RPL: Learning Robust Humanoid Perceptive Locomotion on Challenging Terrains

Yuanhang Zhang[1,2] Younggyo Seo[1] Juyue Chen[1] Yifu Yuan[2]
Koushil Sreenath[1,4] Pieter Abbeel[†1,4] Carmelo Sferrazza[†1] Karen Liu[†1,3] Rocky Duan[†1] Guanya Shi[†1,2]
[1]Amazon FAR [2]Carnegie Mellon University [3]Stanford University [4]UC Berkeley [†]Amazon FAR team co-lead
Page: https://rpl-humanoid.github.io/

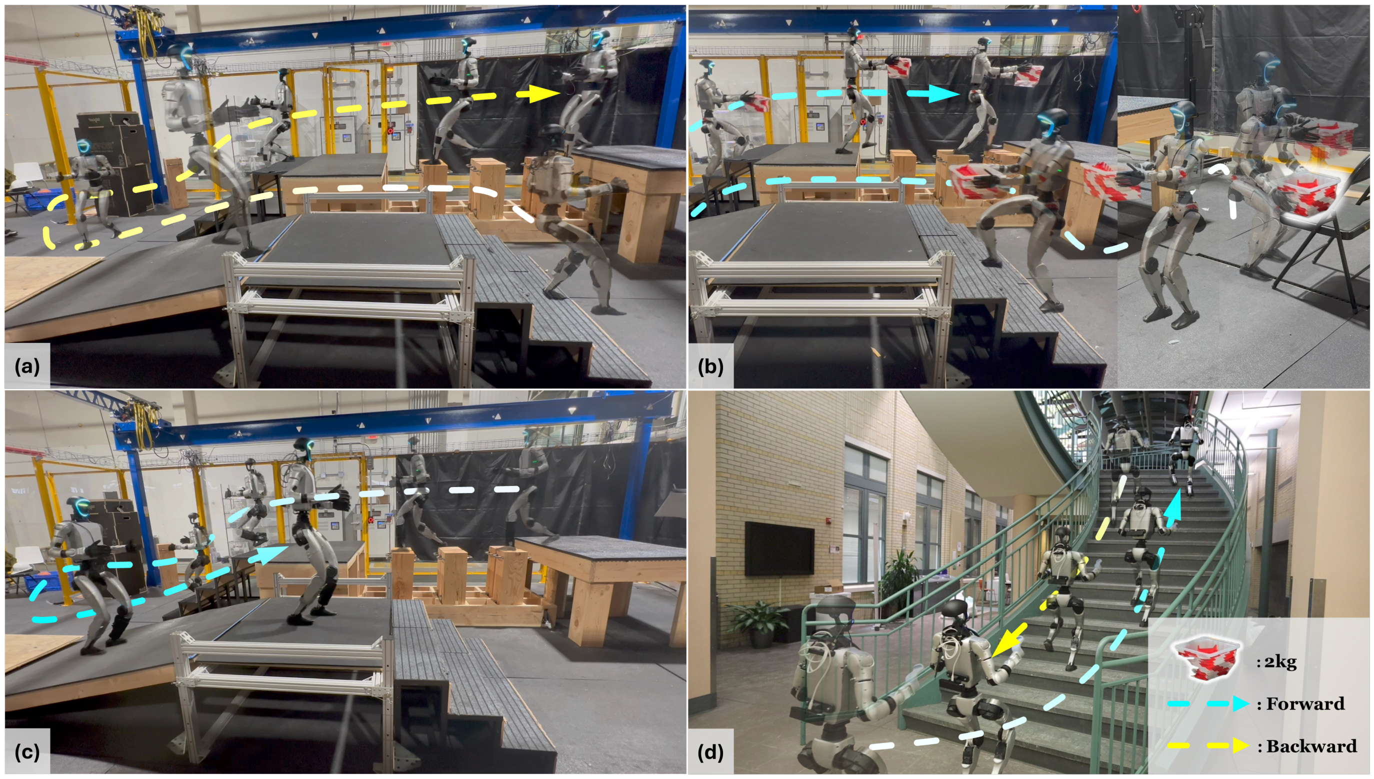

Fig. 1. RPL enables long-horizon bidirectional locomotion and remain robust with payloads on Unitree G1 humanoid robot with a depth-based transformer policy. Terrains include 20° slopes, staircases with different step lengths (22 cm, 25 cm, 30 cm), and 25 cm×25 cm stepping stones with 60 cm gaps.

***Abstract*—Humanoid perceptive locomotion has made significant progress and shows great promise, yet achieving robust multi-directional locomotion on complex terrains remains underexplored. To tackle this challenge, we propose RPL, a two-stage training framework that enables multi-directional locomotion on challenging terrains, and remains robust with payloads. RPL first trains terrain-specific expert policies with privileged height map observations to master decoupled locomotion and manipulation skills across different terrains, and then distills them into a transformer policy that leverages multiple depth cameras to cover a wide range of views. During distillation, we introduce two techniques to robustify multi-directional locomotion, depth feature scaling based on velocity commands and random side masking, which are critical for asymmetric depth observations and unseen widths of terrains. For scalable depth distillation, we develop an efficient multi-depth system that ray-casts against both dynamic robot meshes and static terrain meshes in massively parallel environments, achieving a 5× speedup over the depth rendering pipelines in existing simulators while modeling realistic sensor latency, noise, and dropout. Extensive real-world experiments demonstrate robust multi-directional locomotion with payloads (2kg) across challenging terrains, including 20° slopes, staircases with different step lengths (22 cm, 25 cm, 30 cm), and 25 cm×25 cm stepping stones separated by 60 cm gaps.**

I. Introduction

Humanoid robots hold great potential to execute daily human tasks. Recent years have demonstrated their agile motion tracking capabilities, enabling human-level and even beyond-human whole-body motions [41, 2, 15, 5, 32, 8, 12, 13, 19, 28, 4, 9, 39]. However, beyond expressive whole-body control, a defining capability of humanoid robots—beyond traditional wheeled mobile manipulators—lies in their ability to leverage legged mobility to robustly traverse diverse terrains, while performing meaningful tasks such as carrying payloads.

Existing humanoid perceptive locomotion methods [37, 6, 43, 21, 3, 14, 34, 35, 38, 40, 11] have demonstrated impressive performance across a variety of challenging terrains. However, these approaches typically rely on a single forward camera and assume idealized upper-body conditions, which become brittle when extended to multi-directional locomotion or when upper-body motions introduce dynamic self-occlusions, posing fundamental challenges to perception robustness. In addition, while mapping-based methods [25, 38, 21, 3, 37] may support bidirectional locomotion on challenging terrains such as stepping stones, they rely on explicit state estimation, which is often noisy and complicates system-level optimization. Therefore, end-to-end perceptive control under asymmetric (i.e., different cameras see different types of terrains) visual inputs remains underexplored.

To overcome these limitations, we propose `RPL`, a two-stage learning framework for **R**obust humanoid **P**erceptive **L**ocomotion on challenging terrains. `RPL` first trains terrain-specific expert policies with end-effector force perturbation and privileged height-map as observation to acquire decoupled locomotion and manipulation skills under end-effector force perturbations across diverse terrains. These experts are then distilled into a unified transformer policy that leverages multiple depth cameras enabling robust multi-directional locomotion. To support scalable and realistic depth rendering during distillation, we develop an efficient multi-depth rendering system that ray-casts against both dynamic robot meshes and static terrain meshes in massively parallel environments, while modeling realistic depth sensor latency, Gaussian noise, and dropout to improve sim-to-real robustness. Furthermore, to robustify multi-directional locomotion across terrains with unseen widths, we introduce two complementary techniques during distillation: (1) *depth feature scaling based on velocity commands* (DFSV), which adaptively modulates perception features based on commanded velocity, thereby reducing distribution shift between training and deployment for asymmetric visual inputs from multiple cameras; (2) *random side masking* (RSM) which randomly masks terrain regions on both sides of the depth images with varying widths to improve robustness to unseen terrain widths.

Our experiments show that our depth-based policy enables the humanoid to perform stable back-and-forth locomotion on challenging terrains, including unseen widths of slopes and stairs, as well as stepping stones, while remaining robust with carried payloads. In summary, our key contributions are:

- We propose `RPL`, a two-stage framework that distills terrain-specific expert policies into a unified depth policy for robust, multi-directional humanoid locomotion with payloads on diverse challenging terrains.
- We develop an efficient multi-depth rendering system that captures both dynamic robot meshes (to address self-occlusion during loco-manipulation) and static terrain meshes, achieving an 5× speedup over existing simulators while modeling realistic latency, noise, and dropout.
- We introduce depth feature scaling based on velocity commands (DFSV) and random side masking (RSM) during distillation to enable robust multi-directional locomotion across terrains with unseen widths.
- We validate long-horizon real-world robustness on a humanoid robot, demonstrating stable bidirectional locomotion across challenging terrains under whole-body motions and payload-induced disturbances.

II. Related Works

A. Perceptive Locomotion in Legged Robots

Perceptive locomotion for legged robots has made great progress in recent years [25, 37, 6, 43, 21, 14, 34, 35, 38, 40, 11, 3, 42, 21, 1]. Notably, quadrupeds can sometimes traverse challenging terrains (e.g., stairs and curbs) even with limited perception or blind locomotion [17], benefiting from inherently stable support polygons, whereas humanoids require more precise foot placement and are thus more sensitive to perception errors and partial observations. In terms of the perception input, there are mainly two paradigms: (1) LiDAR-based mapping methods [25, 37, 21, 3, 14, 11, 38], leverage point clouds or reconstructed elevation maps as compact terrain representations for perceptive locomotion. However, these pipelines rely on accurate calibration and robust online mapping/state estimation to maintain sim-to-real consistency, which could also introduce additional delays that affect policy performance. In contrast, (2) end-to-end vision-based methods, typically using depth images [6, 34, 35, 1, 43, 42, 40], directly map raw depth images to actions and reduce dependence on explicit mapping or state estimation.

Despite promising depth-based sim-to-real results, most approaches adopt a single forward-facing camera and primarily demonstrate forward locomotion, leaving bidirectional or omnidirectional locomotion under asymmetric visual inputs underexplored. To enable multi-directional perceptive locomotion, some recent work [33] uses a single downward-facing camera to achieve omnidirectional traversal on structured terrains such as stairs and gaps; however, the narrow field of view provides limited lookahead and makes it difficult to handle sparse and discontinuous footholds (e.g., stepping stones). Other approaches [10] place multiple cameras around the robot to improve perceptual coverage; however, they are typically evaluated on a single terrain family (e.g., stairs) and do not consider the more general setting where different cameras simultaneously observe distinct terrain structures. Meanwhile, depth rendering in existing simulators remains computationally expensive, making scalable multi-view depth simulation challenging. Recent works [42, 38, 34, 36] leverage NVIDIA Warp to accelerate depth rendering, but they either lack support for dynamic meshes or do not demonstrate extension to multiple cameras. Consequently, end-to-end perceptive control under asymmetric multi-view observations across diverse challenging terrains remains largely underexplored. Moreover, these policies often struggle to generalize to unseen real-world terrain geometries (e.g., different terrain widths) beyond training distributions, and rarely evaluate robustness under loco-manipulation disturbances such as crouching down to pick up payloads and transporting them while traversing challenging terrains. In contrast, our method `RPL` targets

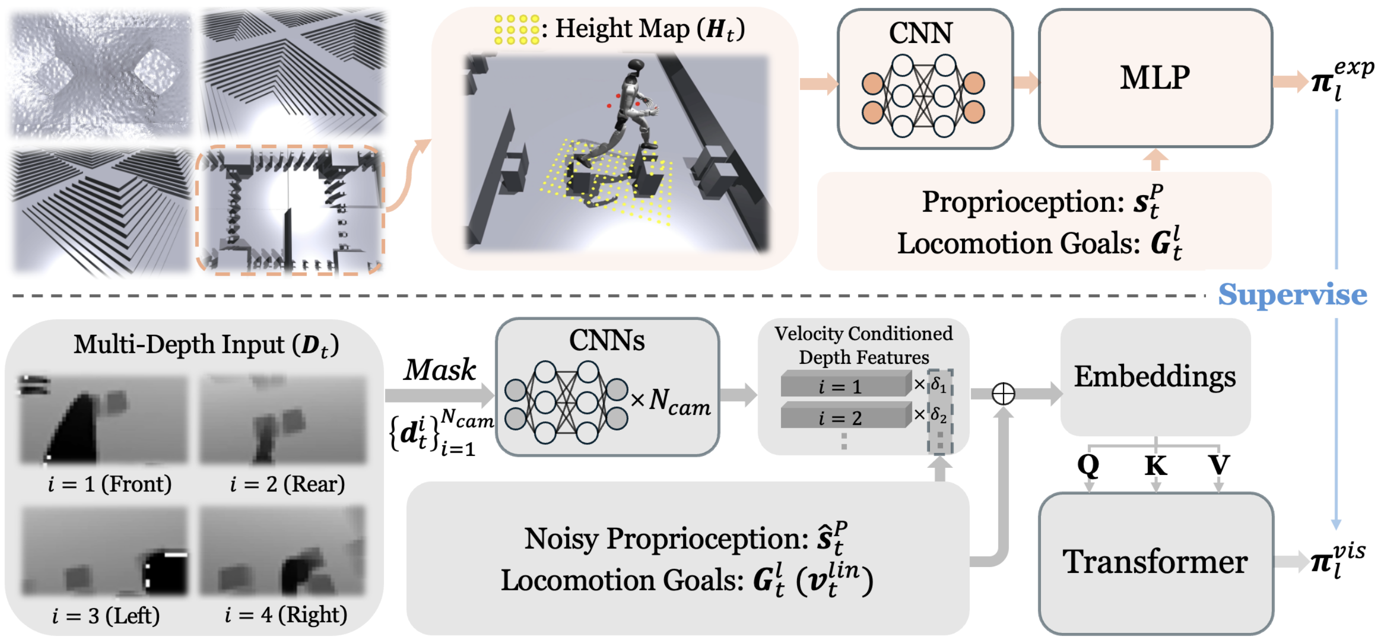

Fig. 2. **Overview of the Two-Stage Training Framework in** `RPL`**. Top (Stage 1)**: We train terrain-specific expert policies using privileged height-map observations to master decoupled locomotion and manipulation. **Bottom (Stage 2)**: We distill the expert policies into a single multi-view, depth-based transformer policy. Multi-camera depth inputs are encoded by CNN backbones with random side masking (Section III-B4) and depth feature scaling based on velocity commands (Section III-B3), fused into visual features, and combined with noisy proprioceptive observations and task goals to predict student actions.

robust multi-directional humanoid perceptive locomotion over challenging terrains with payloads, enabled by scalable multi-depth rendering simulation and distillation techniques designed to generalize across unseen terrain widths.

III. RPL: Two Stage – From Multiple Experts to One General Visual Policy

RPL is a two-stage framework as illustrated in Figure 2. In Stage 1, we train terrain-specialized expert policies using privileged height-map observations over challenging terrains including slopes, stairs up, stairs down, and stepping stones (Figure 7). In Stage 2, we distill these experts into a single unified visual policy that takes front and back depth observations to enable robust bidirectional locomotion. For the policy training, we build upon the dual-agent formulation of FALCON [41], which factorizes whole-body control into a lower-body locomotion policy and an upper-body manipulation policy that act on shared proprioceptive history, which has been proven to be more efficient and achieves better performance than a single agent for whole-body control in previous work [41, 20, 7]. At each timestep t, both agents observe the same proprioceptive history $\mathbf{s}_t^p = \left[\mathbf{q}_{t-4:t}, \dot{\mathbf{q}}_{t-4:t}, \boldsymbol{\omega}_{t-4:t}^{\text{root}}, \mathbf{g}_{t-4:t}, \mathbf{a}_{t-5:t-1}\right]$. The lower-body policy is conditioned on locomotion goals $\mathbf{G}_t^l = \left[\mathbf{v}_t^{\text{lin}}, \mathbf{w}_t^{\text{yaw}}, \boldsymbol{\phi}_t^{\text{stance}}, \mathbf{h}_t^{\text{root}}, \mathbf{o}_t^{\text{torso}}\right]$, which specify target linear velocity, angular velocity, stance/walking mode, root height, and torso orientation. Here, we only enable root height and torso tracking during stance mode. The upper-body policy is conditioned on manipulation goals $\mathbf{G}_t^u = \mathbf{q}_{\text{upper},t}$, i.e., upper-body joint targets. We denote the perceptual observation at timestep t by $\mathbf{H}_t$ in Stage 1 (privileged height map), and by $\mathbf{D}_t = \{\mathbf{d}_t^{(i)}\}_{i=1}^{N_{\text{cam}}}$ in Stage 2 (multi-view depth). Importantly, we include perceptual observation only in the lower-body observation, since terrain perception primarily governs legged locomotion, while upper-body control can remain largely decoupled. The policies are defined as $\boldsymbol{\pi}_l : (\mathbf{s}_t^p, \mathbf{G}_t^l, \mathbf{H}_t \text{ or } \mathbf{D}_t) \rightarrow \mathbf{a}_t^l$, $\boldsymbol{\pi}_u : (\mathbf{s}_t^p, \mathbf{G}_t^u) \rightarrow \mathbf{a}_t^u$, with actions concatenated as $\mathbf{a}_t = [\mathbf{a}_t^l; \mathbf{a}_t^u]$ for the low-level PD controller. (See Table IV in the Appendix for the notation summary.)

A. Stage 1: Training Multiple Terrain Experts via Height Map

To provide high-quality teacher actions for distillation, we first train terrain-specialized expert policies for each terrain family. In this stage, the perceptual observation is the privileged height map. The height map is represented as a local grid of size $1.6\,\text{m} \times 1.0\,\text{m}$ with a resolution of $0.1\,\text{m}$. We jointly train the dual-agent policies with independent reward functions and optimize them using PPO [31]. We also adopt a force curriculum [41], but set the direction ratio $\gamma = [\gamma_x, \gamma_y, \gamma_z] = [0, 0, 1]$ to specifically address the payloads on the end-effector for robust loco-manipulation. We also adopt asymmetric actor–critic training [29], where the critics have access to additional privileged information during training, including root linear velocities and end-effector forces $\boldsymbol{F}_t^{ee}$. To encourage symmetric gait patterns, we further apply symmetry data augmentation [27] to $\mathbf{s}_t^p$, $\mathbf{G}_t^l, \mathbf{G}_t^u$, and $\mathbf{H}_t^l$.

1) Terrain Settings: We train specialized expert policies for four types of terrains as illustrated in Figure 2: (1) **Slopes**: ramps with inclinations up to 37°, arranged in pyramid forms; (2) **Stairs Up / Down**: ascending and descending staircases with step length of 0.25–0.30 m and 0.05–0.27 m heights; (3) **Stepping Stones**: column-shaped discrete footholds with diameters of 0.25–0.40 m, separated by gaps of 0.05–0.70 m, and height variations up to 0.05 m. Note that the stairs and

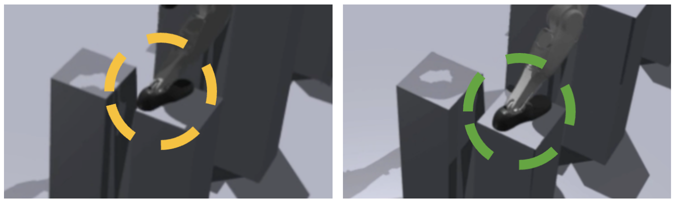

Fig. 3. Stepping stones without (left) and with (right) foot edge penalty.

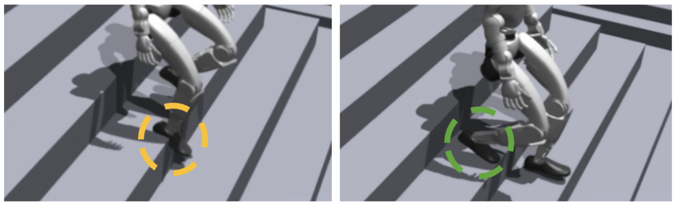

Fig. 4. Stairs without (left) and with (right) foothold penalty.

stepping stones are highly challenging given the robot's $0.21\,\mathrm{m}$ foot length, which is close to the minimum stairs step width and foothold diameter ($0.25\,\mathrm{m}$), leaving little margin for foothold placement errors.

2) Reward Design: We adopt the dual-agent rewards in [41] and add the following rewards for stable and precise terrain traversal as well as torso motions:

- **Foot Edge Penalty**: Similar to [6], we precompute binary dilated edge masks and penalize contacts when any sampled point on the foot sole lies on an edge. As shown in Figure 3, this encourages stable footholds that remain well separated from terrain edges.
- **Foothold Penalty**: We develop a dense sampling-based foothold penalty [37] as shown in Figure 4, which samples a grid across the foot sole and penalizes invalid coverage, discouraging foot placements that result in partial or empty support.
- **Torso Orientation Tracking**: We track the target torso orientation (roll, pitch, yaw) by computing the projected gravity error between the reference and actual torso frames [18]. The reward is formulated as $r_{torso} = \exp(-\|\mathbf{g}_{\text{proj}} - \mathbf{g}_{\text{ref}}\|^2/\sigma)$, where $\mathbf{g}_{\text{proj}}$ is the gravity vector projected into the torso frame. It gives the humanoid robot a large torso-orientation tracking range during stance by coordinating both the waist and hip joints.

We select reward terms based on the terrain type. For stairs, we use a foothold penalty instead of foot-edge, since edge-based dilation becomes overly conservative on narrow treads (0.25m stairs vs. 0.21m feet), causing the valid region to nearly vanish. For stepping stones, we use the foot-edge penalty to enforce center-aligned placements, since the small and isolated stone supports offer little tolerance to foot placement errors, while foothold-based criteria permit fragile boundary contacts that can be amplified by action noise during distillation and sim-to-real transfer.

B. Stage 2: Distillation into Unified Visual Policy

After all the terrain experts based on height maps are obtained, we distill them into one unified visual policy for multi-directional perceptive locomotion. The perceptual observation $\mathbf{P}_t^l$ consists of depth inputs from multiple depth cameras.

1) Distillation Formulation: As shown in Figure 2, we distill the terrain-specialized expert policies into a single unified visual locomotion policy using DAgger [30]. The student lower-body policy $\boldsymbol{\pi}_l^{\text{vis}}$ takes noisy proprioception $\hat{\mathbf{s}}_t^p$ and multi-view depth inputs $\mathbf{D}_t = \{\mathbf{d}_t^{(i)}\}_{i=1}^{N_{\text{cam}}}$, while the expert policies $\boldsymbol{\pi}_{l,k}^{\text{exp}}$ are conditioned on noise-free proprioception $\mathbf{s}_t^p$ and privileged height maps $\mathbf{H}_t$. We train the student by minimizing the action regression loss:

$$\mathcal{L}_{\text{distill}} = \mathbb{E}_{k,t}\left[\left\|\boldsymbol{\pi}_l^{\text{vis}}(\mathbf{s}_t^p, \mathbf{G}_t^l, \mathbf{D}_t) - \boldsymbol{\pi}_{l,k}^{\text{exp}}(\mathbf{s}_t^p, \mathbf{G}_t^l, \mathbf{H}_t)\right\|_2^2\right]. \tag{1}$$

The final deployed controller combines the distilled visual locomotion policy $\boldsymbol{\pi}_l^{\text{vis}}$ with the blind upper-body policy $\boldsymbol{\pi}_u$, producing the whole-body action for low-level PD tracking.

2) Efficient Multi-Depth System Simulation: We implement a GPU-efficient multi-depth camera simulation system using NVIDIA Warp [22], enabling massively parallel depth image synthesis via ray casting over both dynamic robot meshes and a shared static terrain mesh (Algorithm 1). For each environment e, camera c, pixel (x, y) and the corresponding camera intrinsics $\mathbf{K}_e$, we generate a camera-frame ray

$$\mathbf{r}_c = \mathbf{K}_e^{-1}[x, y, 1]^\top, \tag{2}$$

rotate it into the world frame using the camera orientation $\mathbf{q}_c^{e,c}$, and cast the ray from the camera origin $\mathbf{o} = \mathbf{p}_c^{e,c}$ along direction $\mathbf{d}$. The depth value is defined as the Euclidean distance from $\mathbf{o}$ to the closest ray–mesh intersection, clipped by a far-plane distance $d_{\max}$.

To avoid costly per-frame refitting of articulated robot meshes in the world frame [16], we keep each robot body mesh $\mathcal{M}_b$ canonicalized in its local body frame and instead transform world-frame rays into body-local coordinates:

$$\mathbf{o}_b = \mathbf{R}_b^\top(\mathbf{o} - \mathbf{t}_b), \quad \mathbf{d}_b = \mathbf{R}_b^\top \mathbf{d}, \tag{3}$$

where $(\mathbf{t}_b, \mathbf{R}_b)$ denotes the pose of body b in the world frame. For each ray, we first query all dynamic robot body meshes in their local frames and maintain the closest hit distance $z^\star$, which is used as an upper bound for early termination. We then query the shared static terrain mesh in the world frame and update $z^\star$ if a closer hit is found. The final depth value is set to $z^\star$, or $d_{\max}$ if no intersection occurs.

This design achieves high efficiency by replacing expensive mesh refitting with lightweight ray transformations, executing all ray–mesh queries in a single fused Warp kernel parallelized over environments, cameras, and pixels, and capturing the entire ray-casting pipeline into a CUDA graph for low-overhead replay across simulation steps. The system further supports per-environment camera intrinsics randomization by batching $\{\mathbf{K}_e^{-1}\}$ without control-flow divergence. We demonstrate its superior efficiency in Section IV.

Algorithm 1: Multi-Depth Ray Casting Kernel

```
Input: Static terrain mesh $\mathcal{M}_{\text{terrain}}$; robot body meshes
       $\{\mathcal{M}_b\}_{b=1}^{B}$; camera poses $\{(\mathbf{p}_c^{e,c}, \mathbf{q}_c^{e,c})\}$; body
       poses $\{(\mathbf{t}_b^e, \mathbf{q}_b^e)\}$; batched inverse intrinsics
       $\{\mathbf{K}_e^{-1}\}$; far plane $d_{\max}$; principal pixel $(c_x, c_y)$
Output: Depth image $\mathbf{D} \in \mathbb{R}^{N\times C\times H\times W}$
1  foreach $(e, c, x, y) \in [N] \times [C] \times [W] \times [H]$ do
2      $(\mathbf{o}, \mathbf{d}, m) \leftarrow$
         BuildDepthRay$(\mathbf{p}_c^{e,c}, \mathbf{q}_c^{e,c}, \mathbf{K}_e^{-1}, x, y, c_x, c_y)$;
3      $z^\star \leftarrow d_{\max}$; $d_{\text{bound}} \leftarrow d_{\max}$;
       /* 1) Query dynamic robot bodies */
4      for $b \leftarrow 1$ to $B$ do
5          $(\mathbf{o}_l, \mathbf{d}_l) \leftarrow$ WorldToBodyRay$(\mathbf{o}, \mathbf{d}, \mathbf{t}_b^e, \mathbf{q}_b^e)$;
6          if MeshRayQuery$(\mathcal{M}_b, \mathbf{o}_l, \mathbf{d}_l,\ d_{\text{bound}}/m) \to t$ then
7              $z^\star \leftarrow \min(z^\star,\ m\,t)$;
8              $d_{\text{bound}} \leftarrow z^\star$ ; // early termination
       /* 2) Query static terrain */
9      if MeshRayQuery$(\mathcal{M}_{\text{terrain}}, \mathbf{o}, \mathbf{d},\ d_{\text{bound}}/m) \to t$ then
10         $z^\star \leftarrow m\,t$;
11     $\mathbf{D}[e, c, y, x] \leftarrow z^\star$;
```

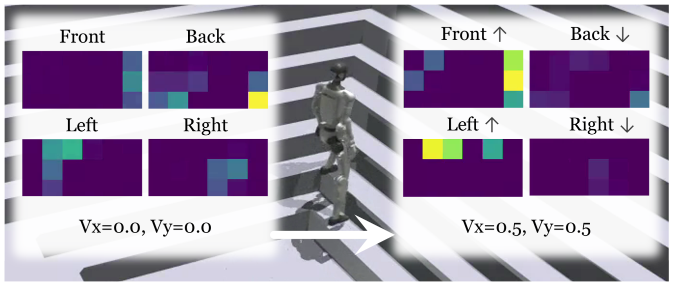

Fig. 5. **Illustration of DFSV.** CNN feature maps from four directional depth cameras under a zero command (v_x=0, v_y=0, left) and a diagonal command (v_x=0.5, v_y=0.5, right), where the heatmaps visualize each camera's relevance to the commanded velocity direction.

3) Depth Feature Scaling Based on Velocity Commands: When the robot is equipped with multiple depth cameras $\{\mathbf{d}_t^{(i)}\}_{i=1}^{N_{\text{cam}}}$ facing different directions, the relevance of each camera's observation depends on the locomotion direction. To address this, we introduce Depth Feature Scaling based on Velocity commands (DFSV), as illustrated in Figure 5. Let $\hat{\boldsymbol{n}}_i \in \mathbb{R}^2$ denote the normalized viewing direction of camera i in the robot's local frame. Given the planar velocity command $\mathbf{v}_l^{lin} = [v_x, v_y]^\top$, we compute attention scales for each camera:

$$\delta_i = 1 - \sigma\left(-k\left(\langle \mathbf{v}_l^{lin}, \hat{\boldsymbol{n}}_i\rangle - v_{\text{th}}\right)\right), \quad i = 1, \ldots, N_{\text{cam}} \quad (4)$$

where k controls the transition sharpness, v_{th} is a small velocity threshold, and $\langle \cdot, \cdot \rangle$ denotes the inner product. This formulation ensures that cameras whose viewing directions align with the commanded velocity are emphasized ($\delta_i \approx 1$), while those facing away are suppressed ($\delta_i \approx 0$). For example, with a front-back dual camera setup ($\hat{\boldsymbol{n}}_{\text{front}} = [1, 0]^\top$, $\hat{\boldsymbol{n}}_{\text{back}} = [-1, 0]^\top$), forward motion ($v_x > 0$) naturally emphasizes the front camera. The scales are applied element-wise to the CNN feature vectors $\mathbf{f}_i$ before concatenation:

$$\mathbf{f}_{\text{fused}} = \bigoplus_{i=1}^{N_{\text{cam}}} \delta_i \cdot \mathbf{f}_i \quad (5)$$

enabling the policy to focus on task-relevant visual information without additional learned attention parameters.

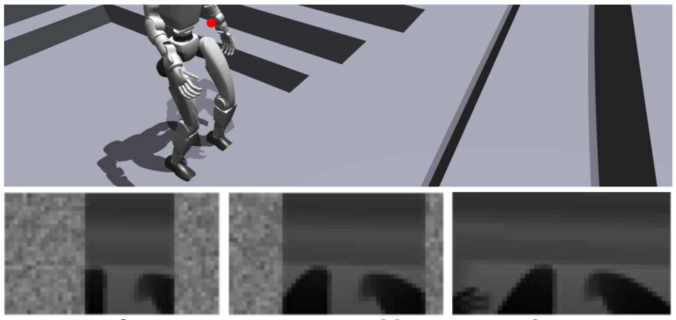

Fig. 6. **Illustration of RSM.** The red dot on the top figure is where the front camera is placed. The bottom three figures are different masking modes.

4) Random Side Masking for Unseen Terrain Widths: In real-world deployment, terrains often have varying widths that differ from the wide terrains in the training environments. To improve generalization to narrower terrains, we introduce Random Side Masking (RSM) as illustrated in Figure 6. During training, we stochastically occlude peripheral regions of the depth image with random noise, forcing the policy to make decisions from the central visible region and thereby reducing reliance on lateral context. For each environment, we sample a masking mode $\in \{\texttt{none}, \texttt{small}, \texttt{large}\}$ with terrain-specific probabilities mode $\sim$ Categorical$(\mathbf{p}_k)$, where $\mathbf{p}_k = \left[p_k^{\texttt{none}}, p_k^{\texttt{small}}, p_k^{\texttt{large}}\right]$ depends on the terrain type k and larger masks correspond to narrower effective terrain widths. Occluded pixels are filled with uniformly sampled random depth values within $[d_{\text{near}}, d_{\text{far}}]$, preventing the policy from exploiting boundary artifacts.

The terrain-specific sampling reflects the differing reliance on lateral information across terrain types. For continuous terrains such as stairs and slopes, traversability depends primarily on the local geometry directly ahead, so larger masks (simulating narrower terrains) can be applied more frequently. However, for discrete terrains such as stepping stones, valid footholds may only be visible in the peripheral field of view, requiring the policy to retain wider lateral visibility and thus favoring smaller masks. We observe (Section III-B4) that RSM improves generalization on continuous terrains with unseen widths, while preserving performance on discrete terrains such as stepping stones due to terrain-specific masking probabilities.

IV. Experiments

In this section, we will present extensive experiments in both simulation and the real world to show the effectiveness and the robustness of our method `RPL`. We aim to answer the following questions:

Method	Dyn. Mesh	Num. Envs	$N_{\text{cam}} = 1$		$N_{\text{cam}} = 2$		$N_{\text{cam}} = 4$	
			VRAM (GB)↓	Iter. (s)↓	VRAM (GB)↓	Iter. (s)↓	VRAM (GB)↓	Iter. (s)↓
IsaacGym PhysX	✓	1024	16.8	35.6±4.1	22.5	70.1±8.3	33.9	146.5±16.1
IsaacSim RTX	✓	1024	17.5	5.3±0.1	23.2	7.6±0.1	34.4	12.6±0.1
IsaacSim Warp	×	1024	**12.8**	3.5±0.1	15.1	5.9±0.1	20.7	9.1±0.1
Ours (Alg. 1)	✓	1024	13.3	**1.3**±0.0	**14.6**	**1.5**±0.1	**17.3**	**1.9**±0.1

TABLE I
DEPTH RENDERING SCALABILITY WITH A FIXED NUMBER OF PARALLEL ENVIRONMENTS ACROSS DIFFERENT NUMBERS OF DEPTH CAMERAS.

- **Q1**: Does RPL enable scalable multi-depth rendering performance compared to the existing methods?
- **Q2**: Does the training architecture in RPL outperform others in terms of networks and distillation losses?
- **Q3**: Do DFSV and RSM robustify our locomotion performance on the terrains of unseen configurations (asymmetric visual inputs, different terrain geometry)?
- **Q4**: Does RPL enable real-world long-horizon locomotion over challenging terrains while remaining robust with carried payloads?

A. Experiment Setup

Training Configurations: We use Unitree G1 as the humanoid robot during training. We train all our policies in NVIDIA IsaacGym [24]. In stage 1, we use 4 NVIDIA L40S GPUs to train all the expert policies with 4096×4 paralleled simulation environments for 24 hours. In stage 2, we use 8 NVIDIA L40S GPUs to train the multi-depth policy with 1024×8 paralleled simulation environments for 12 hours.

Perception Settings: During distillation, we capture depth images at 10 Hz with a horizontal and vertical field of view of 101° × 69° and a raw resolution of 240×135 which are downsampled to 48×27. We could support $N_{\text{cam}} = 4$ depth cameras to cover front, rear, left and right views of the robot for omni-directional locomotion in simulation. Since the humanoid robot Unitree G1 provides mounting holes only at the front and rear of its torso shell, in order to minimize the hardware modification for real-world deployment, we only train and deploy with $N_{\text{cam}} = 2$ depth cameras in a front–back configuration as shown in Figure 8. This setup supports bidirectional locomotion with both linear and angular velocity commands. We use ZED 2i cameras with the *neural_light* depth mode at 15 Hz. The raw depth images are captured at the same field of view and 1280×720 resolution and downsampled to 48 × 27 before being fed into the policy. Furthermore, we apply some depth randomization during distillation to simulate the realistic depth sensor, which include depth process delay, depth noise and dropout probability summarized in Table III.

Real-World Terrain Settings: We conduct the real-world experiments on a self-designed terrain course and a curved staircase in a building. The terrain course consists of (i) a 20° slope, (ii) 1.2 m-wide staircases with a fixed step height of 20 cm and two different step lengths (22 cm and 25 cm), and (iii) stepping stones with a 25 cm side length separated by 60 cm gaps. These terrains are highly challenging for our humanoid, whose single foot is approximately 21 cm long,

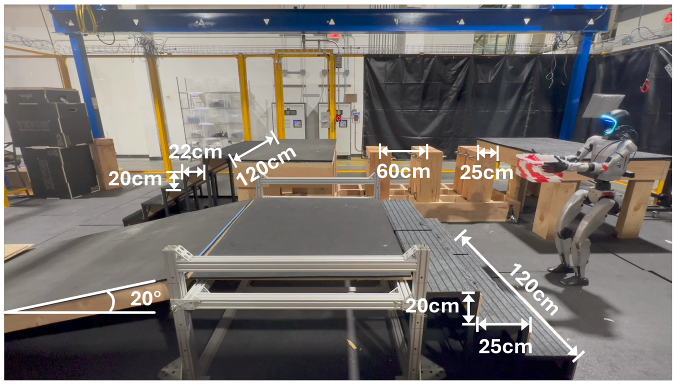

Fig. 7. Overview of the terrain course including slopes, stairs, and stepping stones. Key geometric dimensions are annotated in white.

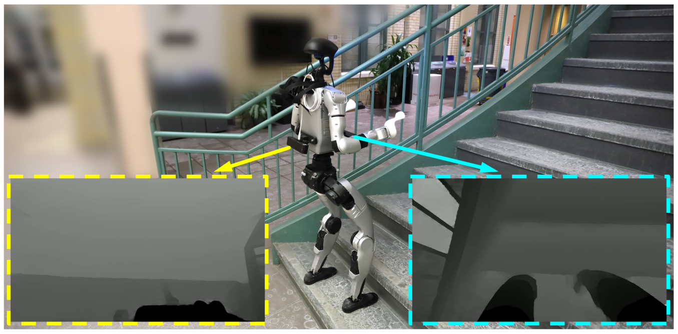
Fig. 8. Real-world robot setup with front and rear depth cameras for bidirectional locomotion.

leaving limited support area and requiring precise foot placement, especially on stairs and stepping stones. The staircase in the building is a curved one with a step height of 20 cm and a step length of 30 cm, posing additional challenges due to required turning motions and non-zero angular velocity.

B. Scalable Multi-Depth Rendering Performance

To answer **Q1** (*Does RPL enable scalable multi-depth rendering performance compared to the existing methods?*), we first benchmark the capability and the speed of different depth rendering pipelines in RPL and the existing simulators as shown in Table I. We evaluate the depth rendering performance in a locomotion-only task setting with $N_{\text{cam}} = 1, 2, 4$ depth cameras over all types of terrains (slopes, stairs and stepping stones), focusing on VRAM usage and iteration time. The depth resolution here is 240 × 135. We use one single NVIDIA L40S GPU for this evaluation. The other depth rendering baselines include: (1) IsaacGym PhysX; (2) IsaacSim [26]

Task	N_{cam} (Config.)	Slopes	Stairs Up	Stairs Down	Stepping Stones
Bidirectional	*Expert Level*	*6.0*	*6.0*	*6.0*	*6.0*
	1 (Down)	**6.0**±0.0	**6.0**±0.0	**5.9**±0.1	5.1±0.1
	2 (F + B)	**6.0**±0.0	**6.0**±0.0	**6.0**±0.0	**6.0**±0.0
	4 (F + B + L + R)	**6.0**±0.0	**6.0**±0.0	**6.0**±0.0	**6.0**±0.0
Omnidirectional	*Expert Level*	*6.0*	*6.0*	*6.0*	*5.6*
	1 (Down)	**6.0**±0.0	**6.0**±0.0	**5.9**±0.1	3.0±0.2
	2 (F + B)	**6.0**±0.0	**6.0**±0.0	**5.9**±0.1	**4.5**±0.1
	4 (F + B + L + R)	**6.0**±0.0	**6.0**±0.1	**6.0**±0.0	**4.6**±0.0

TABLE II
TERRAIN LEVELS ↑ ACHIEVED UNDER DIFFERENT NUMBERS OF DEPTH CAMERAS FOR BIDIRECTIONAL AND OMNIDIRECTIONAL LOCOMOTION.

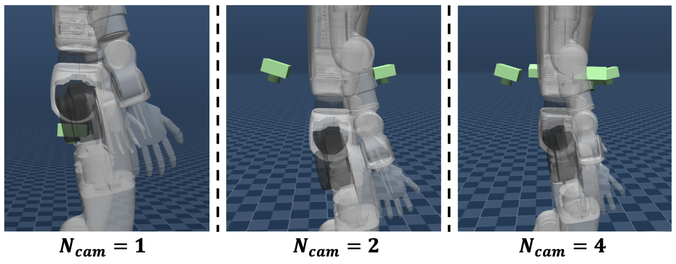

Fig. 9. The camera configurations for different N_{cam}.

RTX (i.e., TiledCamera); (3) IsaacSim Warp (i.e., RayCasterCamera). From Table I, we can conclude that the multi-depth rendering pipeline in RPL allows ray-casting against both dynamic and static meshes as well as a 5× speed up compared to the most efficient one, i.e., IsaacSim Warp which does not support dynamic mesh ray-casting.

To demonstrate the necessity of multiple cameras for multi-directional locomotion, we compare the achieved terrain levels for $N_{\text{cam}} = 1, 2, 4$ under bidirectional and omnidirectional locomotion, as summarized in Table II. Bidirectional locomotion supports forward and backward walking, whereas omnidirectional locomotion enables movement in all planar directions (forward, backward, left, and right). The corresponding camera configurations are shown in Figure 9. When $N_{\text{cam}} = 1$, a downward-facing camera is used to maximize terrain coverage. As the number of depth cameras decreases, performance degrades especially on stepping stones, where sparse footholds with gaps of up to 70 cm demand a wide field of view aligned with the walking direction. These results indicate that reliable bidirectional and omnidirectional locomotion benefits from having at least two depth cameras covering each potential walking direction to ensure sufficient terrain visibility.

C. Training Architecture and Distillation Ablations

To answer **Q2** (*Does the training architecture in* RPL *outperform others in terms of networks and distillation losses?*), we compare several distillation baselines with different backbone architectures and training objectives, and report the loss curves in Figure 10 (a). Specifically, we evaluate: (i) *CNN+MLP*, which encodes depth with a CNN and predicts actions from concatenated visual and proprioceptive features via an MLP; (ii) *CNN+RNN*, which replaces the MLP head in (i) with temporal recurrence; (iii) *CNN+Transformer*, which replaces the MLP head with a self-attention transformer to fuse multi-view observations; and (iv) *U-Net+Transformer*, which reconstructs a height map from multi-view depth and predicts actions from its concatenation with proprioception. Concretely, given the depth input $\mathbf{D}_t = \{\mathbf{d}_t^{(i)}\}_{i=1}^{N_{\text{cam}}}$, a reconstruction network f_{rec} outputs a predicted height map $\hat{\mathbf{H}}_t = f_{\text{rec}}(\mathbf{D}_t)$:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_t \left[\| f_{\text{rec}}(\mathbf{D}_t) - \mathbf{H}_t \|_2^2 \right], \tag{6}$$

The model is trained with a weighted sum of the reconstruction loss and the action distillation loss in Equation (1). Among them, we choose (iii) *CNN+Transformer* in RPL.

Overall, *CNN+Transformer* achieves the lowest distillation loss, outperforming *CNN+RNN*, *CNN+MLP*, and *U-Net+Transformer*. While the U-Net reconstruction baseline introduces an intermediate representation loss, its reliance on pixel-wise reconstruction makes it incompatible with random side masking (RSM), limiting robustness under partial observations. In contrast, attention-based fusion enables integration of multi-view depth inputs without requiring explicit geometric reconstruction, while achieving the lowest loss. These results motivate our choice of the transformer policy in RPL.

D. Robustness to Asymmetric Depth Observations and Unseen Terrain Widths

To answer **Q3** (*Do DFSV and RSM improve robustness on terrains with unseen configurations, e.g., asymmetric visual inputs and different terrain geometry?*), we perform ablations on the two key components used during distillation: (1) Random Side Masking (RSM) and (2) Depth Feature Scaling Based on Velocity Commands (DFSV). We evaluate three variants: (1) RPL; (2) RPL w/o RSM; and (3) RPL w/o DFSV. Interestingly, all variants achieve similar final distillation loss values (cf. Figure 10 (b)), suggesting that the student can match the expert actions even under partially masked depth observations. This indicates that the masked inputs still preserve sufficient task-relevant information for action supervision during distillation.

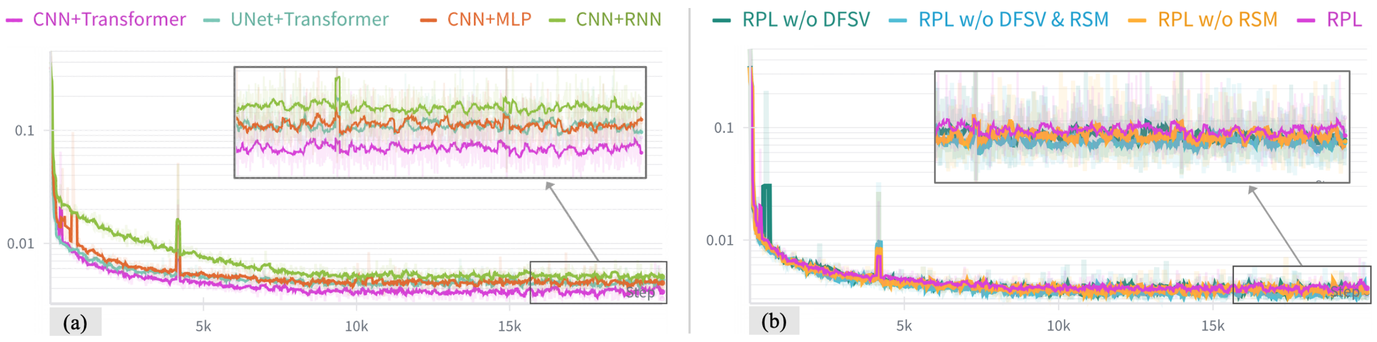

Fig. 10. **Distillation Loss Comparison.** (a) compares the distillation performance among different network architectures. (b) compares among (1) RPL; (2) RPL w/o DFSV & RSM; (3) RPL w/o DFSV; and (4) RPL w/o RSM.

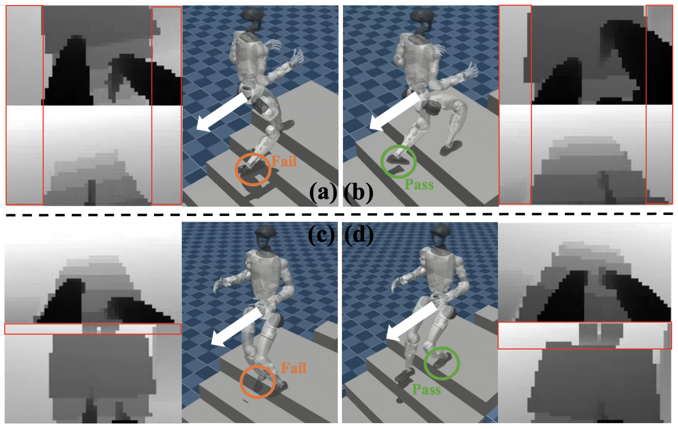

Fig. 11. The locomotion performance of different policies on unseen width of stairs with asymmetric depth observations. (a) RPL w/o RSM; (c) RPL w/o DFSV; (b) and (d) RPL. The top row shows backward descent, while the bottom row shows forward descent.

Despite comparable training losses, the methods differ substantially in out-of-distribution (OOD) generalization. Notably, RPL achieves the most robust back-and-forth locomotion on terrains with unseen narrow widths. To analyze deployment robustness, we visualize sim-to-sim rollouts in MuJoCo on a challenging course consisting of a narrow stair section (1.2m wide) followed by stepping stones. In Figure 11, (a) and (c) show RPL w/o RSM and RPL w/o DFSV, respectively, while (b) and (d) show the full RPL. This setup induces two OOD cases. First, the test stairs are narrower than those seen during training; during backward descent in Figure 11 (a,b), RPL w/o RSM fails to generalize to unseen widths, whereas RPL remains stable. Second, during forward descent in Figure 11 (c,d), the front camera observes stairs while the rear camera still perceives stepping stones, creating asymmetric multi-view inputs; without DFSV, the policy is distracted by irrelevant rear-view cues and fails the traversal. Overall, RSM and DFSV jointly enable robust bidirectional stair locomotion under OOD geometry and asymmetric perception.

E. Robust Real-World Bidirectional Locomotion

To answer **Q4** (*Does RPL work for long-horizon locomotion over challenging terrain course and remain robust with carried payloads?*), we conduct extensive real-world experiments on a 50 m terrain course and a curved staircase in a building, as described in Section IV-A. We use the onboard NVIDIA Jetson Orin NX 16GB to deploy our depth-based transformer policy and multi-camera depth acquisition. The transformer policy runs at 50 Hz. To asynchronously handle depth capture and policy inference, we use shared memory as an inter-process buffer: the depth capture process continuously writes the latest depth images, while the policy inference process reads them as the policy input. The same shared memory technique is also applied to the teleoperation pipeline for picking up the payload, where we write-and-read the Inverse Kinematics (IK) targets from the Apple Vision Pro and use another thread for the IK solver. These asynchronous designs significantly reduce end-to-end onboard deployment latency, enabling stable real-time control on the onboard platform.

Our results, as demonstrated in our **supplementary video** and Figure 1, showcase long-horizon locomotion across diverse challenging terrains. The robot consistently achieves robust bidirectional locomotion, supporting both forward and backward traversal, and remains stable while performing whole-body loco-manipulation tasks, such as bending down to pick up and transport a 2 kg payload throughout the entire terrain course. These results demonstrate the robustness of RPL under long horizons, asymmetric visual observations, narrow terrain widths, and additional payload-induced disturbances.

V. Conclusion

We propose RPL, a two-stage learning framework for multi-directional humanoid perceptive locomotion over complex terrains and remain robust with payloads. RPL first trains terrain-specialized expert policies with privileged height-map observations to learn decoupled locomotion and manipulation, and then distills them into a single depth-based transformer policy that fuses multi-view depth inputs for robust bidirectional locomotion. To enable scalable and realistic depth distillation, we develop a massively parallel multi-depth rendering system that ray-casts against dynamic robot meshes and static terrain meshes, achieving an 5× speedup over existing simulator pipelines while modeling realistic depth latency, noise, and dropout. We further introduce DFSV and RSM to improve robustness under asymmetric multi-view observations and unseen terrain widths. Extensive simulation and real-world experiments on the Unitree G1 demonstrate long-horizon back-and-forth locomotion with payloads across diverse challenging terrains.

VI. Limitations

Despite its strong performance, RPL has two main limitations. First, we do not demonstrate real-world sideways locomotion on the terrains considered in this paper, as achieving expert-level omnidirectional performance across all terrains remains non-trivial for the distilled policy. Second, while RPL improves robustness to asymmetric observations via DFSV, it mostly relies on fixed viewpoints and does not explicitly learn active exploration or viewpoint selection for highly occluded or ambiguous loco-manipulation scenarios.

Acknowledgments

We thank Guanqi He, Zhen Wu, Xiaofeng Guo for their help on the real-world experiments. We also thank Alejandro Escontrela for the insightful discussion on the depth simulation. The only dataset we used in this work is AMASS [23], which is solely for research purposes.

References

[1] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In Conference on robot learning, pages 403–415. PMLR, 2023.

[2] Qingwei Ben, Feiyu Jia, Jia Zeng, Junting Dong, Dahua Lin, and Jiangmiao Pang. HOMIE: humanoid loco-manipulation with isomorphic exoskeleton cockpit. CoRR, abs/2502.13013, 2025. doi: 10.48550/ARXIV.2502.13013. URL https://doi.org/10.48550/arXiv.2502.13013.

[3] 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.

[4] Karim Bouyarmane, Kevin Chappellet, Joris Vaillant, and Abderrahmane Kheddar. Quadratic programming for multirobot and task-space force control. IEEE Transactions on Robotics, 35(1):64–77, 2018.

[5] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. arXiv preprint arXiv:2402.16796, 2024.

[6] 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.

[7] Ziluo Ding, Haobin Jiang, Yuxuan Wang, Zhenguo Sun, Yu Zhang, Xiaojie Niu, Ming Yang, Weishuai Zeng, Xinrun Xu, and Zongqing Lu. Jaeger: Dual-level humanoid whole-body controller. arXiv preprint arXiv:2505.06584, 2025.

[8] Nolan Fey, Gabriel B Margolis, Martin Peticco, and Pulkit Agrawal. Bridging the sim-to-real gap for athletic loco-manipulation. arXiv preprint arXiv:2502.10894, 2025.

[9] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. In Conference on Robot Learning (CoRL), 2024.

[10] Mohitvishnu S Gadde, Pranay Dugar, Ashish Malik, and Alan Fern. No more blind spots: Learning vision-based omnidirectional bipedal locomotion for challenging terrain. In 2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids), pages 601–608. IEEE, 2025.

[11] 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.

[12] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. arXiv preprint arXiv:2406.08858, 2024.

[13] Tairan He, Jiawei Gao, Wenli Xiao, Yuanhang Zhang, Zi Wang, Jiashun Wang, Zhengyi Luo, Guanqi He, Nikhil Sobanbab, Chaoyi Pan, et al. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. arXiv preprint arXiv:2502.01143, 2025.

[14] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. Science Robotics, 9(88):eadi7566, 2024.

[15] Mazeyu Ji, Xuanbin Peng, Fangchen Liu, Jialong Li, Ge Yang, Xuxin Cheng, and Xiaolong Wang. Exbody2: Advanced expressive humanoid whole-body control. arXiv preprint arXiv:2412.13196, 2024.

[16] Mihir Kulkarni, Welf Rehberg, and Kostas Alexis. Aerial gym simulator: A framework for highly parallelized simulation of aerial robots. IEEE Robotics and Automation Letters, 2025.

[17] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. Science robotics, 5 (47):eabc5986, 2020.

[18] Jialong Li, Xuxin Cheng, Tianshu Huang, Shiqi Yang, Ri-Zhao Qiu, and Xiaolong Wang. Amo: Adaptive motion optimization for hyper-dexterous humanoid whole-body control. arXiv preprint arXiv:2505.03738, 2025.

[19] Junheng Li, Junchao Ma, Omar Kolt, Manas Shah, and Quan Nguyen. Dynamic loco-manipulation on hector: Humanoid for enhanced control and open-source research. arXiv preprint arXiv:2312.11868, 2023.

[20] Yitang Li, Yuanhang Zhang, Wenli Xiao, Chaoyi Pan, Haoyang Weng, Guanqi He, Tairan He, and Guanya Shi. Hold my beer: Learning gentle humanoid locomotion and end-effector stabilization control. In RSS 2025 Workshop on Whole-body Control and Bimanual Manipulation:

Applications in Humanoids and Beyond.

[21] 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.

[22] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. In NVIDIA GPU Technology Conference (GTC), volume 3, 2022.

[23] 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.

[24] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In Joaquin Vanschoren and Sai-Kit Yeung, editors, Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/28dd2c7955ce926456240b2ff0100bde-Abstract-round2.html.

[25] 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.

[26] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, et al. Orbit: A unified simulation framework for interactive robot learning environments. IEEE Robotics and Automation Letters, 8 (6):3740–3747, 2023.

[27] Mayank Mittal, Nikita Rudin, Victor Klemm, Arthur Allshire, and Marco Hutter. Symmetry considerations for learning task symmetric robot policies. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 7433–7439. IEEE, 2024.

[28] Masaki Murooka, Kevin Chappellet, Arnaud Tanguy, Mehdi Benallegue, Iori Kumagai, Mitsuharu Morisawa, Fumio Kanehiro, and Abderrahmane Kheddar. Humanoid loco-manipulations pattern generation and stabilization control. IEEE Robotics and Automation Letters, 6(3): 5597–5604, 2021.

[29] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. arXiv preprint arXiv:1710.06542, 2017.

[30] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. CoRR, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[32] Mohsen Sombolestan and Quan Nguyen. Adaptive force-based control of dynamic legged locomotion over uneven terrain. IEEE Transactions on Robotics, 2024.

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

[34] 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.

[35] Wandong Sun, Baoshi Cao, Long Chen, Yongbo Su, Yang Liu, Zongwu Xie, and Hong Liu. Learning perceptive humanoid locomotion over challenging terrain. arXiv preprint arXiv:2503.00692, 2025.

[36] Dewei Wang, Xinmiao Wang, Xinzhe Liu, Jiyuan Shi, Yingnan Zhao, Chenjia Bai, and Xuelong Li. More: Mixture of residual experts for humanoid lifelike gaits learning on complex terrains. arXiv preprint arXiv:2506.08840, 2025.

[37] 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.

[38] Zifan Wang, Teli Ma, Yufei Jia, Xun Yang, Jiaming Zhou, Wenlong Ouyang, Qiang Zhang, and Junwei Liang. Omni-perception: Omnidirectional collision avoidance for legged locomotion in dynamic environments. arXiv preprint arXiv:2505.19214, 2025.

[39] Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C Karen Liu, Rocky Duan, and Guanya Shi. Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. arXiv preprint arXiv:2509.26633, 2025.

[40] Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. arXiv preprint arXiv:2107.03996, 2021.

[41] Yuanhang Zhang, Yifu Yuan, Prajwal Gurunath, Ishita Gupta, Shayegan Omidshafiei, Ali-akbar Agha-mohammadi, Marcell Vazquez-Chanlatte, Liam Pedersen, Tairan He, and Guanya Shi. Falcon: Learning force-adaptive humanoid loco-manipulation. arXiv preprint arXiv:2505.06776, 2025.

[42] 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.

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

Appendix

TABLE III
DOMAIN RANDOMIZATION TERMS DURING DISTILLATION.

Term	Value
Dynamics Randomization	
Friction	$\mathcal{U}(0.5, 1.25)$
Link mass	$\mathcal{U}(0.9, 1.2) \times$ default kg
Base mass	$\mathcal{U}(-1.0, 3.0)$ kg
Base COM range	$x \sim \mathcal{U}(-0.025, 0.025)$m
	$y \sim \mathcal{U}(-0.05, 0.05)$m
	$z \sim \mathcal{U}(-0.05, 0.05)$m
Control delay	$\mathcal{U}(0, 20)$ms
Perception Randomization (Depth Cameras)	
Depth cam translation	$x \sim \mathcal{U}(-0.025, 0.025)$ m
	$y \sim \mathcal{U}(-0.025, 0.025)$ m
	$z \sim \mathcal{U}(-0.025, 0.025)$ m
Depth cam rotation (Euler)	$\phi \sim \mathcal{U}(-2.5, 2.5)^\circ$
	$\theta \sim \mathcal{U}(-3.0, 3.0)^\circ$
	$\psi \sim \mathcal{U}(-2.5, 2.5)^\circ$
Depth cam FOV	$\Delta\text{FOV} \sim \mathcal{U}(-2.0, 2.0)^\circ$
Pixel dropout	$p_{\text{drop}} = 0.05$
Depth noise	$\sigma_d = 0.1 \cdot$ depth

TABLE IV
NOTATION SUMMARY.

Symbol	Meaning
Proprioception	
$\boldsymbol{s}_t^p$	Proprioceptive history
$\boldsymbol{q}_{t-4:t}$	Joint positions history
$\dot{\boldsymbol{q}}_{t-4:t}$	Joint velocities history
$\boldsymbol{\omega}_{t-4:t}^{\text{root}}$	Root angular velocity history
$\boldsymbol{g}_{t-4:t}$	Gravity vector history
$\boldsymbol{a}_{t-5:t-1}$	Action history
Goals	
$\boldsymbol{G}_t^l$	Lower-body locomotion goals
$\boldsymbol{v}_t^{\text{lin}}$	Linear velocity command
$\boldsymbol{w}_t^{\text{yaw}}$	Angular velocity command
$\boldsymbol{\phi}_t^{\text{stance}}$	Stance/walking mode command
$\boldsymbol{h}_t^{\text{root}}$	Root height command
$\boldsymbol{o}_t^{\text{torso}}$	Torso orientation command
$\boldsymbol{G}_t^u$	Upper-body manipulation goals
$\boldsymbol{q}_{\text{upper},t}$	Upper-body joint target
Perceptual Observation	
$\boldsymbol{H}_t$	Privileged height map
$\boldsymbol{D}_t$	Multi-view depth observation
$\boldsymbol{d}_t^{(i)}$	Depth image from camera i at time t
N_{cam}	Number of depth cameras