# Deep Whole-body Parkour

Ziwen Zhuang[12], Shaoting Zhu[12], Mengjie Zhao[1] and Hang Zhao[12†]
[1]IIIS, Tsinghua University, [2]Shanghai Qi Zhi Institute
[†]Corresponding Author

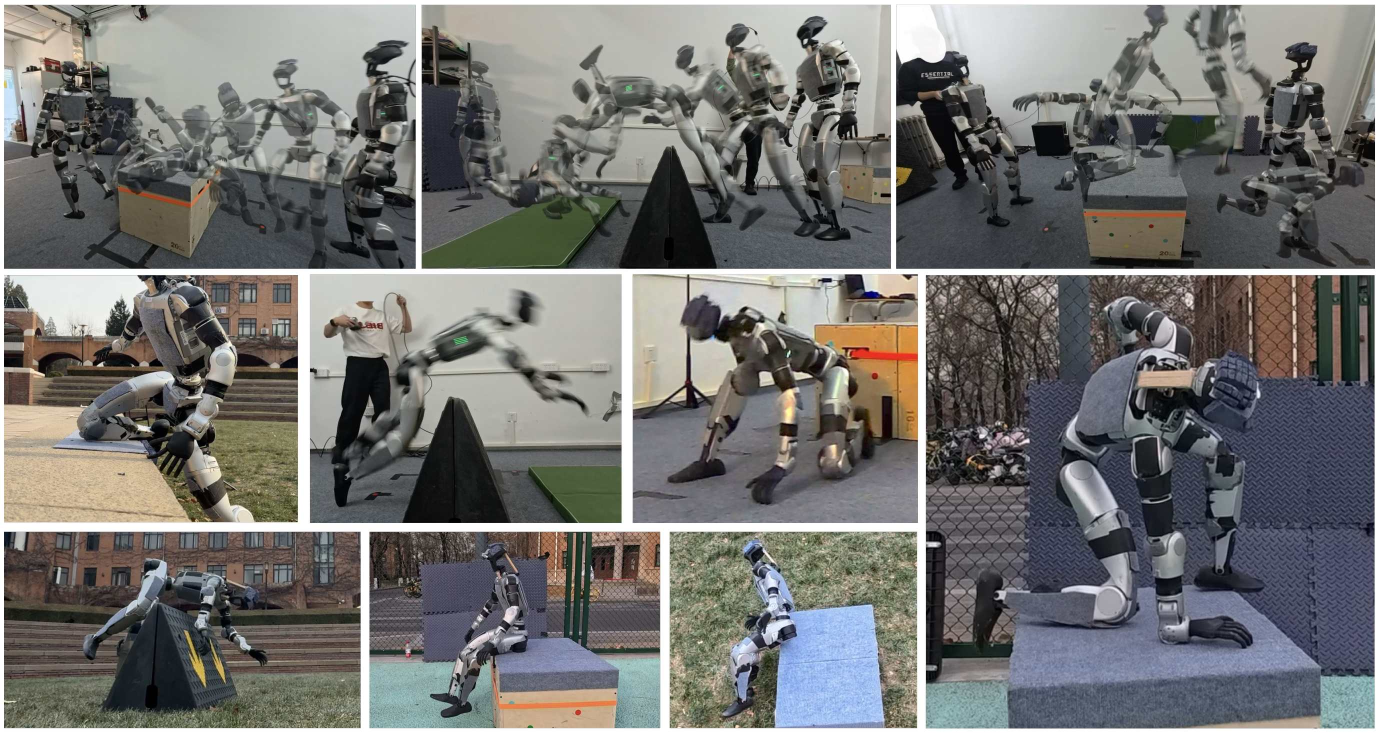

Fig. 1: **Deep Whole-Body Parkour.** Our framework enables a humanoid robot to autonomously traverse challenging obstacles that impose strict geometric constraints on robot odometry. The top row visualizes the dynamic execution of vaulting, diving, and jumping using stroboscopic photography. The bottom rows demonstrate the policy's robustness in diverse indoor and outdoor environments. Crucially, the system utilizes onboard perception to proactively adjust its approach trajectory, ensuring precise foot placement and hand contact for successful whole-body interaction. Please visit the project website and see the open-sourced infrastructure at https://project-instinct.github.io/deep-whole-body-parkour

***Abstract*—Current approaches to humanoid control generally fall into two paradigms: perceptive locomotion, which handles terrain well but is limited to pedal gaits, and general motion tracking, which reproduces complex skills but ignores environmental capabilities. This work unites these paradigms to achieve perceptive general motion control. We present a framework where exteroceptive sensing is integrated into whole-body motion tracking, permitting a humanoid to perform highly dynamic, non-locomotion tasks on uneven terrain. By training a single policy to perform multiple distinct motions across varied terrestrial features, we demonstrate the non-trivial benefit of integrating perception into the control loop. Our results show that this framework enables robust, highly dynamic multi-contact motions—such as vaulting and dive-rolling—on unstructured terrain, significantly expanding the robot's traversability beyond simple walking or running.**

## I. Introduction

Deep Reinforcement Learning (DRL) has fundamentally transformed the landscape of legged robotics, enabling systems to traverse complex environments with unprecedented robustness. By fusing proprioceptive data with exteroceptive observations—such as depth images or height maps—robots can now adapt their gait patterns to unstructured terrain in real-time [2, 25, 5]. This paradigm has rapidly evolved from low-speed navigation on static terrain to high-speed, agile maneuvers. Recent advancements have demonstrated that quadrupeds and humanoids are capable of dynamic parkour, autonomously leaping over gaps and climbing large obstacles [27, 28, 13]. However, despite these aggressive motions, this paradigm remains functionally limited to pedal locomotion. In these frameworks, the robot interacts with the environment exclusively through foot contacts, utilizing the upper body solely for balance rather than for contact-rich interaction. Consequently, while these robots can jump, they lack the capability to execute multi-contact skills—such as vaulting, scrambling, or hand-assisted climbing—that are essential for human-level traversability in constrained spaces.

Conversely, the second paradigm focuses on general motion tracking, a field rooted in computer graphics that prioritizes kinematic over environmental adaptability. In simulation, pre-

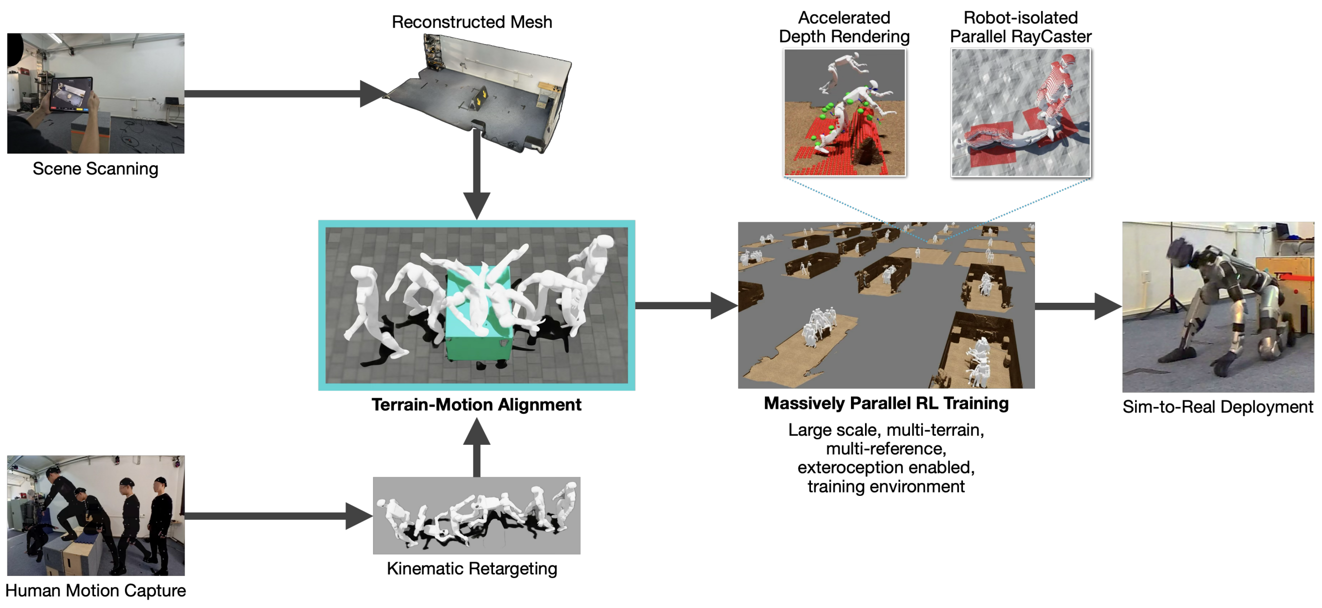
Reconstructed Mesh
Scene Scanning
Accelerated Depth Rendering
Robot-isolated Parallel RayCaster
Terrain-Motion Alignment
Massively Parallel RL Training
Large scale, multi-terrain, multi-reference, exteroception enabled, training environment
Sim-to-Real Deployment
Human Motion Capture
Kinematic Retargeting

Fig. 2: Data-driven whole-body control framework. Real-world environment scans and human demonstrations are processed and aligned to generate feasible motion-terrain pairs. A policy is trained via large-scale reinforcement learning with exteroceptive observations, enabling the robot to replicate agile behaviors in the real world.

vious works such as DeepMimic [18] and AMP [19] demonstrated physically simulated avatars capable of mastering a rich repertoire of skills—including backflips, martial arts, and parkour vaults—by imitating reference motion data. Recently, this paradigm has successfully bridged the sim-to-real gap, with various studies deploying imitation-based policies on physical humanoids to reproduce expressive behaviors in the real world [12, 26, 22, 4]. However, a critical limitation persists: these approaches are predominantly environment-agnostic. They rely solely on tracking a pre-recorded trajectory rather than reacting to the geometry of the world. Consider the specific challenge of vaulting onto a high box platform: a blind tracking policy might perfectly reproduce the kinematics of the vault in open space, but without visual feedback, it cannot adjust its jump timing or hand placement to the specific height and distance of the box. As a result, the robot is destined to either collide with the platform or miss the critical hand contacts entirely.

In this work, we propose to bridge these distinct paradigms by integrating exteroceptive depth perception directly into a whole-body motion tracking framework. We introduce a system where the robot learns not merely to mimic a reference animation but to adapt it based on the visual occupancy of the environment. Crucially, the integration of depth perception yields non-trivial robustness to initial conditions. Unlike blind tracking baselines—which require the robot to be placed at a precise, predefined distance from the obstacle to align the playback trajectory—our visually guided policy operates in a closed loop. This allows the system to tolerate significant variance in the setup: for example, if the robot is initialized at varying distances or angles relative to the platform, the policy leverages visual feedback to autonomously adjust its approach gait—shortening or lengthening its steps—to ensure accurate hand placement and successful vaulting. This capability effectively transforms fragile, fixed-trajectory tracking into robust, spatially-aware capabilities suitable for real-world deployment.

## II. Related Works

*a) Deep Reinforcement Learning for Legged Control:* Deep RL has established a robust standard for locomotion by leveraging proprioceptive history and privileged simulation states [21, 9, 10, 23]. Through teacher-student frameworks or asymmetric actor-critics, policies learn to estimate terrain properties implicitly [8, 10, 9, 17]. While highly stable on continuous irregularities, these ”blind” agents are methodologically limited to reactive reflexes; without exteroception, they cannot anticipate or plan for discrete obstacles.

*b) Perceptive Locomotion:* Integrating exteroception (e.g., depth or elevation maps) enables agents to modulate foot placement for upcoming terrain, facilitating dynamic parkour behaviors [16, 1, 27, 28]. However, these systems typically rely on low-dimensional velocity commands $(v_x, v_y, \omega_z)$, leading to a *task under-specification* problem for humanoids. This low-bandwidth interface cannot disambiguate complex interaction modes—such as vaulting versus jumping—thereby restricting the robot to simple pedal locomotion despite its expressive morphology.

*c) Data-Driven Humanoid Motion Tracking:* Motion tracking approaches, such as DeepMimic [18] and AMP [19], utilize reference motions as a dense task specification, successfully transferring agile skills to physical hardware [14, 22, 6, 7, 12]. However, these frameworks fundamentally operate under a *planar environment assumption*. They treat locomotion purely as kinematic reproduction on flat ground, disregarding the necessity to perceive terrain-dependent information. Consequently, without geometric awareness, they lack the capability to traverse non-planar structures or adapt the reference motion to physical obstacles.

## III. Method

### A. Dataset Curation and Environment Generation

*a) Motivation and Capture:* While large-scale motion datasets such as AMASS [15] provide extensive human kinematic data, and OMOMO [11] introduces object manipulation, they generally lack dynamic, whole-body interactions with large-scale geometry. Agile parkour maneuvers—such as vaulting, climbing, and hurdling—rely critically on precise contacts between the agent and the terrain. To bridge this gap, we curate a custom dataset that strictly couples human dynamics with accurate environmental geometry.

We employ an optical motion capture system to record expert human actors performing parkour maneuvers on physical obstacles. To ensure high-fidelity spatial correspondence between the motion and the terrain, we simultaneously digitize the physical scene using a LiDAR-enabled iPad Pro (via the 3D Scanner App). This process yields a reconstructed mesh that is spatially aligned with the captured motion trajectories (Figure 2).

*b) Motion Retargeting:* The raw human motion is retargeted to the Unitree G1 humanoid robot using the GMR framework [3]. This process utilizes optimization-based kinematic filtering, followed by manual keyframe adjustment, to ensure the resulting trajectories are physically feasible for the robot's morphology. Special attention is paid to enforcing valid contact constraints and eliminating high-frequency capture noise.

*c) Procedural Environment Generation:* To ensure the policy generalizes to diverse environments rather than overfitting to the specific spatial constraints of our capture laboratory, we post-process the scanned meshes for simulation. We isolate the functional geometry (e.g., obstacles, platforms, and rails) by segmenting out surrounding walls, ceilings, and extraneous laboratory context. This yields a set of canonical, context-agnostic obstacle meshes.

Finally, we integrate these assets into NVIDIA Isaac Lab to create a massive-parallel training environment. We treat the retargeted motion and its corresponding obstacle mesh as a single *paired instance*. These pairs are procedurally instantiated across the simulation grid, creating an open-field training setup shown in Figure 2. By stripping the scene of collision group artifacts (like room boundaries) and randomizing the placement of these motion-terrain pairs, we ensure the agent learns to condition its behavior strictly on local obstacle geometry.

*d) Massively Parallel Ray-Caster for Isolated Multi-Agent Training:* To train a perceptive whole-body control policy within a unified framework, high-throughput depth simulation across thousands of parallel environments is essential. While IsaacLab provides GPU-accelerated simulation, clear limitations exist regarding sensor rendering: standard implementations struggle to simultaneously render complex moving articulations and static terrain while strictly isolating parallel environments. Specifically, distinct robots must not perceive "ghost" instances of other robots residing in different environments within the same physics scene.

**Algorithm 1** Massively Parallel Grouped Ray-Casting

```
Require: Ray inputs $\mathbf{R} = \{(\mathbf{o}_i, \mathbf{d}_i, g_i)\}_{i=1}^{N_{rays}}$ (origin, direction, group ID), Mesh instances $\mathcal{M}$, Transforms $\mathbf{T}$, Mesh Group IDs $\mathcal{G}_{mesh}$
Ensure: Depth map $\mathbf{D} \in \mathbb{R}^{N_{rays}}$
    Phase 1: Acceleration Structure Construction (Precompute)
 1: Initialize hash map $\mathcal{H}_{map}$ : Group ID $\rightarrow$ List of Mesh Indices
 2: for each mesh instance $m_j \in \mathcal{M}$ do
 3:     $gid \leftarrow \mathcal{G}_{mesh}[j]$
 4:     Append $j$ to $\mathcal{H}_{map}[gid]$
 5: end for
 6: Define global static group $G_{static} \leftarrow -1$
    Phase 2: Parallel Rendering (GPU Kernel)
 7: for each ray $r_i \in \mathbf{R}$ in parallel do
 8:     $t_{min} \leftarrow \infty$
 9:     $g_{agent} \leftarrow r_i.g_i$
10:                                  ▷ 1. Check Global Static Terrain
11:     for each mesh index $idx$ in $\mathcal{H}_{map}[G_{static}]$ do
12:         $t \leftarrow$ INTERSECT($r_i, \mathcal{M}[idx], \mathbf{T}[idx]$)
13:         if $t < t_{min}$ then $t_{min} \leftarrow t$
14:         end if
15:     end for
16:         ▷ 2. Check Agent-Specific Dynamic Objects (Skip others)
17:     for each mesh index $idx$ in $\mathcal{H}_{map}[g_{agent}]$ do
18:         $t \leftarrow$ INTERSECT($r_i, \mathcal{M}[idx], \mathbf{T}[idx]$)
19:         if $t < t_{min}$ then $t_{min} \leftarrow t$
20:         end if
21:     end for
22:     $\mathbf{D}[i] \leftarrow t_{min}$
23: end for
24: return $\mathbf{D}$
```

To address this, we implement a custom, highly optimized ray-caster utilizing Nvidia Warp. To maximize memory efficiency, we employ mesh instancing; we collect a set of collision mesh prototypes aligned with critical articulations and maintain a global batch of transform matrices for all active instances. We introduce a collision grouping mechanism where static terrain is assigned a universal group ID ($-1$), which is visible to all agents. Conversely, each robot is assigned a unique collision group ID to ensure disjoint perception.

A naive implementation would require every cast ray to iterate through the global table of mesh group IDs to determine visibility, resulting in a complexity linear to the total number of meshes in the scene $O(N)$. With tens of thousands of articulation parts in massively parallel training, this becomes a severe bottleneck. We propose a *Precomputed Grouped Ray-Casting* strategy to accelerate this process, as described in Algorithm 1. We pre-compute a mapping from collision group indices to mesh instance IDs. Consequently, during the ray-marching phase, a ray associated with a specific agent iterates solely over the global static meshes and the specific subset of dynamic meshes belonging to that agent's group. This reduces the search space significantly, resulting in a $10\times$ increase in rendering speed compared to the naive baseline.

## B. Training settings

We train the policy using both proprioception and exteroception, specifically through depth image. Following BeyondMimic [12] and VMP [20], we design the reward terms by tracking the robot local poses in the relative frame and the global pose of the robot root link.

*a) Relative Frame:* By defining the relative frame $T_{\text{rel}}$, we define the reference root transform $T_{\text{ref}}$ as the transform of the root link in the motion reference, and the robot actual root transform $T_{\text{robot}}$. The relative frame $T_{\text{rel}}$ is in between. $T_{\text{rel}}$ has the same x-y coordinate as $T_{\text{robot}}$ but the same z coordinate as $T_{\text{ref}}$. $T_{\text{rel}}$ has the same roll-pitch angle as $T_{\text{ref}}$ but the same yaw angle as $T_{\text{robot}}$.

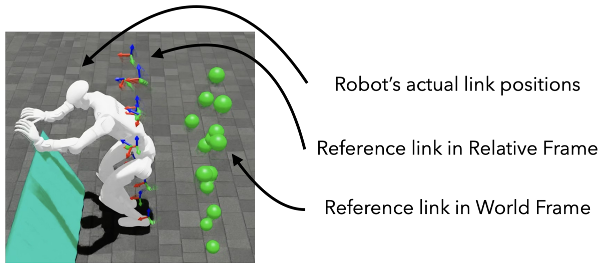
Robot's actual link positions
Reference link in Relative Frame
Reference link in World Frame

Fig. 3: We illustrate the basic concept of relative frame. It has the same x-y and yaw coordinates as the robot base frame and has the same z, roll and pitch coordinate in the reference frame.

Mathematically, we define $T_{\text{rel}}$ as a $4 \times 4$ matrix where

$$T_{\text{rel}} = \begin{bmatrix} R_{\text{rel}} & \mathbf{p}_{\text{rel}} \\ \mathbf{0} & 1 \end{bmatrix}$$

where the translation $\mathbf{p}_{\text{rel}}$ combines the planar position of the robot with the vertical position of the reference,

$$\mathbf{p}_{\text{rel}} = \begin{bmatrix} x_{\text{robot}} \\ y_{\text{robot}} \\ z_{\text{ref}} \end{bmatrix}$$

and the rotation $R_{\text{rel}}$ combines the heading (yaw) of the robot with the tilt (roll and pitch) of the reference

$$R_{\text{rel}} = R_z(\psi_{\text{robot}})\, R_y(\theta_{\text{ref}})\, R_x(\phi_{\text{ref}})$$

More intuitive illustration is shown in Figure 3.

*b) Rewards Definition:* We adopt the policy training using asymmetric PPO. We feed the policy with the synchronized future joint position reference, future joint velocity reference, future root position reference (in reference transform) and future robot rotation reference (in actual robot root transform). Then we feed the policy with one frame of noised depth image and a historical proprioception observation with 8 frames. For critic network, we feed the network with the synchronized future joint position reference, future joint velocity reference, future root position reference (in the actual robot root transform), future robot rotation reference (in the actual robot root transform), actual robot link transforms (in robot's root transform), height-scan to the terrain, and a historical proprioception observation with 8 frames.

Following BeyondMimic [12], we train with simplified tracking rewards as follows:

- Global root position reward based on distance to the reference position.
- Global root rotation reward based on rotation difference to the reference rotation in axis angle.
- Local link position reward based on key links position differences in relative frame.
- Local link rotation reward based on key links rotation differences in relative frame in the magnitude of axis angle.
- Global link linear velocity reward based on the key links linear velocity difference in the world frame.
- Global link angular velocity reward based on the key links angular velocity difference in the world frame.
- Action rate penalty.
- Joint position limit penalty.
- Undesired contact penalty.
- Applied torque limits penalty.

*c) Adaptive Sampling:* We formulate the training environment as a Markov Decision Process (MDP) and employ an adaptive sampling strategy to facilitate curriculum learning. To handle variable-length reference trajectories, we discretize each motion into a sequence of temporal bins with a maximum duration of $t_{\text{bin}} = 1.0s$. This standardization allows us to treat distinct motion segments uniformly, regardless of the total trajectory length. During training rollouts, we track the agent's performance; if a termination or failure occurs, we increment the failure counter for the bin corresponding to the failure timestep. To maintain training stability and prevent abrupt distribution shifts, we apply temporal smoothing to these raw failure counts. This operation converts discrete integer counts into continuous failure scores, effectively diffusing the difficulty signal to adjacent timesteps. Finally, we utilize these smoothed scores to construct a probability distribution over all bins across all motions. The training reset state (which motion and what start time) is then sampled based on these weights, ensuring the agent prioritizes high-failure scenarios. The detailed implementation logic is shown in Algorithm 2.

*d) Early Timeout with Stuck Detection:* However, the randomization terms are not fully applicable to uniform randomization across the entire motion sequence. For example,

**Algorithm 2** Adaptive Sampling with Failure-Based Curriculum

```
Require: Set of reference motions $\mathcal{M} = \{m_1, \dots, m_K\}$
Require: Bin duration $\Delta t_{max} = 1.0s$
Require: Smoothing kernel $\mathcal{K}$ (e.g., Exponential)
 1: Initialize: Discretize each $m_k$ into bins $B_k = \{b_{k,0}, \dots, b_{k,T_k}\}$
 2: Initialize: Failure counts $F_{k,t} \leftarrow 0$ and Sampling Weights $W_{k,t} \leftarrow$ Uniform
 3: for each training iteration do
 4:                                         ▷ 1. Adaptive Selection
 5:     Normalize weights: $P(k,t) \leftarrow \frac{W_{k,t}}{\sum_{k',t'} W_{k',t'}}$
 6:     Sample motion index $k$ and start time $t_{start} \sim P(k,t)$
 7:                                         ▷ 2. Policy Rollout
 8:     Initialize robot state at $S_{t_{start}}$ from motion $m_k$
 9:     Execute policy $\pi$ until completion or failure at $t_{fail}$
10:     if Failure occurred at $t_{fail}$ then
11:         Identify discrete bin index $\hat{t}$ corresponding to $t_{fail}$
12:         $F_{k,\hat{t}} \leftarrow F_{k,\hat{t}} + 1$     ▷ Increment raw failure count
13:                                         ▷ 3. Update Sampling Distribution
14:         for each bin $j$ in motion $m_k$ do
15:             Apply smoothing: $\tilde{F}_{k,j} \leftarrow \sum_i F_{k,i} \cdot \mathcal{K}(j,i)$
16:         end for
17:         Update weights: $W_{k,t} \leftarrow \tilde{F}_{k,t} + \epsilon$     ▷ Add $\epsilon$ to ensure non-zero probability
18:     end if
19: end for
```

the uniform randomization in sampling the robot state from the reference trajectory will place the robot in the middle of some terrain. This generates unsolvable situations for the policy to learn from. To resolve this issue, we truncate the trajectories that have caused the robot to get stuck over a given time limit at the start of the episode.

*e) Network inputs:* Since we design the training framework in one stage, the policy network does not have any odometry-related information. While based on the principle of asymmetric actor critic design, the critic network will have odometry-related information for more accurate value function prediction. Referring to Zhuang and Zhao [26], Chen et al. [4], Ze et al. [24], we train both policy network and critic network with 10 frames of future reference frame. Each frame has 0.1s interval with respect to the previous frame, which sums up to 1.0s future motion expectation.

For policy network, the inputs contain:

- Future joint position reference sequence.
- Future joint velocity reference sequence.
- Future base position reference sequence with respect to the base position of the reference frame.
- Future base rotation reference sequence with respect to the base rotation of the actual robot in simulator.
- Single frame of noised depth image from the robot's camera.
- 8 frames of historical proprioception.

For critic network, the inputs contain:

- Future joint position reference sequence.
- Future joint velocity reference sequence.
- Future base position reference sequence with respect to the base position of the actual robot in simulator.
- Actual key links position in robot's base frame.
- Actual key links rotation (tan-norm) in robot's base frame.
- Height scan data.
- 8 frames of historical proprioception.

*f) Network architecture:* Since the motion tracking task is only spread across a handful number of motions, we use a straight-forward design, with a CNN encoder encoding the depth image and feed to a 3 layer MLP network together with the proprioception history. The detailed parameters are listed in the appendix.

### C. Bridging sim-to-real gap in depth perception

It should be noted that we train our policy using synchronized depth image observations. We deploy the policy using 50Hz depth image using RealSense. However, the builtin filtering algorithm from pyrealsense does not meet the 50Hz requirement on the Nvidia Jetson's CPU. We use the inpainting algorithm from a GPU-based OpenCV implementation. Due to occasionally artifacts pattern in the real world because of reflection, motion blur or stereo algorithm error, we add gaussian noise, patched artifacts to the depth image in simulation similar to Zhuang et al. [27, 28]. As shown in Figure 4, the depth images fed into the network between simulation and real-world are quite similar in terms of pixel pattern.

### D. Deployment

To deploy the policy onboard with depth sensing running at 50Hz, we adopt ONNX as our neural network accelerator. Even though the motions are trained to a single depth-based neural network, we still need to choose the motion reference as input. Thus, we implemented a state machine mechanism to select the motion during deployment.

Also, to generate running logs during the real-world experiments while reducing the impact on the neural network process as low as possible, we run rosbag recording in a separate process. Also, we bind the CPUs separately for running the neural network and rosbag recording process to prevent further CPU scheduling.

Considering the policy is trained with root position reference in motion reference frame, this part of the observation does not depend on the odometry of the robot, neither in simulation nor in the real-world. Thus, we do not need any odometry system providing global position error between the robot and the motion reference. We provide the exact same joint position, joint velocity reference to the policy as it is trained. We provide the relative rotation difference based on the real-time IMU readings of the robot, before which the start heading of the motion reference was aligned to the robot's actual heading. After correcting the heading of the motion reference, as well as the direction of the reference

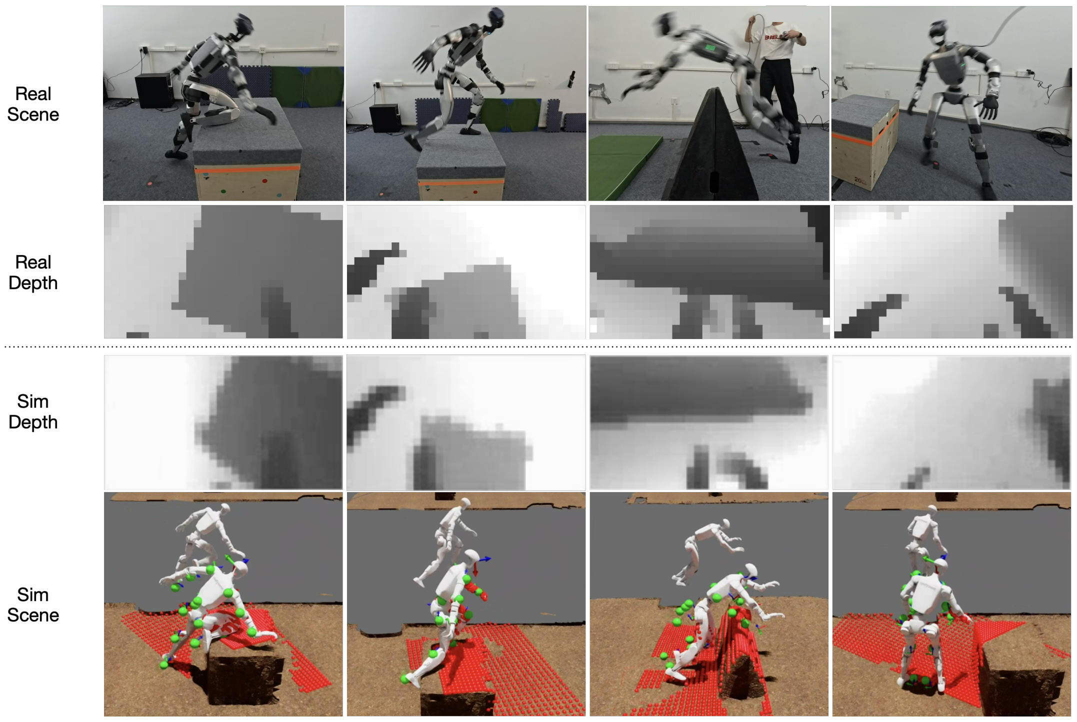

Fig. 4: To bridge the sim-to-real gap in depth visualization, we apply several noise patterns in simulation and applied inpainting algorithm from GPU-based OpenCV implementation.

position sequence, we provide the policy with only the position reference relative to the frame of reference position without any odometry information.

## IV. Experiments

In this section, we aim to raise and answer some questions, addressing the pros and cons of adding exteroception to the highly agile motion tracking system. Also, to test the generalization across multiple terrains and multiple motions in the same terrain, we curate 4 motions across 3 different terrains. One is the triangular barrier formed by two road ramps. One is a big wooden box the size of $0.5m \times 0.6m \times 0.4m$. The last terrain is only the flat terrain, as the complementary experiment to show that the system can still perform non-interactive motions in flat terrain. The 4 motions consist of 'kneel climb', 'roll vault', 'dive roll', and 'jump sit'.

We first run real-world experiments to the effectiveness of our pipeline that can be successfully deployed in the real robot, then we analyze the effect of adding depth vision to the system by proposing the following questions:

1) What benefit does the additional depth vision provide to this agile whole-body control system that interacts with the unstructured terrain?
2) Are there any drawbacks when depth vision is provided as additional information? Is it robust to unseen but trivial distractors in the scene?
3) Throughout the entire training pipeline, what contributes the most to the successful sim-to-real deployment on the real robot without any odometry system?

### A. *Real world experiment*

In the real world experiment, we deploy our entire inference system using ROS2 on a Unitree G1 29-DOF, September 2025 version. The neural network is accelerated using ONNX (CPU version) onboard. We acquire the depth image using an Intel RealSense D435i installed on the head of the Unitree G1 in an individual process separate from the network inference process. We integrate a simple walking process to keep balance when the perceptive parkour policy is not activated. At each test, we control the robot to walk to a rough starting point with no accurate odometry or localization system. We then trigger the motion tracking policy with a specific motion reference trajectory. The system switches back to a walking policy when the motion reference trajectory is exhausted.

We perform real-world tests both indoors and outdoors. With indoor environments, we were able to collect running logs using a network cable from another computer. In outdoor environments, we only connect to the onboard compute without an additional logging system. Please notice that, in outdoor environments, the starting point of the motion is not determined. We only put the robot in front of the obstacle without any further calibration.

### B. *Non-trivial benefit of depth vision*

We investigate how exteroception contributes to the success and generalization of these agile whole-body motions. We uniformly spawn the robot around the initial starting position of the reference motion to test to what extent the policy is able to recover and successfully finish the motion. As shown

TABLE I: Comparison of training variants across four motions. Each motion is evaluated using two metrics (MPJPE$_g$ and MPJPE$_b$).

| **Training Variant** | **Dive Roll** | | **Kneel Climb** | | **Roll Vault** | | **Jump Sit** | |
|---|---|---|---|---|---|---|---|---|
| | MPJPE$_g$ | MPJPE$_b$ | MPJPE$_g$ | MPJPE$_b$ | MPJPE$_g$ | MPJPE$_b$ | MPJPE$_g$ | MPJPE$_b$ |
| *Testing with no position randomization* | | | | | | | | |
| w/o depth (BeyondMimic) | 0.1365 | 0.0462 | 0.1162 | 0.0378 | 0.1109 | 0.0358 | 0.1909 | 0.0341 |
| w/o stuck detection | 0.1367 | 0.0464 | 0.1073 | 0.0394 | 0.1162 | 0.0393 | 0.1170 | 0.0389 |
| local frame reward | 0.1213 | 0.0242 | 0.1145 | 0.0383 | 0.1131 | 0.0332 | 0.1358 | 0.0299 |
| ours | 0.1468 | 0.0459 | 0.1063 | 0.0388 | 0.1123 | 0.0351 | 0.1197 | 0.0362 |
| *Testing with position randomization* | | | | | | | | |
| w/o depth (BeyondMimic) | 0.2267 | 0.0477 | 0.1822 | 0.0405 | 0.1969 | 0.0385 | 0.2693 | 0.0377 |
| w/o stuck detection | 0.1661 | 0.0456 | 0.1449 | 0.0412 | 0.1461 | 0.0316 | 0.1322 | 0.0354 |
| local frame reward | 0.1892 | 0.0389 | 0.1477 | 0.0402 | 0.1453 | 0.0311 | 0.1312 | 0.0301 |
| ours | 0.1649 | 0.0477 | 0.1218 | 0.0399 | 0.1421 | 0.0371 | 0.1446 | 0.0382 |

in Figure 7 we visualize the convergence of the robots in a batch, which start at different positions. As time goes on, all robots converge to the target obstacle and successfully finish the motion tracking task. Even though all trainings randomize the starting position at $0.3m \times 0.3m$ horizontally around the reference motion. We test the motion convergence behavior with the range of $0.5m \times 0.5m$ horizontally around the reference starting point.

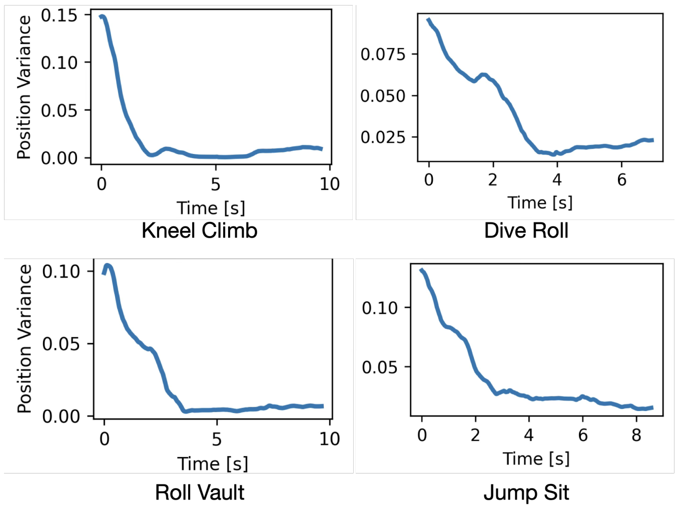

Fig. 5: We show the $x - y$ position variance in a single batch of motion reference example to illustrate the emergence of positional correction ability when introducing depth vision to the end-to-end motion tracking system.

We then plot the position variance in x-y coordinate to further verify this observation in Figure 5. The position variance drastically converges to a really low value. When the motion reaches its final stage, some of the position variances increase due to the common visual features from the depth camera.

To further test the generalization ability of this visual-guided motion tracking pipeline, we perform a more aggressive out-of-distribution tests, even though the randomization during training is only from $-0.15m$ to $+0.15m$ in $x$ and $y$ coordinate. We plot the success rate headmap from $-0.6m$ to $+0.6m$ around the initial motion reference starting point. This forms a $1.2m \times 1.2m$ range of initialization. As shown in Figure 6, the success rate of the motion tracking system is still 100%. It only drops at the boundary of these experiments.

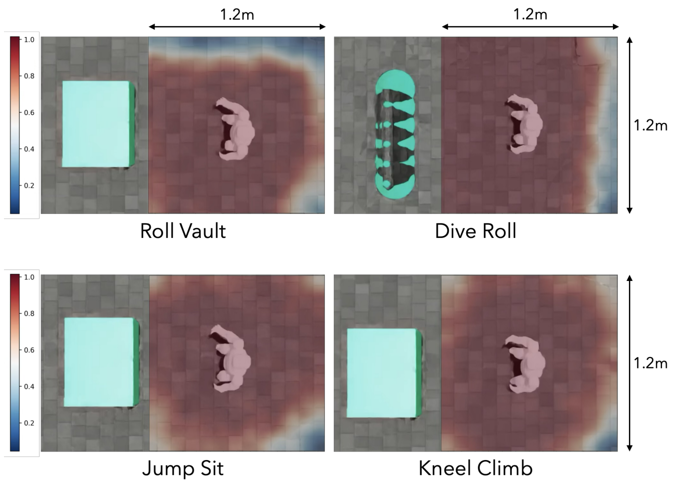

Fig. 6: We do a grid search around the starting position of the motion reference frame. We plot the headmap of the success rate of each motion at a $1.2m \times 1.2m$ space. Red suggests the 100% success rate, while dark blue suggests the 0% success rate.

## C. Vision robustness

In this section, we aim to study the robustness of this additional depth information in the motion tracking system. We build several scenes in the simulator that have never been used during training. They are the out-of-distribution cases for the trained motion tracking network. Considering the reference motions are still the same, the added entities in the scene should block the robot's dynamics as little as possible.

As shown in Figure 8, we construct different objects in the scene as distractors and make sure the robot sees the additional objects. Some of the distractors do disrupt the dynamics of the entire motion, e.g, the bottom left example. We tested different

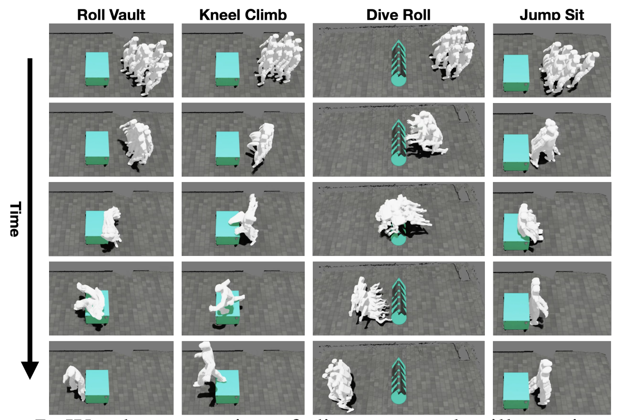

Fig. 7: We show a series of direct example, illustrating the positional convergence when introducing depth vision in this end-to-end motion tracking system. As shown in this figure, the position converges automatically even before the scene interaction. Otherwise, for difficult motions like roll-vaulting and kneel-climbing, misplace at the scale of $0.4m$ will lead to catastrophic failure of the scene-interaction task.

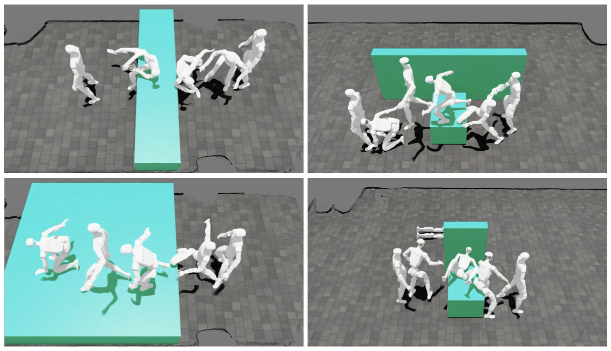
Fig. 8: Here we provide examples of adding distractors in the scene to test the robustness of adding unseen objects in the middle of the motion trajectory. We define "wide distractor" (top-left), "plane distractor" (bottom-left) and "wall distractor" (right). They present different visual sensor information while not disrupting the original motion trajectory too much.

variants of distractors and observed no drop in the success rate when running the same motion tracking system.

TABLE II: Motion Tracking Evaluation: MPJPE-Global MPJPE$_g$ and MPJPE-Base MPJPE$_b$ across different distractor conditions.

| **Distractor Condition** | **Kneel Climb** | | **Roll Vault** | |
|---|---|---|---|---|
| | MPJPE$_g$ | MPJPE$_b$ | MPJPE$_g$ | MPJPE$_b$ |
| No Distractor | 0.1022 | 0.0390 | 0.1159 | 0.0349 |
| Wide Distractor | 0.4848 | 0.0405 | 0.1543 | 0.0373 |
| Plane Distractor | 0.4261 | 0.0417 | 0.5598 | 0.0809 |
| Wall Distractor | 0.1603 | 0.0401 | 0.1157 | 0.0372 |

We focus on 2 motions that are suitable to add distractors while not significantly blocking the dynamics of the motion trajectory. As shown in Table II, we test the MPJPE in both the base frame and the world frame compared with the original terrain used in training. In total, there are 4 variants for both roll-vaulting and kneel-climbing. For each variant and the original training scene, we run a batch of 100 robots using the same policy network in the simulator at once. To make a fair comparison, we remove randomization when initializing the robot in the scene. As shown in Table II, only the plane distractor significantly increase the MPJPE metric, since the long platform changes the potential motion dynamics compared to the original reference motion. For the wide distractor, it also leads to larger MPJPE. By analyzing the motion, the wide beam disrupts the localization ability of the motion tracking policy. Thus, the MPJPE metric increases.

### D. *Ablation on the training recipe*

In this section, we run ablation studies on several critical components of the entire training framework. There are several critical components in our training framework:

1) Training with depth input.
2) Randomizing the initialization strategy with stuck detection, which prevents physically impossible rollouts.
3) Defining local tracking rewards in the relative frame instead of the robot's base frame.

As shown in Table I, training motion tracking without exteroception is not robust to initial position perturbation. The starting position has to be exact, or it may face catastrophic failure during scene interaction. When trained only with local frame reward instead of the reward in the relative frame, the policy tends to track more accurate locally, but performs worse than our pipeline globally in interaction intensive cases. Also, we show the non-trivial benefit of applying stuck detection mechanism to prevent large portion of rollout steps becoming useless data samples. Training without stuck detection leads to slightly higher MPJPE error, while still successfully finished the entire motion tracking trajectory.

## V. Conclusion

In this work, we present a generalizable paradigm for humanoids to learn agile, physics-based interactions directly from human demonstrations and onboard sensing. We advance beyond standard motion tracking by unifying it with perceptive control into a **single scalable training framework**. Our analysis highlights the non-trivial benefits of depth vision for robustness against environmental distractors and validates the system through successful deployment in diverse indoor and outdoor real-world scenarios.

Furthermore, by enabling fully end-to-end training with exteroception, this framework provides the critical infrastructure required for scaling up the training system of humanoid whole-body control. This end-to-end approach unlocks the potential to train extensive libraries of motion skills with intensive scene interaction. While challenges regarding data volume and autonomous skill selection remain, they outline a clear path for future research. Ultimately, this work establishes a baseline for developing general-purpose humanoid controllers capable of mastering contact-rich environments in a fully learned, end-to-end manner.

## Acknowledgments

This work is accomplished with the help of Xiangting Meng, Siqiao Huang, Yizhuo Gao, and Tairan He.

## References

[1] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged Locomotion in Challenging Terrains using Egocentric Vision. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=Re3NjSwf0WF.

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged Locomotion in Challenging Terrains using Egocentric Vision. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 403–415. PMLR, December 2023. URL https://proceedings.mlr.press/v205/agarwal23a.html.

[3] Joao Pedro Araujo, Yanjie Ze, Pei Xu, Jiajun Wu, and C. Karen Liu. Retargeting Matters: General Motion Retargeting for Humanoid Motion Tracking. *arXiv preprint arXiv:2510.02252*, 2025.

[4] Zixuan Chen, Mazeyu Ji, Xuxin Cheng, Xuanbin Peng, Xue Bin Peng, and Xiaolong Wang. GMT: General Motion Tracking for Humanoid Whole-Body Control. *arXiv:2506.14770*, 2025.

[5] Helei Duan, Ashish Malik, Jeremy Dao, Aseem Saxena, Kevin Green, Jonah Siekmann, Alan Fern, and Jonathan Hurst. Sim-to-Real Learning of Footstep-Constrained Bipedal Dynamic Walking, 2022. URL https://arxiv.org/abs/2203.07589. _eprint: 2203.07589.

[6] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv:2403.04436*, 2024.

[7] Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Changliu Liu, Guanya Shi, Xiaolong Wang, Linxi Fan, and Yuke Zhu. HOVER: Versatile Neural Whole-Body Controller for Humanoid Robots. *arXiv preprint arXiv:2410.21229*, 2024.

[8] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 2019.

[9] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

[10] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 2020.

[11] Jiaman Li, Jiajun Wu, and C. Karen Liu. Object Motion Guided Human Motion Synthesis. *ACM Trans. Graph.*, 42(6), December 2023. ISSN 0730-0301. doi: 10.1145/3618333. URL https://doi.org/10.1145/3618333. Place: New York, NY, USA Publisher: Association for Computing Machinery.

[12] 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, 2025. URL https://arxiv.org/abs/2508.08241. _eprint: 2508.08241.

[13] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning Humanoid Locomotion with Perceptive Internal Model, 2024. URL https://arxiv.org/abs/2411.14386. _eprint: 2411.14386.

[14] Zhengyi Luo, Jinkun Cao, Alexander W. Winkler, Kris Kitani, and Weipeng Xu. Perpetual Humanoid Control for Real-time Simulated Avatars. In *International Conference on Computer Vision (ICCV)*, 2023.

[15] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision*, pages 5442–5451, October 2019.

[16] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022. doi: 10.1126/scirobotics.abk2822. URL https://www.science.org/doi/abs/10.1126/scirobotics.abk2822. _eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abk2822.

[17] I. Made Aswin Nahrendra, Byeongho Yu, and Hyun Myung. DreamWaQ: Learning Robust Quadrupedal Locomotion With Implicit Terrain Imagination via Deep Reinforcement Learning, March 2023. URL http://arxiv.org/abs/2301.10602. arXiv:2301.10602 [cs].

[18] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311. Place: New York, NY, USA Publisher: ACM.

[19] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, 40(4), July 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459670. URL https://doi.org/10.1145/3450626.3459670. Place: New York, NY, USA Publisher: Association for Computing Machinery.

[20] Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. VMP: Versatile Motion Priors for Robustly Tracking Motion on Physical Characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '24, pages 1–11, Montreal, Quebec, Canada, 2024. Eurographics Association. doi: 10.1111/cgf.15175. ISSN: 1467-8659.

[21] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent

Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. 2018. URL https://arxiv.org/pdf/1804.10332.pdf.

[22] Yufei Xue, Wentao Dong, Minghuan Liu, Weinan Zhang, and Jiangmiao Pang. A Unified and General Humanoid Whole-Body Controller for Fine-Grained Locomotion. In *Robotics: Science and Systems (RSS)*, 2025.

[23] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data Efficient Reinforcement Learning for Legged Robots. *arXiv:1907.03613 [cs]*, October 2019. URL http://arxiv.org/abs/1907.03613. arXiv: 1907.03613.

[24] Yanjie Ze, Zixuan Chen, João Pedro Araújo, Zi-ang Cao, Xue Bin Peng, Jiajun Wu, and C. Karen Liu. TWIST: Teleoperated Whole-Body Imitation System, May 2025. URL http://arxiv.org/abs/2505.02833. arXiv:2505.02833 [cs].

[25] Chong Zhang, Nikita Rudin, David Hoeller, and Marco Hutter. Learning Agile Locomotion on Risky Terrains. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11864–11871, 2024. doi: 10.1109/IROS58592.2024.10801909.

[26] Ziwen Zhuang and Hang Zhao. Embrace Contacts: humanoid shadowing with full body ground contacts. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 3720–3735. PMLR, September 2025. URL https://proceedings.mlr.press/v305/zhuang25b.html.

[27] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher G. Atkeson, Sören Schwertfeger, Chelsea Finn, and Hang Zhao. Robot Parkour Learning. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 73–92. PMLR, November 2023. URL https://proceedings.mlr.press/v229/zhuang23a.html.

[28] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid Parkour Learning. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 1975–1991. PMLR, November 2024. URL https://proceedings.mlr.press/v270/zhuang25a.html.