TTT-Parkour: Rapid Test-Time Training for Perceptive Robot Parkour

Shaoting Zhu[12*], Baijun Ye[12*], Jiaxuan Wang[1†], Jiakang Chen[1†], Ziwen Zhuang[12],
Linzhan Mou[3], Runhan Huang[1], Hang Zhao[12‡]

[1]Tsinghua University, [2]Shanghai Qi Zhi Institute, [3]Princeton University
[*]Equal contribution [†]Equal contribution [‡]Corresponding author

***Abstract*—Achieving highly dynamic humanoid parkour on unseen, complex terrains remains a challenge in robotics. Although general locomotion policies demonstrate capabilities across broad terrain distributions, they often struggle with arbitrary and highly challenging environments. To overcome this limitation, we propose a real-to-sim-to-real framework that leverages rapid test-time training (TTT) on novel terrains, significantly enhancing the robot's capability to traverse extremely difficult geometries. We adopt a two-stage end-to-end learning paradigm: a policy is first pre-trained on diverse procedurally generated terrains, followed by rapid fine-tuning on high-fidelity meshes reconstructed from real-world captures. Specifically, we develop a feed-forward, efficient, and high-fidelity geometry reconstruction pipeline using RGB-D inputs, ensuring both speed and quality during test-time training. We demonstrate that *TTT-Parkour* empowers humanoid robots to master complex obstacles, including wedges, stakes, boxes, trapezoids, and narrow beams. The whole pipeline of capturing, reconstructing, and test-time training requires less than 10 minutes on most tested terrains. Extensive experiments show that the policy after test-time training exhibits robust zero-shot sim-to-real transfer capability. Project Page: https://ttt-parkour.github.io.**

I. Introduction

Recent advancements in deep reinforcement learning have fundamentally revolutionized humanoid locomotion control, enabling robots to demonstrate robust mobility in diverse real-world environments [15, 35, 55]. By leveraging massive parallel simulation and sim-to-real transfer techniques [32, 31], humanoid robots can now traverse unstructured terrains. Although some general policies demonstrate capabilities across broad terrain distributions, they struggle to traverse unseen and complex obstacles. Bridging this gap to achieve true athletic intelligence is critical for the deployment of humanoids in challenging environments.

Although large-scale simulation training has improved the capabilities of humanoid robots, relying solely on procedural generation to create terrains has inherent limitations. Synthetic terrains composed of simple geometric primitives often fail to capture the vast spectrum of terrain typologies and their complex spatial configurations in the real world. It is impossible to exhaustively cover every potential environment during pre-training. A policy trained on such constrained data distributions inevitably suffers from the out-of-distribution deployed in the real world. Thus, there is a critical need for a paradigm that enables rapid adaptation at test time.

Moreover, manually reproducing realistic geometric features in simulation is labor-intensive. While per-scene optimization methods like NeRF [22, 53, 47, 5] and 3DGS [29, 34, 48, 24] excel in view synthesis and extend to physical interaction [10, 54, 14], the overall workflow is often computationally intensive and time-consuming, making it incompatible with the rapid adaptation requirements of test-time training. Conversely, feed-forward [42, 43, 44, 21] and generative approaches [7] are faster but often yield scale-ambiguous or distorted geometries in some terrains, leaving them unsuitable for physics simulation in our parkour settings. Thus, generating collision-accurate, simulation-ready meshes within the tight time windows required for rapid adaptation remains a critical bottleneck.

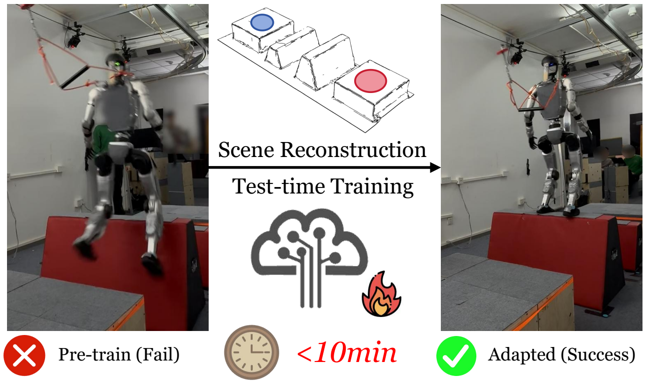


Fig. 1: Rapid test-time training on unseen terrain. By reconstructing the scene and fine-tuning in simulation, our framework enables the robot to master challenging obstacles within 10 minutes, turning failure (left) into success (right).

To address these challenges, we introduce *TTT-Parkour*, a real-to-sim-to-real framework designed for rapid humanoid adaptation on challenging terrains, including wedges, stakes, boxes, trapezoids, and narrow beams. Our approach is built upon a two-stage end-to-end perceptive locomotion learning paradigm. We first pre-train a general policy on a diverse set of procedurally generated terrains. Subsequently, we fine-tune the policy on meshes reconstructed from real-world terrains. We develop an efficient and high-fidelity geometry reconstruction pipeline using RGB-D input. We employ a feed-forward method with automatic scale recovery and frame alignment to directly reconstruct simulation-ready meshes. This pipeline enables test-time training of the policy on accurate geometric constraints, significantly accelerating adaptation and mitigating the sim-to-real gap. Notably, our efficient and automated framework allows for rapid adaptation. It finishes the capture, reconstruction, and test-time training phases within 10 minutes for most tested terrains. This allows the robot to rapidly update its policy, ensuring robust and agile parkour performance even when encountering geometric irregularities that were never seen during pre-training. Extensive experiments show that both the pre-training stage with curriculum learning and the test-time training stage with specific terrain are essential for achieving robust performance on extremely challenging terrains. In summary, our main contributions are as follows:

- We introduce a **two-stage end-to-end perceptive locomotion learning paradigm** consisting of pre-training and rapid test-time training, both of which are essential for traversing extremely challenging terrains.
- We develop a **fast, feed-forward and high-fidelity geometry reconstruction pipeline** to generate simulation-ready mesh from RGB-D inputs, enabling an efficient real-to-sim-to-real parkour workflow.
- Experiments demonstrate that agile and robust **humanoid parkour capabilities** emerge rapidly on extremely challenging terrains, significantly surpassing baselines.

II. Related Works

A. Perceptive Locomotion

Integrating exteroceptive perception is crucial for agile locomotion, enabling robots to transition from blind, reactive recovery [23, 18, 13] to proactive obstacle traversal. Traditional map-based approaches typically leverage LiDAR coupled with precise localization to construct elevation maps [15, 28, 41, 35] or voxel grids [4]. However, these methods are susceptible to state estimation drift and motion distortion during high-dynamic parkour, making global maps unreliable on extremely challenging terrains. Alternatively, recent works have explored utilizing depth images for policy input [37, 38, 9], retaining intermediate heightmap representations. Following the paradigm of [56, 8, 55], we employ a forward depth camera to train a policy end-to-end. This approach is robust during high-speed traversal. Furthermore, the higher frequency of depth cameras compared to LiDARs makes them inherently more suitable for tasks requiring precise foothold selection. However, most learning-based approaches, including both heightmap-based and end-to-end methods, remain limited to structured terrains and often fail to generalize to unstructured, complex environments with extremely sparse footholds.

B. Fast Adaptation

Test-Time adaptation [45, 39, 3, 2] originates from classical machine learning as a lightweight online fine-tuning paradigm for mitigating out-of-distribution shifts during inference, and has been extended in foundation models with gradient updates [3, 1]. In robot learning, fast adaptation at test time has been extensively explored in recent years: Diffusion-based controllers [19, 27, 17] incorporate gradients of explicit reward and cost functions as guidance at test time to optimize for desired behavioral outcomes. While this class of methods offers flexibility, the generated trajectories may be pulled away from the in-distribution data manifold by gradient updates, resulting in performance collapse. Others use generative models for test-time adaptation to novel environments and embodiments [6]. To the best of our knowledge, we are the first to leverage rapid test-time training paradigm to enable humanoid robots to master unseen, complex terrains.

C. Scene Reconstruction

High-fidelity reconstruction is critical for enhancing simulation realism and bridging the sim-to-real gap. While per-scene optimization methods like NeRF [22, 53, 47, 5] and 3DGS [29, 48, 34, 46, 49, 52, 24, 25, 26] excel in visual synthesis, and recent extensions extract meshes for physical interaction [10, 54, 14, 40], they fail to meet the automation and efficiency requirements of test-time training (TTT). Their reliance on offline, multi-stage processes (e.g., COLMAP or iterative optimization) [54, 29] prevents rapid online adaptation. Conversely, recent feed-forward techniques [42, 43, 21] bypass optimization but suffer from scale ambiguity or exhibit significant metric discrepancies. While RoLA [51] enables manipulation learning from a single image, it is confined to tabletop settings with small objects. In contrast, parkour environments involve long-span terrains featuring complex layouts and occlusions. Thus, the spatial layouts from single-image generation [7] often contain severe geometric distortions. Such geometric infidelities make the resulting meshes unsuitable for parkour tasks. To address this, we introduce an efficient pipeline integrating feed-forward reconstruction with automatic scale and frame alignment, producing *simulation-ready* meshes with the speed and fidelity required for test-time parkour training.

III. Method

A. Problem Definition

We define the task as traversing a series of discrete platforms $\mathcal{P} = \{p_{\text{start}}, p_1, \dots, p_n, p_{\text{end}}\}$ elevated above the ground $\mathcal{G}_{\text{ground}}$. The robot aims to travel from p_{start} to p_{end} following a fixed forward velocity command without an explicit angular velocity command. To ensure valid traversal, contact with the ground plane $\mathcal{G}_{\text{ground}}$ is treated as a failure state, preventing the robot from bypassing obstacles by moving on the ground. The terrains consist of geometric primitives with limited contact areas (e.g., wedges, stakes, boxes, trapezoids, and narrow beams), requiring the policy to maintain stability through precise foothold selection.

B. Policy Pre-training

We formulate the perceptive locomotion task as a Reinforcement Learning (RL) problem and optimize the policy using Proximal Policy Optimization (PPO) [36]. The policy employs a CNN-based depth encoder to extract latent features, which are then concatenated with proprioception and fed into an MLP to predict the final actions.

Observations: The policy's observation space is designed to provide comprehensive state information for stable locomotion. The actor's observation $\mathbf{o}_t^a$ incorporates both proprioception data and visual perception. Specifically, the proprioception

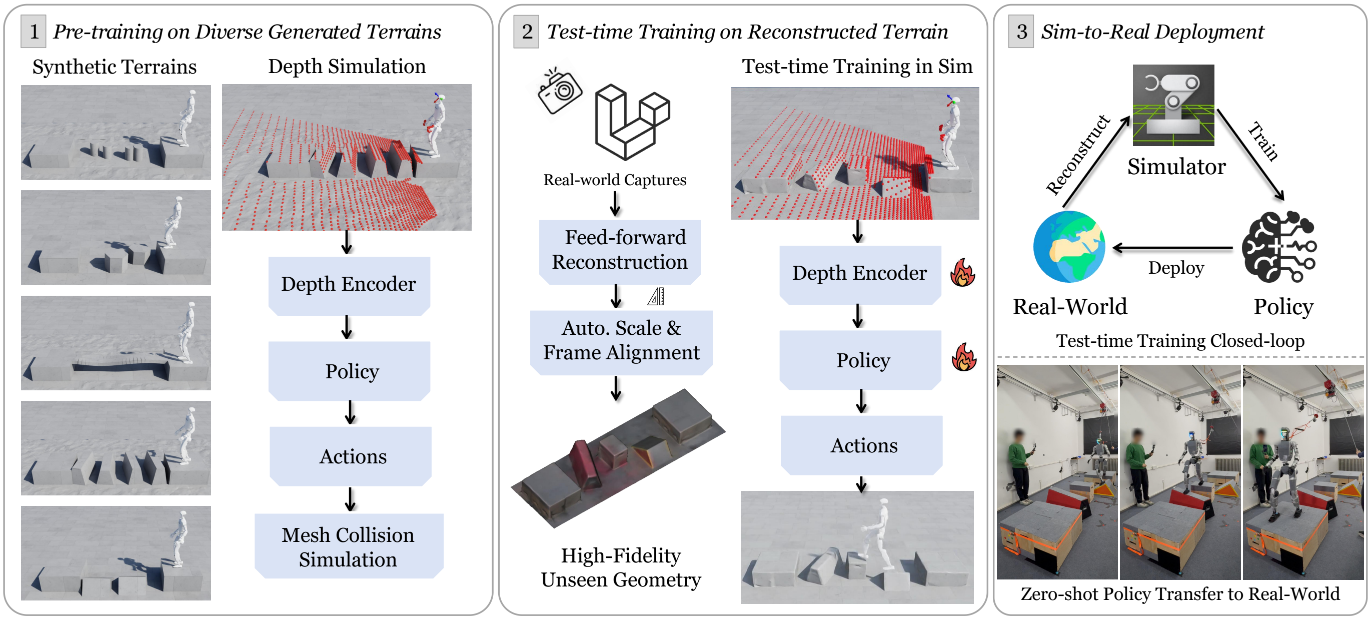

1 Pre-training on Diverse Generated Terrains
Synthetic Terrains
Depth Simulation
Depth Encoder
Policy
Actions
Mesh Collision Simulation
2 Test-time Training on Reconstructed Terrain
Real-world Captures
Feed-forward Reconstruction
Auto. Scale & Frame Alignment
High-Fidelity Unseen Geometry
Test-time Training in Sim
Depth Encoder
Policy
Actions
3 Sim-to-Real Deployment
Simulator
Reconstruct
Train
Deploy
Real-World
Policy
Test-time Training Closed-loop
Zero-shot Policy Transfer to Real-World

Fig. 2: **TTT-Parkour**. Our framework consists of three stages: (1) Pre-training: A general policy is pre-trained on diverse procedurally generated terrains to learn robust locomotion primitives. (2) Test-time Training (TTT): We reconstruct high-fidelity and simulation-ready meshes from real-world captures using feed-forward reconstruction with automatic scale recovery and frame alignment. The policy is then rapidly fine-tuned on these specific terrains in simulation. (3) Sim-to-Real Deployment: The adapted policy is directly deployed to the real-world humanoid robot for zero-shot traversal of complex unseen obstacles.

includes the base angular velocity $\boldsymbol{\omega}_t$, projected gravity vector $\mathbf{g}_t$, velocity commands $\mathbf{c}_t$, joint positions $\mathbf{q}_t$, joint velocities $\dot{\mathbf{q}}_t$, and the previous action $\mathbf{a}_{t-1}$. To handle partial observability and capture motion dynamics, we employ a history sliding window of length h. The final observation is a concatenation of the proprioceptive history and the sequence of depth images $\mathbf{I}_t \in \mathbb{R}^{W\times H}$:

$$\mathbf{o}_t^a = \left[\mathbf{p}_{t-h+1:t}, \mathbf{H}_t\right], \tag{1}$$

where $\mathbf{p}_t = (\boldsymbol{\omega}_t, \mathbf{g}_t, \mathbf{c}_t, \mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{a}_{t-1})$ denotes the proprioception vector at time t. We use strided windows for depth images to get a long history.

$$\mathbf{H}_t = \{\mathbf{I}_{t-k\cdot\ell} \mid k = 0, 1, \ldots, m-1\}, \tag{2}$$

During training, we inject stochastic noise into the actor's input to enhance robustness and bridge the sim-to-real gap.

We adopt an asymmetric actor-critic architecture. The critic has access to privileged information, including noise-free state and base linear velocity $\mathbf{v}_t \in \mathbb{R}^3$, to guide the learning process.

Actions: The policy outputs target joint positions $\mathbf{a}_t \in \mathbb{R}^{29}$. These are converted into joint torques $\boldsymbol{\tau}_t$ via a PD controller:

$$\boldsymbol{\tau}_t = k_p(\mathbf{a}_t - \mathbf{q}_t) - k_d\dot{\mathbf{q}}_t. \tag{3}$$

where the gains k_p and k_d are adopted from [27].

Terminations: The training episode terminates if any of the following conditions are met: (1) The robot is stuck at the starting position for more than 4 seconds; (2) Any body link contacts the ground; (3) The base orientation exceeds the permissible thresholds.

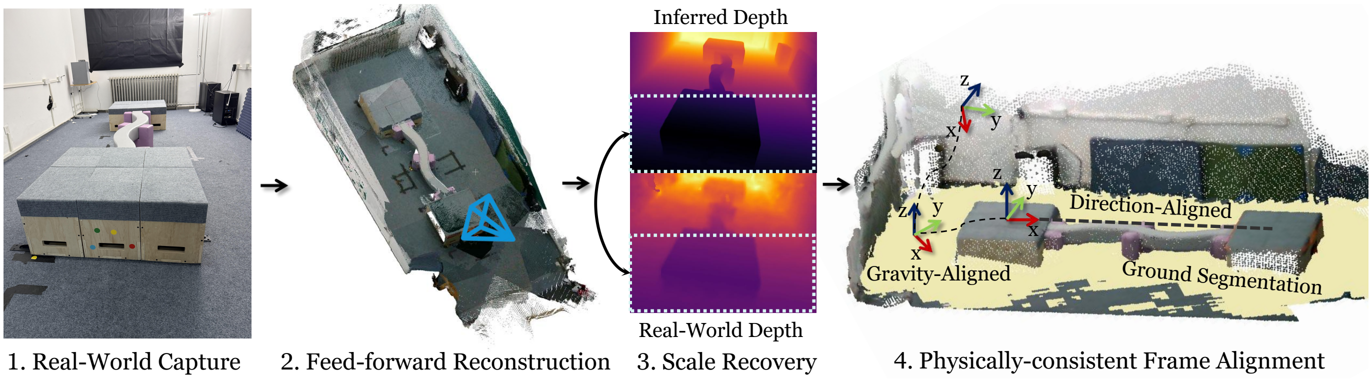
Rewards: The reward function comprises task r_{task}, regularization r_{reg}, safety r_{safe}, and AMP r_{AMP} terms. We formulate the task as goal position tracking, where the target velocity is derived from the goal vector and clipped to a maximum value to prevent reward hacking (e.g., turning around at the start). Notably, we do not provide angular commands; the robot must autonomously decide its steering. We utilize a dense velocity-tracking reward to regulate speed rather than a sparse goal reward. Regularization terms penalize foothold on terrain edges, energy consumption, and action rate to prevent oscillations, while safety terms enforce joint limits. Furthermore, we leverage Adversarial Motion Priors (AMP) [33] trained on MPC-generated datasets [12] to encourage natural and robust motion styles. See Appendix for details.

C. *Efficient Geometry Reconstruction*

To facilitate rapid test-time training to unseen terrains, we introduce an efficient, automated, and high-fidelity reconstruction pipeline that integrates feed-forward reconstruction with automatic scale recovery and frame alignment, shown in Figure 3. Formally, we define the reconstruction problem as transforming raw real-world captures $\mathcal{P}_{1:n}$ into *simulation-ready* mesh $\mathcal{M}$ that is strictly aligned with both the gravity axis $\mathbf{g}$ and the start-to-goal traversal direction $\mathbf{d}$.

Feed-forward Terrain Reconstruction: We initiate the reconstruction process by employing a feed-forward model [43] that takes RGB sequences as input to reconstruct a scale-ambiguous point cloud. Subsequently, we apply screened poisson surface reconstruction [20] to recover the mesh.

Scale Recovery: Existing rgb-only and metric feed-forward



1. Real-World Capture 2. Feed-forward Reconstruction 3. Scale Recovery 4. Physically-consistent Frame Alignment

Fig. 3: **Efficient Geometry reconstruction.** Our pipeline consists of four stages: (1) Real-World Capture. (2) Feed-forward Reconstruction provides initial scene geometry from RGB sequences. (3) Scale Recovery corrects metric scale discrepancies by aligning inferred depth with sensor depth. (4) Physically-consistent Frame Alignment registers the terrain to the simulation coordinate system by aligning the z-axis with gravity and the x-axis with the traversal direction using 3D semantic segmentation.

approaches [43, 21] are unreliable for predicting precise absolute scale on some terrains, as shown in Table IV. This ambiguity is critical because standard scale randomization (e.g., $s \in [0.9, 1.1]$) is insufficient to compensate for arbitrary scale biases on unseen terrains. For instance, if the scale is significantly under-estimated on geometry-constrained terrains like stakes or narrow beams, the effective contact area in the simulator will become infeasibly small. This makes the task physically intractable, preventing policy convergence.

To ensure precise alignment, we calculate a scaling factor by aligning the predicted depth from the feed-forward model to the metric depth from the RGB-D camera. Specifically, we compute the ratio of median depth values derived from the *lower half* of the depth images. This region-of-interest selection effectively focuses on the terrain while mitigating interference from distant background outliers. This alignment step is essential, as it minimizes the sim-to-real gap and ensures that any residual scale deviation falls within the tractable bounds of standard domain randomization.

Coordinate System Alignment: Given the scaled point cloud and reconstructed mesh, our goal is to register the terrain into a *physically consistent world frame*. In this frame, the origin is anchored at the centroid of the start platform p_{start}, while the z-axis and x-axis are aligned with the gravity and the intended traversal direction, respectively.

We assume the physical ground plane is orthogonal to the gravity vector. To estimate this plane robustly, we first utilize a 3D segmentation model [50] to extract points semantically labeled as ground. We apply RANSAC [11] to robustly filter outliers from the segmented ground points, followed by PCA to precisely estimate the surface normal $\mathbf{n}$. The entire scene is then rotated to align $\mathbf{n}$ with the z-axis in simulation.

To align the traversal direction, we identify the centroids of the starting (p_{start}) and ending (p_{end}) platforms derived from the semantic segmentation. The scene is then rotated around the z-axis such that the vector connecting these centroids aligns with the x-axis in simulation. This alignment ensures that the robot's forward velocity command in simulation is geometrically consistent with the physical terrain layout.

D. *Policy Rapid Test-time Training*

Following pre-training, we perform rapid test-time training on specific target terrains, leveraging simulation-ready meshes derived from our reconstruction pipeline. Crucially, we maintain the same Markov Decision Process (MDP) formulation used during pre-training, preserving the same observation space, action space, termination criteria, and reward functions. Building upon the pre-trained policy, we investigate four distinct fine-tuning strategies.

(1) Full Fine-tuning: Updates all parameters of the policy network end-to-end, starting directly from the pre-trained checkpoint.

(2) Adapter Modules: Inserts lightweight adapters after each layer of the depth encoder and MLP. We freeze the original weights and optimize only the adapter modules. Crucially, adapter outputs are zero-initialized to preserve the original feature modulation at the start.

(3) Residual Learning: Adds a parallel network to learn an additive action correction ($\mathbf{a}_{total} = \mathbf{a}_{base} + \mathbf{a}_{res}$). The base policy is frozen, and the residual output layer is zero-initialized so that starts at zero, effectively maintaining the original policy behavior initially.

(4) Last Layer Fine-tuning: Freezes the depth encoder and intermediate MLP layers, restricting updates exclusively to the final linear layer of the actor policy.

IV. Experiments

In this section, we conduct a comprehensive evaluation of the *TTT-Parkour* framework in both simulation and real-world scenarios. We deploy the policy to a suite of extremely challenging terrains designed to push the limits of humanoid locomotion, including wedges, stakes, boxes, trapezoids, and narrow beams. We try to answer the following key questions:

Q1. Necessity & Efficiency: Are both pre-training and rapid test-time training essential for enabling agile locomotion on unseen, extremely challenging terrains, and is the process sufficiently efficient for practical deployment?

Q2. Ablation of Strategies: How do different fine-tuning strategies compare in terms of performance and stability?

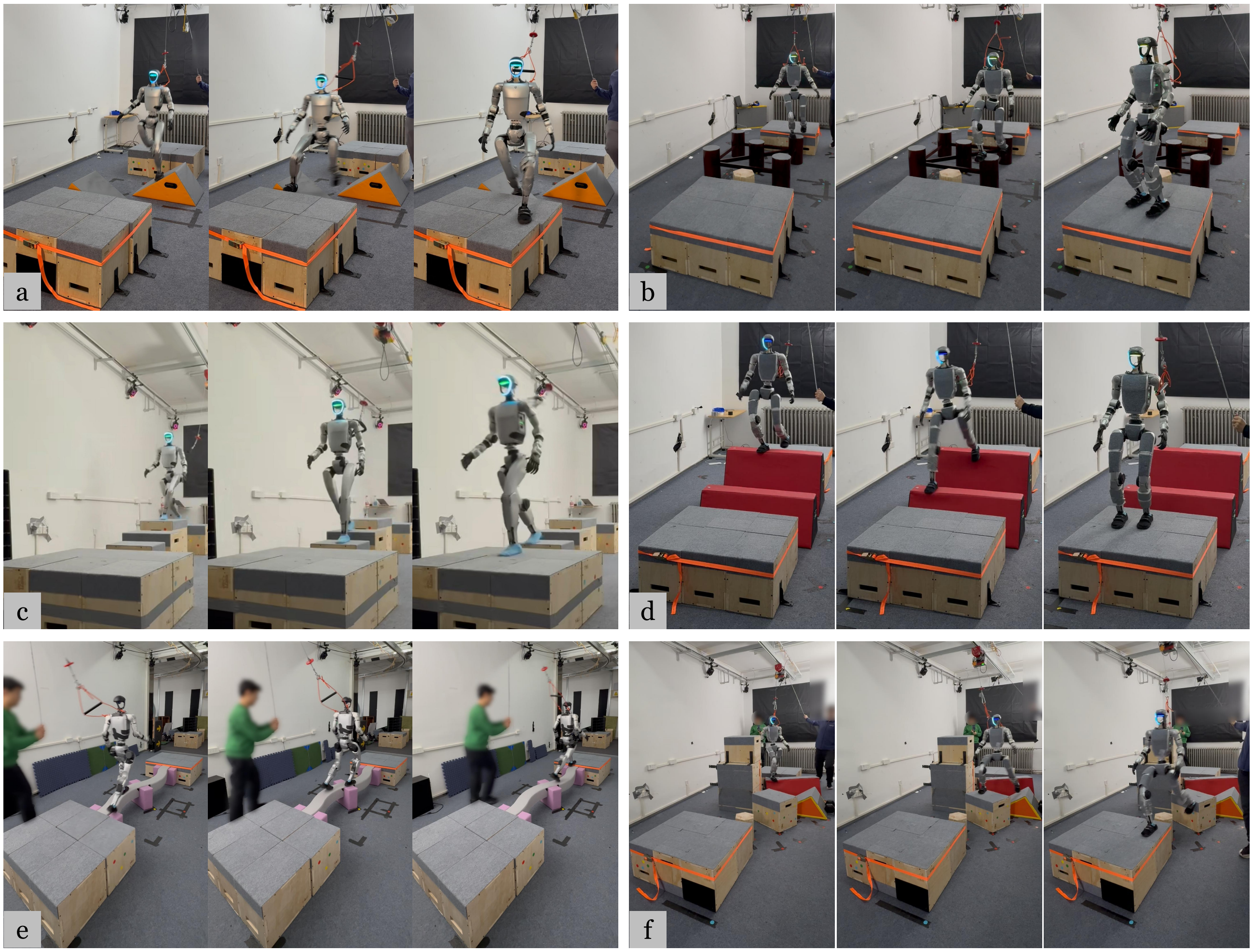


Fig. 4: **Real-world experiments.** The robot successfully traverses extremely challenging terrains, including: (a) Wedges, (b) Stakes, (c) Boxes, (d) Trapezoids, (e) Narrow beam, and (f) Mixed terrain. See videos for more.

Q3. Reconstruction Quality & Speed: How does reconstruction fidelity and speed vary across different data sources, and is RGB-D necessary compared to RGB-only methods?

Q4. Convergence Analysis: What factors influence the sample efficiency and the required number of iterations for the test-time training process?

A. Experiment Configurations

1) Training Setup: We utilize IsaacLab [32] powered by IsaacSim for high-fidelity physics simulation and policy training. All experiments are conducted on a workstation equipped with an NVIDIA RTX 5090 GPU, parallelized with 4096 humanoid robot agents, where each iteration takes less than 4 seconds. We leverage the NVIDIA Warp framework [30] to implement a GPU-accelerated ray-caster depth simulation. The robot employed is the Unitree G1 29Dof. We pre-train the policy for 100,000 iterations, utilizing a history length of $h = 8$ to capture temporal observations. Then we capture and reconstruct the scene for test-time training.

2) Policy Deployment on Real Robot: We perform a zero-shot transfer of the policy trained in simulation to the real robot. The policy is deployed on the robot's onboard NVIDIA Jetson Orin NX computer using ROS2, with the inference loop running at 50 Hz. For perception, we utilize the onboard Intel RealSense D435i camera operating at 60 Hz. The raw depth images are captured at a resolution of 480×270, downsampled to 64×36, and subsequently cropped to a 32×18 patch covering the center-bottom region to focus on the immediate terrain geometry.

3) Tested Terrains: Our experiment focuses on five terrain categories: wedges, stakes, boxes, trapezoids, and narrow beams. Across all experimental setups, the start and end platforms are constructed as large boxes measuring approximately 90 cm in length, 80 cm in width, and 35 cm in height. All intermediate obstacles are arranged to require the robot to traverse them without touching the ground. In the **pre-training** stage, we employ procedural generation to create diverse variations of these five categories in simulation. To ensure robustness, we randomize the position, size, and shape of each obstacle. The simulation environment is organized as a 20×10 grid comprising 5 distinct terrain categories, with each category occupying 4 columns. Within each column, the 10 rows follow a curriculum training strategy [16], where difficulty progressively increases from the first to the last

TABLE I: Simulation success rates across 13 terrains. We compare the **Pre-train** policy (trained on procedural terrains), the **Scratch-1** policy (trained from scratch on the single target terrain), the **TTT-13** policy (fine-tuned simultaneously on all 13 terrains), and the **TTT-1** policy (fine-tuned exclusively on the single target terrain).

Methods / Terrains	**Boxes**	**Wedges**	**Nar.1**	**Nar.2**	**Nar.3**	**Trap.1**	**Trap.2**	**Boston**	**Stake1**	**Stake2**	**Stake3**	**Mix1**	**Mix2**
Pre-train	98.6%	0.1%	81.2%	88.4%	65.6%	0.0%	7.4%	0.0%	4.4%	0.0%	9.9%	0.0%	0.1%
Scratch-1 (25k iters)	0.0%	0.0%	**100.0%**	**100.0%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
TTT-13 (1k iters)	98.7%	**100.0%**	99.9%	**100.0%**	**99.6%**	**100.0%**	99.6%	73.6%	**100.0%**	**100.0%**	**100.0%**	**99.9%**	99.5%
TTT-1 (Converged)	**100.0%**	**100.0%**	**100.0%**	**100.0%**	99.4%	**100.0%**	**100.0%**	**99.9%**	**100.0%**	**100.0%**	**100.0%**	**99.9%**	**100.0%**

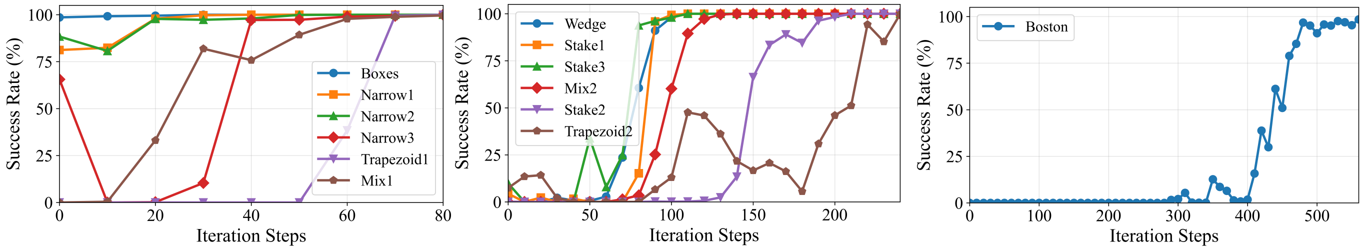


Fig. 5: Success rate progression over test-time training (TTT-1) iterations. The policy rapidly converges to high performance on previously unseen terrains.

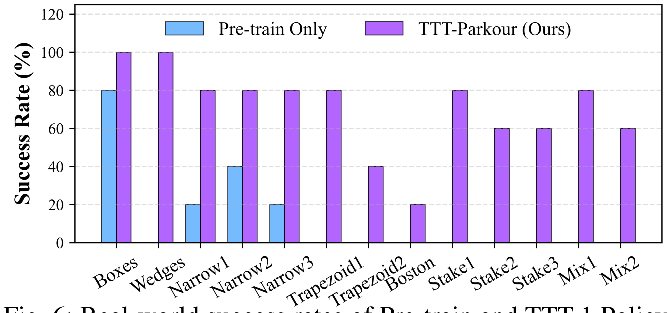


Fig. 6: Real-world success rates of Pre-train and TTT-1 Policy.

row. Specifically, difficulty is modulated by varying geometric parameters: higher difficulty levels correspond to larger gaps and smaller platform dimensions. Representative examples are visualized on the left side of Figure 2. For the **real-world test-time training and deployment** stage, we physically construct 13 distinct testing terrains spanning the aforementioned categories. Each terrain is characterized by randomized spatial arrangements and orientations to strictly challenge the robot's adaptability on unseen geometries. Detailed specifications are provided in the Appendix.

4) Evaluation Metrics: We adopt the success rate as the primary metric for our experiments. In each trial, the robot is initialized on the start platform and commanded with a constant forward linear velocity and zero angular velocity. A trial is recorded as a success if the robot successfully traverses the terrain and reaches the end platform, without touching the ground or falling down. To ensure statistical reliability, we conduct 5 trials for each real-world experiment and 1,000 trials for each simulation experiment.

B. Traversability Analysis

To determine the necessity and efficiency of test-time training, we conduct a comparative analysis against four baselines in simulation. The methods are defined as follows:

(1) Pre-train: The base policy is trained on large-scale procedurally generated terrains and deployed directly without adaptation.

(2) Scratch-1: A policy trained from random initialization directly on a single, specific reconstructed real-world terrain.

(3) TTT-13: A policy fine-tuned simultaneously on all 13 available reconstructed real-world terrains.

(4) TTT-1 (Ours): The proposed framework, which performs rapid test-time fine-tuning on a single, specific reconstructed mesh from a real-world environment.

We first conduct experiments in simulation. As illustrated in Figure 5, the success rate improves rapidly compared to the pre-trained baseline (iteration 0). The test-time training converges to a high success rate on most terrains within 120 iterations, which corresponds to a total adaptation time of approximately 10 minutes (combining the capturing and reconstruction stages). Notably, the *Boston* terrain features a circular arrangement of wedges, requiring complex turning behaviors unlike the linear traversal of other terrains. Thus, it requires more iterations for effective adaptation. Moreover, we compare success rates across different policies in Table I, revealing interesting insights:

(1) The pre-trained policy fails on most test terrains despite their similarity to the procedural training set. This indicates that the policy has limited zero-shot generalizability regarding precise geometric differences in highly challenging scenarios. After test-time training, the fine-tuned policies achieve high success rates across all terrains, highlighting the necessity of test-time adaptation.

(2) Training from scratch fails on most terrains, even after extensive training on a single terrain (25k iterations). Success is limited to only two narrow terrains without wide gaps. This suggests that the curriculum strategy inherent in large-scale pre-training is essential.

(3) TTT-13 slightly underperforms TTT-1. We attribute this to the inherent challenges of multi-task optimization, where gradient interference and reduced sampling density per task hinder convergence.

TABLE II: Comparison of different sources for reconstruction. Time indicates duration from data capture to final mesh.

Methods	**RGB-D**	**LiDAR Scanner**	**iPhone**	**Hand-crafted**
Pros	Balanced quality and efficiency	Professional, best scale accuracy	Most accessible, fast acquisition	No artifacts
Cons	Scale precision slightly lower than LiDAR	Terrain junction artifacts, tedious post-processing	Unstable reconstruction results, flying point artifacts	High manual effort, over-perfect geometry causes sim-to-real gap
Time	**2min 10s**	20min	4min	1h

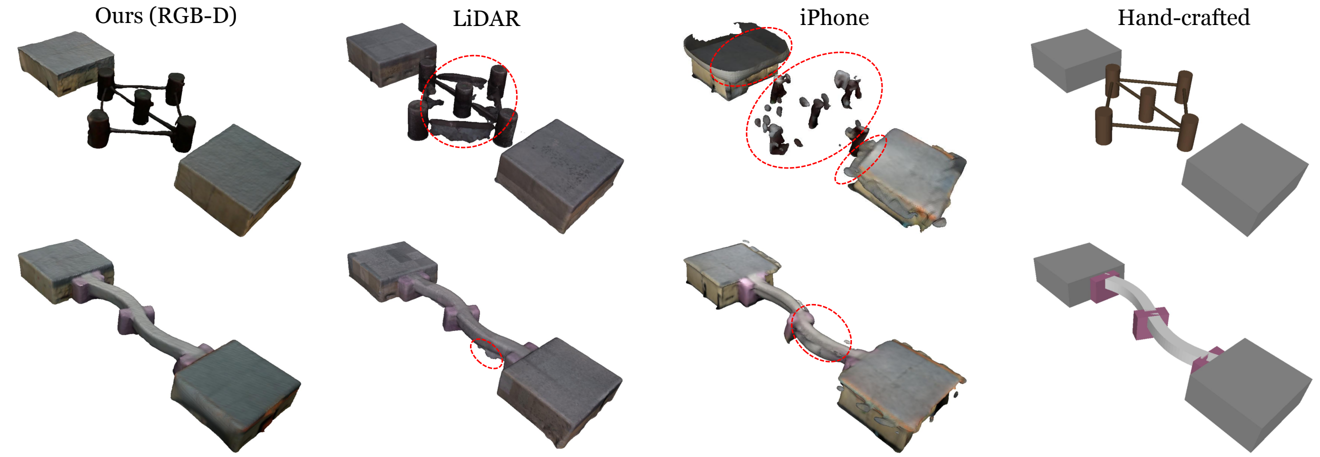


Fig. 7: **Geometry comparison.** Our pipeline with RGB-D input balances quality and efficiency to reconstruct simulation-ready meshes. It maintains realistic geometric fidelity with significantly fewer artifacts compared to LiDAR and iPhone scans, where major artifacts are highlighted by red circles. Top row: Stake1, bottom row: Narrow1.

TABLE III: Success rate in the real world using different reconstruction sources.

Terrains / Methods	RGB-D (Ours)	LiDAR	iPhone	Hand-crafted
Stake1	80%	80%	0%	40%
Narrow1	80%	100%	80%	20%

In the real-world experiments, we deploy the pre-train policy and terrain-specific policy obtained via test-time training **(TTT-1)** on each respective terrain. The success rates are shown in Figure 6. As illustrated, the pre-trained general policy struggles significantly with unseen geometries, failing completely (0% success rate) on most of the challenging obstacles such as Wedges, Trapezoids, and Stakes. In contrast, *TTT-Parkour* demonstrates robust adaptation capabilities, increasing success rates from near zero to more than 60% % on most complex terrains and achieving 100% on Boxes and Wedges. This significant improvement validates that our test-time training pipeline effectively empowers the robot's ability for diverse unseen environments. Some snapshots are shown in Figure 4. Real-world success rates are slightly lower than those in simulation. We attribute this sim-to-real gap to hardware instabilities (e.g., camera noise, actuator dynamics) and environmental mismatches. Unlike the static, rigid simulation, physical terrain elements often wobble or shift during robot interaction. These unmodeled dynamics can lead to failure in real-world tests. Additionally, discrepancies between the reconstructed geometry and the physical terrain still persist, further contributing to the performance gap.

TABLE IV: Comparison of absolute relative scale error. We achieve metric scale fidelity comparable to industrial LiDAR.

Terrains / Methods	RGB-D (Ours)	LiDAR	iPhone	MapAnything	Pi3
Stake1	**0.002**	<u>0.016</u>	0.074	0.383	0.863
Narrow1	<u>0.028</u>	**0.005**	0.056	0.720	1.172

C. Mesh Reconstruction Analysis

As demonstrated in Figure 7 and Table II, our geometry reconstruction pipeline supports diverse input modalities. In the experiments, we utilize a Realsense D435i for RGB-D sensing, a Lixel K1 for LiDAR scanning, an iPhone 16 Pro (via the *3D Scanner App*) for mobile scanning, and hand-crafted meshes generated via Python scripts using the *Trimesh* library, parameterized by manual measurements of the physical terrain.

Among these acquisition methods, RGB-D cameras leverage depth information to recover accurate physical scales, effectively minimizing the sim-to-real gap while producing fewer artifacts than consumer-grade alternatives. It offers the optimal trade-off between reconstruction quality and efficiency.

While LiDAR scanners provide professional-grade scale accuracy, they are expensive and tend to generate artifacts at terrain junctions and fine details. Furthermore, the LiDAR workflow is labor-intensive and time-consuming, necessitating multi-pass scanning, SLAM-based mapping, and extensive post-processing (e.g., denoising, smoothing). Consumer devices like the iPhone, though accessible and fast, yield inconsistent results that are often prone to significant noise, such as flying artifacts. Hand-crafted reconstruction proves to be the least effective for transfer. Although these manually

TABLE V: Comparison of convergence efficiency: Iterations to reach a 97% success rate.

Methods / Terrains	Boxes	Wedges	Nar.1	Nar.2	Nar.3	Trap.1	Trap.2	Boston	Stake1	Stake2	Stake3	Mix1	Mix2
Scratch-1	> 25k	> 25k	17k	17k	> 25k	> 25k	> 25k	> 25k	> 25k	> 25k	> 25k	> 25k	> 25k
TTT-13	0	250	70	70	70	230	**170**	> 1000	350	400	260	160	270
TTT-1 (Ours)	**0**	**100**	**20**	**20**	**40**	**70**	240	**560**	**100**	**200**	**100**	**60**	**120**

TABLE VI: Comparison of different test-time training strategies: Iterations to reach 97% success rate.

Methods / Terrains	Narrow1	Trapezoid1	Mix2
Last Layer	120	**60**	200
Residual	40	70	> 1000
Adapter	40	**60**	260
Full Fine-tuning (Ours)	**20**	70	**120**

designed meshes appear ideal to the human eye, they lack the surface irregularities and geometric noise found in the physical world. This excessive geometric perfection creates a severe domain mismatch, leading to poor sim-to-real performance.

To evaluate the impact of reconstruction quality on sim-to-real transfer, we deployed policies test-time trained on meshes from different sources into the real world, as shown in Table III. While all policies converged to high success rates on their own terrain source in simulation, their real-world performance varied significantly. Hand-crafted terrains consistently failed due to the aforementioned domain mismatch. The iPhone-based reconstruction succeeds on the simpler *Narrow1* terrain but completely fails on *Stake1* due to excessive artifacts in the complex multi-stake environment. RGB-D and LiDAR achieved comparable high success rates. Although LiDAR slightly outperformed RGB-D on *Narrow1* due to superior scale accuracy, its reliance on expensive hardware and high time costs makes it unsuitable for test-time parkour training.

We evaluate metric accuracy by registering point clouds to the hand-crafted one with actual scale to compute the absolute relative scale error. In Table IV, ours matches the precision of industrial LiDAR while remaining the fastest. It significantly outperforms RGB-only baselines like Pi3 (scale-ambiguous) and MapAnything (inaccurate inferred metric scale), validating the necessity of RGB-D input to minimize sim-to-real gap.

D. Analysis of Test-Time Training Strategies

We evaluate various test-time training strategies mentioned in subsection III-D by measuring the number of iterations required to reach a 97% success rate in simulation. As shown in Table VI, **Full Fine-Tuning** achieves the most robust performance across the three tested terrains. In contrast, Residual and Adapter methods require the random initialization of new network parameters, which introduces sensitivity and necessitates additional steps for initial convergence. The Last Layer method restricts the trainable parameter space, thereby limiting the policy's adaptability.

While Parameter-Efficient Fine-Tuning (PEFT) methods (including Last Layer, Residual, and Adapter) are typically designed to trade off slight performance degradation for reduced computational costs, this trade-off proves disadvantageous in our setting. Experimentally, we observe that PEFT methods consistently underperform compared to Full Fine-Tuning. We attribute this performance gap to the significant domain shift between the pre-training and testing terrains. The limited parameter space of PEFT restricts the model's capacity to adapt to such drastic environmental variations, whereas full fine-tuning retains the full expressivity required for this adaptation. Furthermore, since the primary computational bottleneck in our RL pipeline is physical simulation rather than gradient calculation, PEFT provides negligible savings in wall-clock time or resources for each iteration. Thus, given that PEFT degrades performance without offering meaningful efficiency gains, we adopt full fine-tuning as our standard approach.

E. Convergence Analysis

We evaluate the convergence efficiency of three training strategies—training from scratch, multi-terrain Test-Time Training (TTT-13), and terrain-specific TTT (TTT-1) across 13 terrains in simulation. Efficiency is quantified by the number of iterations required to reach a 97% success rate. As detailed in Table V, training from scratch fails to converge within a reasonable time (more than 25k iterations) for all terrains.

Generally, TTT-13 exhibits slower convergence compared to TTT-1. We attribute this to sample dilution: simultaneously optimizing for 13 terrains reduces the effective number of samples available for any specific terrain within a training batch, thereby slowing gradient updates for distinct geometries. However, *Trapezoid2* presents a notable exception where TTT-13 converges faster than TTT-1 (170 vs. 240 iterations). We hypothesize that this terrain shares geometric similarities with the *Narrow Beams*. Thus, Multi-terrain TTT likely benefits from positive transfer, leveraging features learned from the *Narrow Beams* to accelerate adaptation on *Trapezoid2*.

V. Conclusion

In this paper, we introduce *TTT-Parkour*, a framework significantly enhancing the robot's ability to traverse challenging terrains. We establish a two-stage pre-training and test-time training paradigm, alongside a rapid, high-fidelity geometry reconstruction pipeline. Our experiments demonstrate that by performing test-time training on accurately reconstructed terrains, a humanoid robot can master agile and robust parkour on extremely difficult terrains within minutes, including wedges, stakes, boxes, trapezoids, and narrow beams.

Despite these advancements, limitations remain regarding deployment efficiency and task diversity. First, the current 10-minute adaptation process serves as a proof-of-concept. It is still too long for industrial applications and relies on manual terrain capturing. Future work will explore generating local terrain meshes directly from a single robot-centric image, while leveraging improved computational hardware and physical simulators to reduce training time to seconds. Second, our framework relies on static geometric reconstruction, neglecting physical properties such as friction, mass, and compliance. Future work will explore inferring these dynamic parameters to model terrain instability, creating interactive simulations that further minimize the sim-to-real gap.

References

[1] Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

[2] Marco Bagatella, Mert Albaba, Jonas Hübotter, Georg Martius, and Andreas Krause. Test-time offline reinforcement learning on goal-related experience. *arXiv preprint arXiv:2507.18809*, 2025.

[3] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[4] Qingwei Ben, Botian Xu, Kailin Li, Feiyu Jia, Wentao Zhang, Jingping Wang, Jingbo Wang, Dahua Lin, and Jiangmiao Pang. Gallant: Voxel grid-based humanoid locomotion and local-navigation across 3d constrained terrains. *arXiv preprint arXiv:2511.14625*, 2025.

[5] Arunkumar Byravan, Jan Humplik, Leonard Hasenclever, Arthur Brussee, Francesco Nori, Tuomas Haarnoja, Ben Moran, Steven Bohez, Fereshteh Sadeghi, Bojan Vujatovic, et al. Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. *arXiv preprint arXiv:2210.04932*, 2022.

[6] Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, Chenfeng Xu, Quan Vuong, and Ken Goldberg. Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. *arXiv preprint arXiv:2402.19249*, 2024.

[7] Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.

[8] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[9] Helei Duan, Bikram Pandit, Mohitvishnu S Gadde, Bart Van Marum, Jeremy Dao, Chanho Kim, and Alan Fern. Learning vision-based bipedal locomotion for challenging terrain. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 56–62. IEEE, 2024.

[10] Alejandro Escontrela, Justin Kerr, Arthur Allshire, Jonas Frey, Rocky Duan, Carmelo Sferrazza, and Pieter Abbeel. Gaussgym: An open-source real-to-sim framework for learning locomotion from pixels. *arXiv preprint arXiv:2510.15352*, 2025.

[11] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[12] Manuel Yves Galliker. Whole-body humanoid mpc: Realtime physics-based procedural loco-manipulation planning and control. https://github.com/1x-technologies/wb_humanoid_mpc, 2024.

[13] Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning. *arXiv preprint arXiv:2408.14472*, 2024.

[14] Xiaoshen Han, Minghuan Liu, Yilun Chen, Junqiu Yu, Xiaoyang Lyu, Yang Tian, Bolun Wang, Weinan Zhang, and Jiangmiao Pang. Re^3 sim: Generating high-fidelity simulation data via 3d-photorealistic real-to-sim for robotic manipulation. *arXiv preprint arXiv:2502.08645*, 2025.

[15] Junzhe He, Chong Zhang, Fabian Jenelten, Ruben Grandia, Moritz Bächer, and Marco Hutter. Attention-based map encoding for learning generalized legged locomotion. *Science Robotics*, 10(105):eadv3604, 2025.

[16] Nicolas Heess, Dhruva Tb, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[17] Runhan Huang, Haldun Balim, Heng Yang, and Yilun Du. Flexible locomotion learning with diffusion model predictive control. *arXiv preprint arXiv:2510.04234*, 2025.

[18] Runhan Huang, Shaoting Zhu, Yilun Du, and Hang Zhao. Moe-loco: Mixture of experts for multitask locomotion. *arXiv preprint arXiv:2503.08564*, 2025.

[19] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[20] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[21] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction. *arXiv preprint arXiv:2509.13414*, 2025.

[22] Justin Kerr, Letian Fu, Huang Huang, Yahav Avigal, Matthew Tancik, Jeffrey Ichnowski, Angjoo Kanazawa, and Ken Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In *6th annual conference on robot learning*, 2022.

[23] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[24] Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia,

Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real robotic gaussian splatting simulator. *arXiv preprint arXiv:2411.11839*, 2024.

[25] Yue Li, Qi Ma, Runyi Yang, Huapeng Li, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Ender Konukoglu, Theo Gevers, et al. Scenesplat: Gaussian splatting-based scene understanding with vision-language pretraining. *arXiv preprint arXiv:2503.18052*, 2025.

[26] Yue Li, Qi Ma, Runyi Yang, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Theo Gevers, Luc Van Gool, Danda Pani Paudel, et al. Chorus: Multi-teacher pretraining for holistic 3d gaussian scene encoding. *arXiv preprint arXiv:2512.17817*, 2025.

[27] Qiayuan Liao, Takara E Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv preprint arXiv:2508.08241*, 2025.

[28] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning humanoid locomotion with perceptive internal model. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9997–10003. IEEE, 2025.

[29] Haozhe Lou, Yurong Liu, Yike Pan, Yiran Geng, Jianteng Chen, Wenlong Ma, Chenglong Li, Lin Wang, Hengzhen Feng, Lu Shi, et al. Robo-gs: A physics consistent spatial-temporal model for robotic arm with hybrid representation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15379–15386. IEEE, 2025.

[30] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, March 2022. NVIDIA GPU Technology Conference (GTC).

[31] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[32] 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 multimodal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[33] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40(4):1–20, 2021.

[34] M Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhisesh Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6502–6509. IEEE, 2025.

[35] 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.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[37] Haolin Song, Hongbo Zhu, Tao Yu, Yan Liu, Mingqi Yuan, Wengang Zhou, Hua Chen, and Houqiang Li. Gait-adaptive perceptive humanoid locomotion with realtime under-base terrain reconstruction. *arXiv preprint arXiv:2512.07464*, 2025.

[38] Jingkai Sun, Gang Han, Pihai Sun, Wen Zhao, Jiahang Cao, Jiaxu Wang, Yijie Guo, and Qiang Zhang. Dpl: Depth-only perceptive humanoid locomotion via realistic depth synthesis and cross-attention terrain reconstruction. *arXiv preprint arXiv:2510.07152*, 2025.

[39] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

[40] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

[41] Huayi Wang, Zirui Wang, Junli Ren, Qingwei Ben, Tao Huang, Weinan Zhang, and Jiangmiao Pang. Beamdojo: Learning agile humanoid locomotion on sparse footholds. *arXiv preprint arXiv:2502.10363*, 2025.

[42] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[43] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025.

[44] Zirui Wu, Zeren Jiang, Martin R Oswald, and Jie Song. From rays to projections: Better inputs for feed-forward view synthesis. *arXiv preprint arXiv:2601.05116*, 2026.

[45] Zehao Xiao and Cees GM Snoek. Beyond model adaptation at test time: A survey. *arXiv preprint arXiv:2411.03687*, 2024.

[46] Runyi Yang, Zhenxin Zhu, Zhou Jiang, Baijun Ye, Xiaoxue Chen, Yifei Zhang, Yuantao Chen, Jian Zhao, and Hao Zhao. Spectrally pruned gaussian fields with neural compensation. *arXiv preprint arXiv:2405.00676*, 2024.

[47] Baijun Ye, Caiyun Liu, Xiaoyu Ye, Yuantao Chen, Yuhai Wang, Zike Yan, Yongliang Shi, Hao Zhao, and Guyue Zhou. Blending distributed nerfs with tri-stage robust

pose optimization. *arXiv preprint arXiv:2405.02880*, 2024.

[48] Baijun Ye, Minghui Qin, Saining Zhang, Moonjun Gong, Shaoting Zhu, Zebang Shen, Luan Zhang, Lu Zhang, Hao Zhao, and Hang Zhao. Gs-occ3d: Scaling vision-only occupancy reconstruction with gaussian splatting. *arXiv preprint arXiv:2507.19451*, 2025.

[49] Saining Zhang, Baijun Ye, Xiaoxue Chen, Yuantao Chen, Zongzheng Zhang, Cheng Peng, Yongliang Shi, and Hao Zhao. Drone-assisted road gaussian splatting with cross-view uncertainty. *arXiv preprint arXiv:2408.15242*, 2024.

[50] Yujia Zhang, Xiaoyang Wu, Yixing Lao, Chengyao Wang, Zhuotao Tian, Naiyan Wang, and Hengshuang Zhao. Concerto: Joint 2d-3d self-supervised learning emerges spatial representations. *arXiv preprint arXiv:2510.23607*, 2025.

[51] Siheng Zhao, Jiageng Mao, Wei Chow, Zeyu Shangguan, Tianheng Shi, Rong Xue, Yuxi Zheng, Yijia Weng, Yang You, Daniel Seita, et al. Robot learning from any images. In *Conference on Robot Learning*, pages 4226–4245. PMLR, 2025.

[52] Yuhang Zheng, Xiangyu Chen, Yupeng Zheng, Songen Gu, Runyi Yang, Bu Jin, Pengfei Li, Chengliang Zhong, Zengmao Wang, Lina Liu, et al. Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping. *IEEE Robotics and Automation Letters*, 2024.

[53] Allan Zhou, Moo Jin Kim, Lirui Wang, Pete Florence, and Chelsea Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17907–17917, 2023.

[54] Shaoting Zhu, Linzhan Mou, Derun Li, Baijun Ye, Runhan Huang, and Hang Zhao. Vr-robo: A real-to-sim-to-real framework for visual robot navigation and locomotion. *IEEE Robotics and Automation Letters*, 2025.

[55] Shaoting Zhu, Ziwen Zhuang, Mengjie Zhao, Kun-Ying Lee, and Hang Zhao. Hiking in the wild: A scalable perceptive parkour framework for humanoids. *arXiv preprint arXiv:2601.07718*, 2026.

[56] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. *arXiv preprint arXiv:2406.10759*, 2024.

Appendix

A. Reward Formulation

We use the same rewards for both pre-training and test-time training. The reward function is designed to encourage velocity tracking while enforcing physical safety. It consists of four components: task reward, regularization reward, safety reward, and AMP style reward. The detailed definition of each term, along with its corresponding weight and key parameters, is provided in Table VII.

B. Geometric Specifications of Real-World Terrains

We present the key dimensions of the terrains used in our real-world experiments, as shown in Figure 8, Figure 9, and Figure 10. In the figure, the red point denotes the starting platform p_{start}, and the blue point indicates the goal platform p_{end}. Several terrains feature extremely sparse or narrow geometries, posing significant challenges for the robot to secure stable footholds.

TABLE VII: Detailed specification of reward terms, weights, and mathematical formulations.

Reward Term	**Weight**	**Mathematical Formulation**
Task Reward		
Linear Vel Tracking (v_{xy})	2.0	$\exp(-\lVert\mathbf{v}^*_{xy} - \mathbf{v}_{xy}\rVert^2/0.5^2)$
Angular Vel Tracking (ω_z)	0.1	$\exp(-(\omega^*_z - \omega_z)^2/0.5^2)$
Heading Error	-1.0	$\lvert\omega^*_z\rvert$
Don't Wait	-0.5	$\mathbb{I}(v^*_x > 0.3) \cdot (\mathbb{I}(v_x < 0.15) + \mathbb{I}(v_x < 0) + \mathbb{I}(v_x < -0.15))$
Is Alive	3.0	$+1$
Stand Still	-0.3	$(\lVert\mathbf{q} - \mathbf{q}_{default}\rVert_1 - 4.0) \cdot \mathbb{I}(\lVert\mathbf{v}^*\rVert < 0.15) \cdot \mathbb{I}(\lvert\omega^*_z\rvert < 0.15)$
Regularization Reward		
Edge Penetration	-1.0	$\sum_{i=1}^{\lvert\mathcal{P}\rvert} \lVert\mathbf{d}_i\rVert \cdot (\lVert\mathbf{v}_i\rVert + \epsilon)$
Feet Air Time	0.5	$\min_f(t_{phase,f}) \cdot \mathbb{I}(\sum c_f = 1) \cdot \mathbb{I}(\lVert\mathbf{v}^*\rVert > 0.15)$
Feet Slide	-0.4	$\sum_f \lVert\mathbf{v}_{xy,f}\rVert \cdot \mathbb{I}(c_f)$
Joint Deviation (Hip)	-0.5	$\sum_{j\in\text{hips}}(q_j - q_{j,default})^2$
Base Ang Vel (XY) (L_2)	-0.05	$\lVert\omega_{xy}\rVert^2$
Joint Torques (L_2)	-1.5e-7	$\lVert\boldsymbol{\tau}_{legs}\rVert^2$
Joint Acc (L_2)	-1.25e-7	$\lVert\ddot{\mathbf{q}}\rVert^2$
Joint Vel (L_2)	-1.0e-4	$\lVert\dot{\mathbf{q}}\rVert^2$
Action Rate (L_2)	-0.005	$\lVert\mathbf{a}_t - \mathbf{a}_{t-1}\rVert^2$
Flat Orientation	-3.0	$\lVert\mathbf{g}^{proj}_{xy}\rVert^2$
Pelvis Orientation	-3.0	$\lVert\mathbf{g}^{proj,pelvis}_{xy}\rVert^2$
Feet Orientation	-0.4	$\sum_f \lVert\mathbf{g}^{proj}_{xy,f}\rVert \cdot \mathbb{I}(c_f)$
Feet Height Error	-0.1	$\sum_f \sum_p \text{clip}(h_f - h_{terr,p} - 0.035, 0, 0.3) \cdot \mathbb{I}(c_f)$
Feet Distance	1.0	$\exp(-\max(0, 0.12 - \lvert y^R_L - y^R_R\rvert)/0.05) - 1$
Energy Consumption	-5.0e-5	$\sum_j(\tau_j\dot{q}_j/k_j)^2$
Freeze Upper Body	-0.004	$\lVert\mathbf{q}_{upper} - \mathbf{q}^{default}_{upper}\rVert_1$
Safety Reward		
Joint Pos Limits	-1.0	$\sum_j(\max(0, q_j - q_{j,max}) + \max(0, q_{j,min} - q_j))$
Joint Vel Limits	-1.0	$\sum_j \max(0, \lvert\dot{q}_j\rvert - 0.9\dot{q}_{j,max})$
Torque Limits	-0.01	$\sum_j \max(0, \lvert\tau_j\rvert - 0.8\tau_{j,max})^2$
Undesired Contacts	-1.0	$\mathbb{I}(\text{count}(\text{collision}_{body\setminus feet}) > 0)$
AMP Reward		
AMP Style	0.25	$\max\left[0, 1 - 0.25(D(\mathbf{S}_t) - 1)^2\right]$

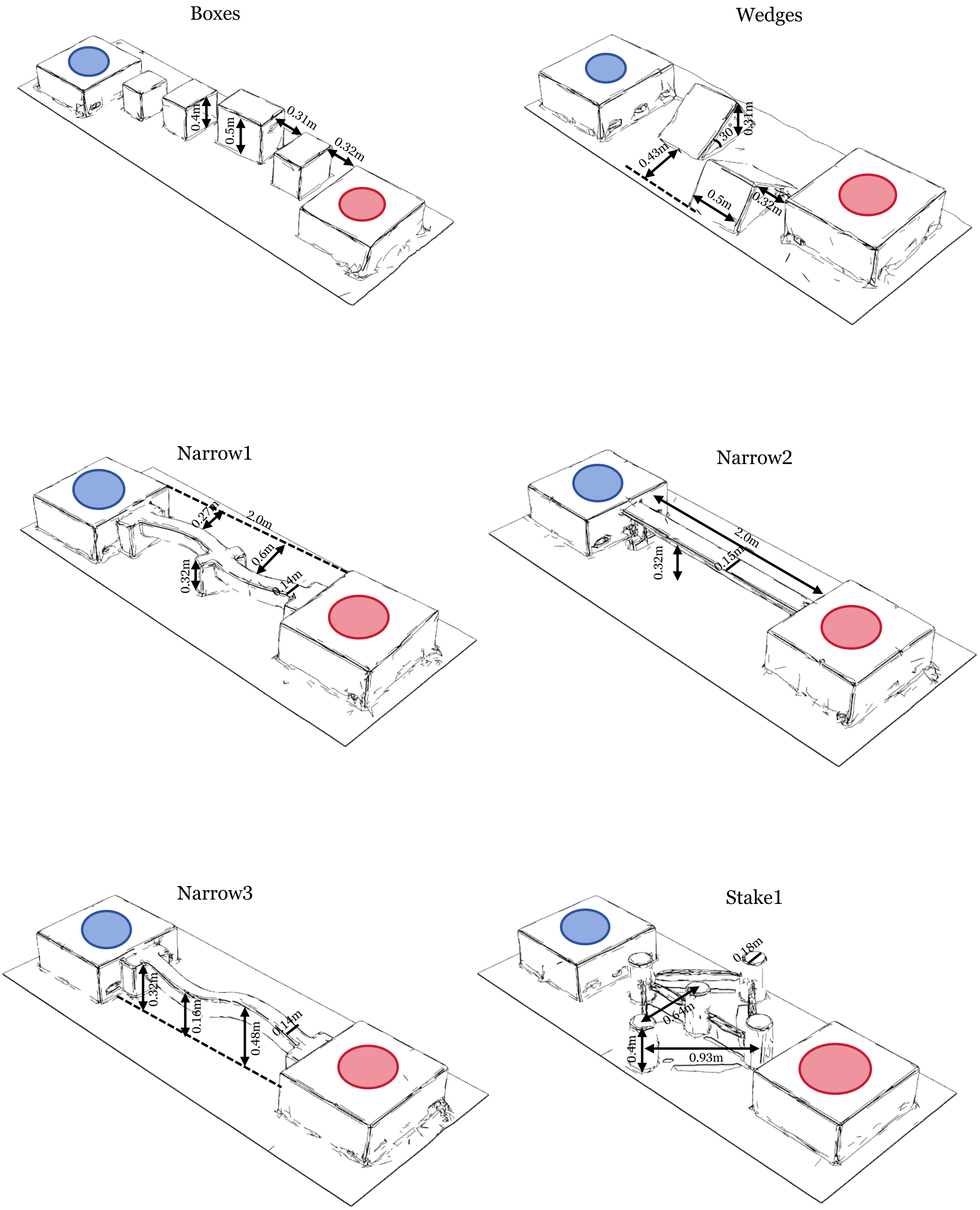


Fig. 8: Detailed dimensions of the real-world terrains.

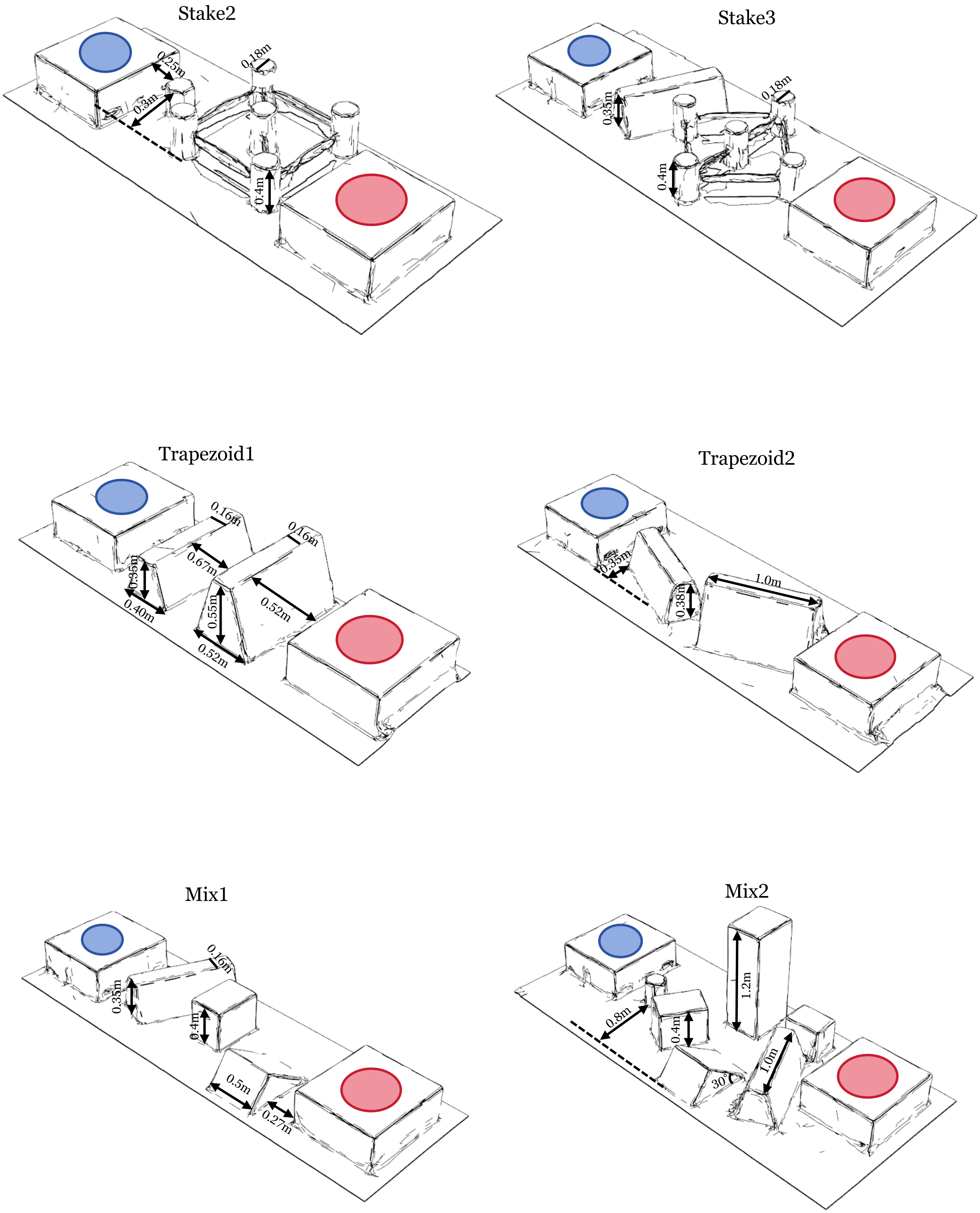


Fig. 9: Detailed dimensions of the real-world terrains.

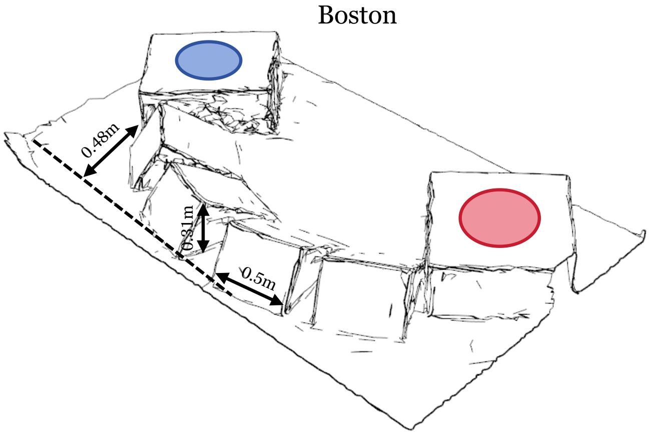


Fig. 10: Detailed dimensions of the real-world terrains.