	Pall.	Train	OOD	All	Sparse	Step.	Cont.
FPM (ours)	84.6	<i>83.1</i>	80.2	77.7	<i>81.0</i>	<i>3.3</i>	5.2	89.2	74.2	85.6	99.5
HM	81.6	80.8	69.0	<i>70.5</i>	73.2	4.6	6.2	87.8	42.7	<i>93.8</i>	98.7
HM CNN	81.9	78.5	65.7	58.9	67.9	5.4	13.3	88.7	44.8	94.3	99.6
Wild	80.2	78.6	66.4	66.7	67.0	4.7	9.2	82.5	36.0	86.5	96.3
Naive	77.6	71.8	57.3	47.3	59.2	7.6	13.8	76.0	42.5	73.5	92.4
AME2	<i>84.3</i>	83.6	78.8	69.8	87.1	2.4	5.5	87.3	<i>71.7</i>	86.8	94.2

B. Terrain traversal performance

We compare terrain traversal performance in simulation in Table II. The table aggregates the results of two tests: the locomotion test, tracking a randomly sampled linear velocity command on 8 m tiles, and the corridor test, tracking a forward velocity command through a narrow corridor with tall walls. The corridor test checks how each policy handles OOD terrain outside of the robot’s commanded path. We categorize terrains as *Sparse* (e.g., stepping stones), *Stepped* (e.g., boxes), and *Continuous* (e.g., rough). Both tests use the terrain floors, but the corridor constrains the terrain to a 1.3 m wide hall enclosed by one of three wall types: solid walls 1.5 m to 2.5 m tall, a lattice of 0.20 m pillars with gaps, and recessed blocks that modulate the hall’s width from 1.3 m to 1.5 m.

We define the success rate as the ratio of the distance traversed in the commanded direction to the distance the velocity command would have covered over the episode, so a fall counts against the full commanded distance. Falls

therefore lower the success rate, and we also report the fall rate of the locomotion test in Table II. A success rate of 100% is not realistically achievable as it would require perfect velocity command tracking over the entire length of the course.

Our FPM policy performs best or second best in most experiments. In the locomotion test, FPM has the highest success rate on the training terrains (84.6%), the best performance on the sparse stepping stones (80.2%) and gaps (77.7%), and the lowest fall rate on the OOD terrains (5.2%). AME2 performs better on the OOD terrains (83.6% against 83.1%), including the sparse pallets (87.1%), and has a lower fall rate on the training terrains (2.4%). Both policies outperform the other tested methods on aggregate performance in this test. This demonstrates that our lightweight sampling approach matches the performance of a much more expensive attention-based approach.

The results of the corridor test are more conclusive about the benefits of the FPM sampling approach on sparse terrain. FPM has the highest overall success rate (89.2%), scoring highest on the sparse terrains (74.2%) and second highest on the continuous terrains (99.5%). AME2 performs second best on the sparse terrains (71.7%), and both are well ahead of all other approaches, which cover less than half the commanded distance.

C. Sim-to-Real

We deployed our FPM-based locomotion policy in indoor and outdoor environments on a variety of terrains, as shown in the videos on the project page. Fig. 4 shows the Unitree G1 traversing a course of in-distribution and OOD terrains. The terrain consists of stairs, a 0.55 m gap and a 0.6 m drop outside the training distribution, and a variety of blocks.

TABLE III

Ablation over prediction horizon and the addition of an exteroceptive CNN encoder. Best result in each column marked in **bold**, second best in *italics*.

Method	Success (%)		Falls (%)		Corridor (%)			
	Train	OOD	Train	OOD	All	Sparse	Step.	Cont.
FPM	84.6	83.1	3.3	5.2	89.2	74.2	85.6	99.5
FPM1	81.3	78.8	4.8	5.0	84.7	62.2	83.1	95.6
FPM3	79.5	77.3	2.1	4.4	85.0	52.7	86.3	96.4
FPM CNN	82.4	81.1	1.4	4.0	85.6	56.2	85.8	97.1
FPM1 CNN	82.7	82.0	2.3	4.2	86.3	62.7	86.7	95.3
FPM3 CNN	79.3	78.9	2.2	6.5	84.8	53.8	86.2	95.5

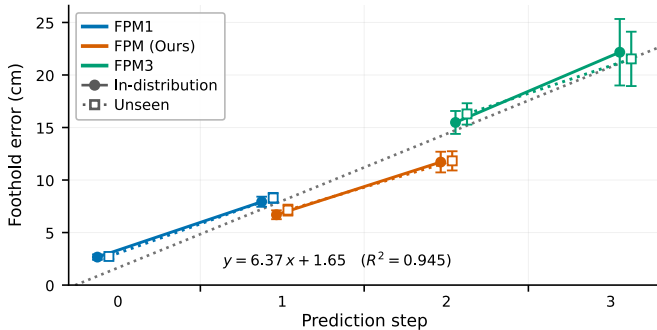


Fig. 5. Foothold prediction error by prediction step of various FPM horizons for training and OOD terrains and a regression line describing the relationship between prediction step and mean error. Whiskers show one standard deviation across seeds.

The linear velocity command is plotted along the length of the course, showing the tracking performance relative to the obstacle. We observe the greatest error between the commanded and measured velocity when traversing the OOD obstacles.

The policy demonstrates a variety of learned behaviors, not explicitly rewarded, that keep locomotion safe on unseen terrains. The policy slows down before crossing the gap, which we hypothesize is caused by terrain uncertainty in the FPM-gated elevation map. It shows similar cautious behavior before drops, slowing down and lowering its body before jumping off with one foot forward to catch itself, showing emergent behavior for terrains unseen in training.

The FPM-based locomotion policy additionally traversed a variety of outdoor terrains, including a staircase with 18 continuous steps. Shown in Fig. 1, the policy also successfully traverses rough terrain in corridors, validating the simulation performance of the corridor test in Sec. IV-B. The reader is encouraged to watch the videos on the project page for a full overview of the conducted hardware tests.

D. Ablations

We examine how the foothold prediction horizon affects locomotion performance. We define two alternative horizons, denoted by the furthest step in the future that they predict, so our FPM is FPM2 under this naming. All variants output two foothold predictions from the contact sequence but differ in the horizon. Our FPM predicts the next two future footholds, p^1 and p^2 , FPM1 pairs the latest foothold p^0 with p^1 , and FPM3 outputs p^2 and p^3 . FPM1 tests whether the terrain

around the latest and the next foothold suffices for the current policy step. FPM3 instead explores whether a longer horizon benefits locomotion at the cost of no longer observing the next foothold. We also test adding a CNN encoder to each FPM ablation, trained alongside the policy, to extract further features from the sampled elevation-map patches.

As shown in Table III, FPM1 performs below FPM, most clearly on the sparse corridor terrains (62.2% against 74.2%). The CNN encoder improves the success rate of FPM1 on both the training and OOD terrains and gives it the second best overall corridor performance (86.3%). FPM CNN has the lowest fall rate on both the training and OOD terrains (1.4% and 4.0%). This suggests possible benefits of additionally encoding the sampled patches; however, FPM without an encoder still performs best overall, suggesting that the samples already form effective features for the locomotion policy.

FPM3 performs lower overall on the locomotion test. However, FPM3 shows a lower fall rate than FPM and FPM1, and performs well on the stepped corridor terrains (86.3%). Fig. 5 reports the error of each FPM horizon at each prediction step on the training and OOD terrains. It shows that FPM3 predicts the second future foothold less accurately than FPM, and that its third-step prediction has roughly 22 cm of error, approximately the length of the G1 foot. Adding a CNN to FPM3 leaves success within two points of FPM3 and does not lower the OOD fall rate, likely because the further features lie too far ahead of the robot to matter for the current step.

Fig. 5 also shows a roughly linear relationship between the prediction horizon and the error, of roughly 6 cm per step. Given the Unitree G1 foot of 20.8 cm length and 7.6 cm width, this suggests that the prediction is accurate to within a foot length up to the second future foothold. The standard deviation increases with the horizon, likely due to the difficulty of predicting far ahead on rough terrain from a current motion that is only weakly related to it.

V. DISCUSSION AND LIMITATIONS

Our co-trained, state-based sampling shrinks the perception learning problem: rather than finding the relevant cells in the full map, the policy learns how to use two small patches chosen for it. FPM learns faster than the tested full-map methods (Sec. IV-A). Wild and Naive, which sample as many heights as FPM or more, but around the current feet or along the commanded direction, perform worse than HM (Table II), suggesting that the benefit comes from placing the patches at the predicted footholds, not from the smaller observation. Despite the limited exteroceptive observations, the FPM-based policy performs best on most tested sparse terrains. On hardware, it slows down and adjusts its body and steps before crossing large gaps or stepping off high drops (Sec. IV-C), demonstrating emergent cautious behavior. We conjecture that because the observations depend on the locomotion policy’s actions, this may encourage it to probe the terrain before committing while still tracking the velocity command.

AME2 is the only alternative that matches FPM on the locomotion test, at a much higher training cost of 6.3 s per policy update on four GPUs compared to 0.55 s on a single GPU for FPM. Its weaker corridor results (Sec. IV-B) suggest that the attention mechanism can be distracted by obstacles it has never seen, whereas the FPM never shows the policy the wall outside the commanded direction. The corridor test also marks the limits of our approach. Full-map observation methods lead FPM on the stepped terrains, and all methods perform similarly on continuous terrain, where a dense scan suffices. The FPM discards all terrain the robot does not intend to step on, and obstacles approaching from the side would enter the patches late or not at all, which our corridor walls, parallel to the commanded path, do not test.

Finally, our effective prediction horizon is limited to predicting the second future foothold. The FPM3 ablation (Fig. 5) shows that the step predictions become less accurate with greater variance as the horizon increases, also degrading the locomotion policy's performance (Table III).

VI. CONCLUSION

In this paper, we presented a perceptive locomotion framework for humanoid robots in which a lightweight foothold prediction model, trained alongside the locomotion policy, predicts the next two footholds from proprioception and the velocity command, and we sample the elevation map only around them. The policy learns several times faster than policies that observe the complete map, matches an attention-based encoder at a fraction of its training cost, performs best on most sparse terrains, and transfers zero-shot to a Unitree G1.

In the future this work may be extended to an active depth-based perception system that does not rely on an elevation map, but instead directly observes the future predicted footholds. This approach may also extend to manipulative locomotion, using predicted hand and foot contacts to guide the policy to observe points for arm and leg support.

ACKNOWLEDGMENT

The authors acknowledge the use of generative AI to assist with code generation and language refinement. We thank Tianxu An, Pwint Thu Aung, Dario Hug, Shengzhi Wang, and Fan Yang for their assistance with testing and filming, Victor Klemm for discussions that inspired the assistive force curriculum, and René Zurbrugg for his help in preparing the visuals for this work.

REFERENCES

- [1] N. Rudin *et al.*, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. Conf. Robot Learn. (CoRL)*, 2022, pp. 91–100.
- [2] T. Miki *et al.*, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, p. eabk2822, 2022.
- [3] P. Fankhauser *et al.*, "Robot-centric elevation mapping with uncertainty estimates," in *Mobile Service Robotics*. World Scientific, 2014, pp. 433–440.
- [4] J. S. Matthis, J. L. Yates, and M. M. Hayhoe, "Gaze and the control of foot placement when walking in natural terrain," *Curr. Biol.*, vol. 28, no. 8, pp. 1224–1233.e5, 2018.
- [5] D. Hoeller *et al.*, "ANYmal parkour: Learning agile navigation for quadrupedal robots," *Sci. Robot.*, vol. 9, no. 88, p. eadi7566, 2024.
- [6] X. Cheng *et al.*, "Extreme parkour with legged robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 11 443–11 450.
- [7] J. Long *et al.*, "Learning humanoid locomotion with perceptive internal model," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2025, pp. 9997–10 003.
- [8] Q. Ben *et al.*, "Gallant: Voxel grid-based humanoid locomotion and local-navigation across 3-D constrained terrains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2026, pp. 28 086–28 095.
- [9] J. He *et al.*, "Attention-based map encoding for learning generalized legged locomotion," *Sci. Robot.*, vol. 10, no. 105, p. eadv3604, 2025.
- [10] C. Zhang *et al.*, "AME-2: Agile and generalized legged locomotion via attention-based neural map encoding," arXiv:2601.08485, 2026.
- [11] S. Fu *et al.*, "Global-local attention decomposition for terrain encoding in humanoid perceptive locomotion," arXiv:2606.00637, 2026.
- [12] P. Li *et al.*, "TAGA: Terrain-aware active gaze learning for generalizable agile humanoid locomotion," arXiv:2606.05880, 2026.
- [13] Z. Zhuang, S. Yao, and H. Zhao, "Humanoid parkour learning," in *Proc. Conf. Robot Learn. (CoRL)*, 2024.
- [14] R. Yu *et al.*, "SSR: Scaling surefooted and symmetric humanoid traversal to the open world," arXiv:2605.30770, 2026.
- [15] D. Kanoulas and M. Vona, "Bio-inspired rough terrain contact patch perception," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 1719–1724.
- [16] D. Kanoulas, N. G. Tsagarakis, and M. Vona, "Curved patch mapping and tracking for irregular terrain modeling: Application to bipedal robot foot placement," *Robot. Auton. Syst.*, vol. 119, pp. 13–30, 2019.
- [17] S. Fahmi *et al.*, "ViTAL: Vision-based terrain-aware locomotion for legged robots," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 885–904, 2023.
- [18] K. Wang *et al.*, "When and where to step: Terrain-aware real-time footstep location and timing optimization for bipedal robots," *Robot. Auton. Syst.*, vol. 179, p. 104742, 2024.
- [19] O. A. Villarreal Magaña *et al.*, "Fast and continuous foothold adaptation for dynamic locomotion through CNNs," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2140–2147, 2019.
- [20] D. Belter *et al.*, "Single-shot foothold selection and constraint evaluation for quadruped locomotion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 7441–7447.
- [21] Z. Xie *et al.*, "ALLSTEPS: Curriculum-driven learning of stepping stone skills," *Comput. Graph. Forum*, vol. 39, no. 8, pp. 213–224, 2020.
- [22] H. Wang *et al.*, "BeamDojo: Learning agile humanoid locomotion on sparse footholds," in *Proc. Robot.: Sci. Syst. (RSS)*, 2025.
- [23] F. Jenelten *et al.*, "DTC: Deep tracking control," *Sci. Robot.*, vol. 9, no. 86, p. eadh5401, 2024.
- [24] A. Montenegro *et al.*, "Mind your steps: A general learning framework for accurate humanoid foothold tracking," in *Proc. Robot.: Sci. Syst. (RSS)*, 2026.
- [25] M. Dai *et al.*, "Walk the PLANC: Physics-guided RL for agile humanoid locomotion on constrained footholds," arXiv:2601.06286, 2026.
- [26] V. Tsounis *et al.*, "DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3699–3706, 2020.
- [27] J. Schulman *et al.*, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.
- [28] NVIDIA *et al.*, "Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning," arXiv:2511.04831, 2025.
- [29] T. Miki *et al.*, "Elevation mapping for locomotion and navigation using GPU," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2022, pp. 2273–2280.
- [30] M. Mittal *et al.*, "Symmetry considerations for learning task symmetric robot policies," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2024, pp. 7433–7439.
- [31] Z. Luo *et al.*, "SONIC: Supersizing motion tracking for natural humanoid whole-body control," *Sci. Robot.*, vol. 11, no. 117, p. ead4592, 2026.
- [32] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA: MIT Press, 1986.