Omni-Perception: Omnidirectional Collision Avoidance for Legged Locomotion in Dynamic Environments

Zifan Wang[1], **Teli Ma**[1], **Yufei Jia**[4], **Xun Yang**[1], **Jiaming Zhou**[1], **Wenlong Ouyang**[1], **Qiang Zhang**[1,5], **Junwei Liang**[†1,2,3]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Spatialtemporal AI [3]The Hong Kong University of Science and Technology
[4]Department of Eletronic Engineering, Tsinghua University
[5]Beijing Innovation Center of Humanoid Robotics Co., Ltd.
https://github.com/aCodeDog/OmniPerception

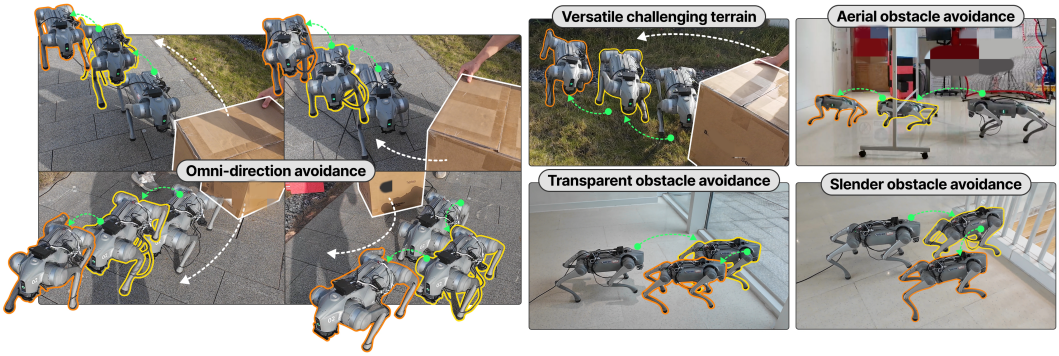


Figure 1: Validation scenarios for the Omni-Perception framework. Effective omnidirectional collision avoidance is demonstrated on the left, where the robot reacts to obstacles from various approach vectors. Robustness against diverse environmental features is shown on the right, including successful negotiation of aerial, transparent, slender, and ground obstacles. These results highlight the capacity of the Omni-Perceptio to achieve collision-free locomotion in challenging 3D settings directly from raw LiDAR input.

Abstract: Agile locomotion in complex 3D environments requires robust spatial awareness to safely avoid diverse obstacles such as aerial clutter, uneven terrain, and dynamic agents. Depth-based perception approaches often struggle with sensor noise, lighting variability, computational overhead from intermediate representations. In contrast, direct integration of LiDAR sensing into end-to-end learning for legged locomotion remains underexplored. We propose **Omni-Perception**, an end-to-end locomotion policy that achieves 3D spatial awareness and omnidirectional collision avoidance by directly processing raw LiDAR point clouds. At its core is PD-RiskNet (Proximal-Distal Risk-Aware Hierarchical Network), a novel perception module that interprets spatio-temporal LiDAR data for environmental risk assessment. To facilitate efficient policy learning, we develop **a high-fidelity LiDAR simulation toolkit** with realistic noise modeling and fast raycasting, compatible with platforms such as Isaac Gym, Genesis, and MuJoCo, enabling scalable training and effective sim-to-real transfer. Learning reactive control policies directly from raw LiDAR data enables the robot to navigate complex environments with static and dynamic obstacles more robustly than approaches relying on intermediate maps or limited sensing. We validate Omni-Perception through real-world experiments and extensive simulation, demonstrating strong omnidirectional avoidance capabilities and superior locomotion performance in highly dynamic environments.

Keywords: Legged Robot, Locomotion, Reinforcement Learning, LiDAR Perception

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

1 Introduction

Legged robots possess the unique potential to navigate complex, human-centric environments where wheeled or tracked systems often fail. Unlocking their full capabilities, however, requires not only agile locomotion but also robust perception and rapid reaction to diverse 3D surroundings [1, 2]. Achieving safe, high-speed locomotion among static and dynamic obstacles—ranging from ground clutter and uneven terrain to overhanging structures and moving agents—requires holistic spatial awareness and tightly coupled reactive control [3, 4]. Enabling such agile, collision-free movement in unstructured 3D environments remains a fundamental challenge [5, 6].

Most state-of-the-art controllers operate blindly using only proprioception, limiting their ability to handle complex terrain or unexpected obstacles [7, 8, 9, 2, 10, 11, 12]. While methods incorporating exteroception via depth cameras [13, 14] have advanced locomotion over uneven terrain, depth-based perception suffers from limitations such as sensitivity to lighting conditions, limited fields of view, noise susceptibility, and the computation overhead of maintaining intermediate representations like elevation maps [15, 16]. These representations, in turn, struggle with aerial clutter and complex non-planar obstacles. Alternative approaches that decouple navigation planning from locomotion control [17, 18] often lead to conservative behaviors, preventing full exploitation of a robot's agility.

Meanwhile, LiDAR sensors have become instrumental in fields such as autonomous driving, offering dense, lighting-invariant 3D measurements, and manipulation [19, 20, 21, 22, 23, 24, 25, 26]. LiDAR delivers rich geometric information ideal for navigating spatially complex environments. Despite its promise, the integration of LiDAR—**especially direct point cloud processing**—into end-to-end learning for legged locomotion remains underexplored [27, 28, 29]. This gap arises from the challenges of real-time point cloud processing within fast control loops, and from difficulties in accurately modeling LiDAR physics for effective sim-to-real transfer [30].

This work aims to bridge this critical gap. We posit that directly leveraging raw, spatio-temporal LiDAR point clouds within an end-to-end Reinforcement Learning (RL) policy can unlock robust 3D environmental awareness for legged robots. This enables the robot to achieve omnidirectional collision avoidance and agile navigation without relying on handcrafted intermediate representations or restrictive decoupled planning. We introduce **Omni-Perception**, a novel framework centered around **an end-to-end policy** trained using RL in a high-fidelity LiDAR simulator. Our core technical contribution is the **Proximal-Distal Risk-Aware Hierarchical Network (PD-RiskNet)**, a novel perception architecture that efficiently processes raw spatio-temporal LiDAR streams to assess multi-level environmental risks. To enable scalable and realistic training, we also develop a high-fidelity, cross-platform LiDAR simulation toolkit with realistic noise modeling and efficient parallel raycasting, supporting multiple physics engines including Isaac Gym, Genesis, MuJoCo, and others. By learning directly from 3D spatial data, Omni-Perception enables legged robots to dynamically track velocity commands while avoiding complex, multi-axis threats (e.g., aerial obstacles, ground traps, moving agents), pushing the boundaries of agile and safe navigation in unstructured environments.Our contributions are summarized as follows:

1. **End-to-End LiDAR-Driven Locomotion Framework:** We present Omni-Perception, the first framework to achieve 3D spatial awareness for legged robots by directly processing raw LiDAR point clouds within an end-to-end RL architecture.
2. **Novel LiDAR Perception Network (PD-RiskNet):**
3. **High-Fidelity LiDAR Simulation Toolkit:** We develop a high-performance, cross-platform LiDAR simulation toolkit featuring realistic noise models and fast parallel raycasting, enabling effective zero-shot sim-to-real policy transfer.
4. **Demonstration of Robust LiDAR-Driven Agility:** We validate Omni-Perception through extensive simulation and real-world experiments, demonstrating strong velocity tracking and omnidirectional avoidance capabilities across diverse and challenging environments containing static and dynamic 3D obstacles.

2 Related Work

2.1 Learning-Based Legged Locomotion

Reinforcement Learning (RL) has emerged as a powerful paradigm for developing sophisticated locomotion controllers for legged robots, demonstrating remarkable agility and robustness surpassing traditional model-based methods in many scenarios [9, 31, 32]. RL approaches have successfully generated controllers for high-speed running [8, 11] and challenging terrain traversal [33, 7]. These policies typically map proprioceptive states—and in some cases, limited exteroceptive information—directly to joint commands.

2.2 Exteroceptive Perception for Locomotion

Incorporating environmental perception is crucial for enabling legged robots to move through unknown or dynamic environments.

Depth-Based Perception: A significant body of work utilizes depth cameras. Many approaches reconstruct the environment into intermediate representations like 2.5D elevation maps, for foothold planning or policy input [34, 35]. However, elevation maps inherently struggle with non-planar obstacles like overhangs or aerial clutter and are sensitive to sensor noise. Other works attempt to use depth data more directly, either by learning end-to-end policies from depth images [36, 37, 13, 38] or extracting sparse features like ray distances from depth [3]. Nevertheless, these methods inherit the limitations of depth sensors, including sensitivity to lighting and limited range. Furthermore, sim-to-real transfer often requires extensive domain randomization, particularly noise injection into depth data, to address real-world sensor characteristics [37].

LiDAR-Based Perception: LiDAR sensors provide direct 3D measurements and are largely invariant to lighting conditions, making them fundamental in domains such as autonomous driving and mobile robot mapping [39, 40, 41]. However, their application within *learning-based, end-to-end legged locomotion* has been very limited. Existing uses often involve traditional pipelines (SLAM, path planning) rather than direct integration into RL policies[30, 27, 42]. Challenges include the high dimensionality and unstructured nature of point clouds, the computational cost of processing them at high control frequencies, and the difficulty of creating accurate, efficient LiDAR simulations for training [43, 44]. Although some works use simulated LiDAR-like inputs (e.g., ray casting) for perception [3], to the best of our knowledge, **Omni-Perception is the first framework to directly learn end-to-end locomotion policies from raw spatio-temporal LiDAR point clouds**, enabling robust omnidirectional collision avoidance and navigation.

2.3 Collision Avoidance for Mobile Robots

Collision avoidance is a fundamental problem in robotics[45]. Classical approaches often rely on geometric methods and explicit planning in configuration space [46, 47, 48]. Model Predictive Control (MPC) has been applied to legged robots, incorporating collision constraints into the optimization [49, 50]. However, MPC relies on accurate dynamic models, are computationally demanding, and tend to produce slow movements to ensure constraint satisfaction [51, 52].

Learning-based methods, particularly RL, offer an alternative by enabling robots to learn reactive avoidance behaviors[3, 53]. In aerial robotics, RL has been successfully applied to dynamic navigation tasks using perception inputs from RGB-D cameras or simulated laser scans [54]. Our work advances this line by introducing an end-to-end RL-based collision avoidance strategy that leverages rich 3D information directly from raw LiDAR streams, processed by our specialized PD-RiskNet, integrated tightly within the locomotion control policy.

3 Method

3.1 Problem Formulation

We address the problem of learning a continuous locomotion policy π for a legged robot, enabling it to track desired velocity commands while performing omnidirectional obstacle avoidance in complex 3D environments using onboard sensing. The policy is trained end-to-end using RL within a high-fidelity simulation environment featuring realistic LiDAR sensing.

The core task is to learn a policy $\pi : \mathcal{O} \to \Delta(\mathcal{A})$ that maps observations $o_t \in \mathcal{O}$ to a distribution over actions $a_t \in \mathcal{A}$ at each time step t. The observation space $\mathcal{O}$ comprises the information available to the policy at time t. It includes:

- **Proprioceptive State (o_t^{prop}):** A history of N_{hist} kinematic information, including joint positions (q_t), joint velocities ($\dot{q}_t$), base linear and angular velocities (v_t, ω_t), and base orientation represented by the projected gravity g_t.
- **Exteroceptive State (P_t):** A history of N_{hist} raw 3D point cloud data, providing spatio-temporal information about the surrounding environment.
- **Task Command (c_t):** The desired velocity command, typically a target linear velocity v_t^{cmd} and angular velocity ω_t^{cmd} in the robot's base frame.

Thus, the observation is $o_t = (o^{\text{prop}}_{t-N_{hist}+1:t}, P_{t-N_{hist}+1:t}, c_t)$. Our proposed perception network, PD-RiskNet (Sec. 3.3.1), processes the sequence of point clouds $P_{t-N_{hist}+1:t}$ and integrates the extracted features with the proprioceptive history $o^{\text{prop}}_{t-N_{hist}+1:t}$ and command c_t to inform the policy.

The action space $\mathcal{A}$ consists of low-level motor commands a_t, specifically the target joint positions issued to the robot's actuators. We frame the learning problem as optimizing the policy within an infinite-horizon discounted Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, r, \gamma, T)$. While the policy directly uses observations o_t, the underlying state $s_t \in \mathcal{S}$ may potentially include hidden simulator states or history. The transition dynamics $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ are governed by the physics simulator and sensor models. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (or approximated as $r(o_t, a_t)$) is designed to encourage specific behaviors (detailed in Sec. 3.3.2). The objective is to find the optimal policy π^* that maximizes the expected discounted sum of future rewards:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1, \dots)$ is a trajectory generated under policy π, $a_t \sim \pi(o_t)$, and $\gamma \in [0, 1)$ is the discount factor.

3.2 Custom LiDAR Rendering Framework

Motivation and Backend. While existing simulators, like NVIDIA Isaac Gym/Sim [55], Gazebo [56], Genesis [57], provide physics simulation capabilities, their rendering solutions often lack support for diverse lidar sensor models like Non-repetitive scan lidar or impose limitations on parallelism and cross-platform execution. To address these gaps, inspired by [58], we introduce a custom, high-performance LiDAR rendering framework that leverages NVIDIA Warp [59] for GPU acceleration and Taichi [60] for cross-platform compatibility, enabling execution on systems lacking dedicated NVIDIA GPUs, including CPU-only environments.

Optimized Mesh Management for Parallel Simulations. Simulating numerous environments with frequently moving dynamic objects poses a significant challenge. Transforming each mesh and rebuilding acceleration structures (BVHs) for each environment independently at each timestep [61] become computationally prohibitive in such scenarios. To overcome this bottleneck, we employ a specialized mesh partitioning and update strategy:

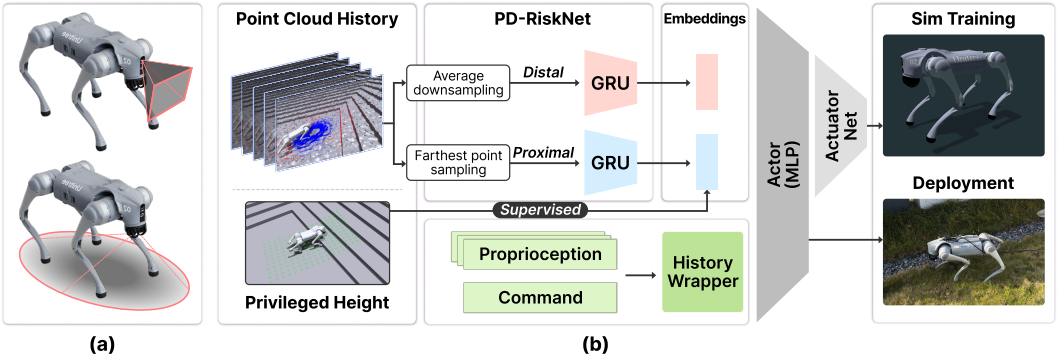


Figure 2: Proposed System Framework. (a) Visualization of differing sensor coverage: the typically narrow, forward-directed field of view of a depth camera (top) contrasted with the broader and longer range, coverage of a LiDAR sensor (bottom), shown on the Unitree Go2 robot. (b) Detailed diagram of the perception and control pipeline. Raw point cloud history is processed via two pathways within PD-RiskNet (average downsampling for distal features, farthest point sampling for proximal features, each fed to a GRU) generating spatial embeddings. In the point cloud history visualization, the dense blue points represent the proximal point cloud, while the sparse red points depict the distal point cloud. These embeddings are concatenated with historical proprioception and commands processed by a History Wrapper, and supervised using privileged height information. The combined features are input to an Actor (MLP) network that outputs actuator targets for sim-to-real training and deployment.

- **Per-Environment Static Mesh:** For each environment $\boldsymbol{\mathcal{E}}_i$, non-moving geometry (terrain, fixed obstacles) is represented as a static mesh $\boldsymbol{\mathcal{M}}_i^{\text{static}}$. Its BVH structure is built only once upon initialization, minimizing redundant computations.
- **Global Shared Dynamic Mesh:** We aggregate the meshes of *all* dynamic entities (e.g., robot parts, moving obstacles) from *all* N_{envs} parallel environments into a single, global dynamic mesh $\boldsymbol{\mathcal{M}}_{\text{global},t}^{\text{dynamic}}$. At simulation time t, only the vertex positions of this shared mesh are updated based on the latest transformations ($\mathbf{T}_{j,t}$) of the corresponding dynamic objects $\boldsymbol{\mathcal{M}}_j^{\text{dynamic}}$ across all environments.

Synchronous Update and Raycasting. LiDAR simulation involves raycasting against both the appropriate static mesh ($\boldsymbol{\mathcal{M}}_i^{\text{static}}$) and the *single* global dynamic mesh ($\boldsymbol{\mathcal{M}}_{\text{global},t}^{\text{dynamic}}$). Crucially, the vertex update for $\boldsymbol{\mathcal{M}}_{\text{global},t}^{\text{dynamic}}$ occurs synchronously for all environments, typically matching the sensor's update frequency. This centralized update mechanism avoids the substantial overhead of individually updating dynamic meshes and rebuilding their acceleration structures for each of the N_{envs} environments at every step. This enables all parallel environments to efficiently query the same up-to-date dynamic geometry, resulting in scalable, high-performance simulation across diverse hardware configurations.

LiDAR Model Support and Scan Pattern Simulation Leveraging this framework, we have implemented support for a wide range of commercial LiDAR models. This includes non-repetitive scanning LiDARs, such as the Livox series (e.g., Mid-360, Avia), where we simulate their characteristic scan patterns by utilizing real device data to determine scan angles synchronized with the simulation time t. Furthermore, support extends to conventional rotating LiDARs, with pre-configured models including the Velodyne series (e.g., HDL-64, VLP-32) and various Ouster sensors.

3.3 Omni-Perception Framework

Our framework, shown in Figure 2, consists of the PD-RiskNet for processing LiDAR data and a locomotion policy that integrates this perception with proprioception and commands.

3.3.1 PD-RiskNet: Processing Spatio-Temporal Point Clouds

The PD-RiskNet architecture is designed to process spatio-temporal point cloud data acquired from a legged robot's LiDAR sensor. The initial step involves partitioning the raw point cloud P_{raw}

into two distinct subsets: the proximal point cloud $P_{proximal}$ and the distal point cloud P_{distal}. This partitioning is based on a vertical angle threshold θ, distinguishing near-field points (higher θ) from far-field points (lower θ), effectively separating dense local geometry from sparse distant observations.

Proximal Point Cloud Processing: $P_{proximal}$, representing the near-field environment, is generally characterized by higher point density and relatively smaller variations in range. To efficiently process this dense data while preserving crucial local geometric details, Farthest Point Sampling (FPS)[62] is employed to obtain a sparse yet representative subset of points. Following sampling, these points are structured by sorting based on their **spherical coordinates** (θ then ϕ). This ordered representation of the sampled proximal point cloud is then input to a dedicated Gated Recurrent Unit (GRU). The feature extraction capability of this GRU is enhanced during training using a Privileged Height signal as supervision, guiding the network to learn representations specifically pertinent to local terrain properties and potential risks for legged locomotion.

Distal Point Cloud Processing: Conversely, P_{distal}, covering the far-field environment, is typically sparse and exhibits larger variations in range values. To handle the data characteristics and reduce the influence of outliers in this sparse data, Average Downsampling is applied. This operation yields a more uniform representation of distant structures. Downsampled distal point clouds from the current and preceding N_{hist} frames are combined, forming a spatio-temporal sequence. This sequence, structured by sorting points based on their spherical coordinates (θ then ϕ) after downsampling, is processed by a separate GRU module to extract features capturing the dynamic environmental context at range.

3.3.2 Risk-Aware Locomotion Policy

The locomotion policy is implemented as an MLP that takes the concatenated features from PD-RiskNet and the proprioceptive history as input.

Observation Space. As detailed in Sec. 3.1, the observation includes proprioceptive history ($o^{\text{prop}}_{t-N_{hist}+1:t}$), processed LiDAR features from PD-RiskNet (f_t^{PD}) derived from $P_{t-N_{hist}+1:t}$, and the current velocity command (c_t). Hence, the policy input is represented as $o_t^{policy} = (o^{\text{prop}}_{t-N_{hist}+1:t}, f_t^{PD}, c_t)$.

Action Space. The policy outputs target joint positions a_t for a low-level PD controller.

Reward Function. The reward function $r_t = r(s_t, a_t)$ is designed to train the robot to follow velocity commands while actively avoiding collisions by leveraging LiDAR data. We use a similar reward function as [13]. A complete list of reward functions can be found in Appendix. 5. It primarily consists of two novel components focused on velocity tracking with integrated avoidance and maximizing environmental clearance: **(1) Linear Velocity Tracking with Avoidance (r_{vel_avoid}):** This encourages tracking a modified target linear velocity that combines the external command v_t^{cmd} with a dynamically computed avoidance velocity V_t^{avoid}. The avoidance velocity V_t^{avoid} actively pushes the robot away from nearby obstacles detected by LiDAR. The robot's 360° horizontal surroundings are divided into $N_{sec} = 36$ angular sectors. Within each sector j, the minimum obstacle distance d_t^j is found. If d_t^j is below a threshold d^{thresh} (e.g., 1m), an avoidance velocity component $V_{t,j}^{avoid}$ is generated, pointing away from the sector j. Its magnitude is calculated based on proximity:

$$\|V_{t,j}^{avoid}\| = \exp(-d_{j,t} \cdot \alpha_{avoid})$$

The total avoidance velocity $V_t^{\text{avoid}} = \sum_{j=1}^{N_{sec}} V_{t,j}^{avoid}$ is the vector sum over all sectors. The reward term penalizes deviation from this combined target velocity:

$$r_{vel_avoid} = \exp(-\beta_{va} * \|v_t - (v_t^{cmd} + V_{avoid,t})\|^2)$$

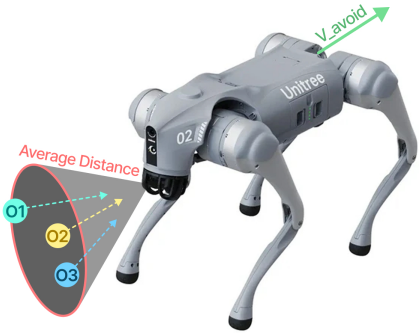


Figure 3: The calculation of the sector-based avoidance velocity.

where v_t is the base linear velocity, and β_{va} is weighting coefficients. **(2) Distance Maximization (r_{rays}):** This reward encourages maintaining a safety margin and pushing towards open areas by maximizing the capped LiDAR ray distances. Using the capped distances $\hat{d}_{t,i} = \min(d_{t,i}, d_{\max})$ from n representative distal LiDAR rays, the reward is:

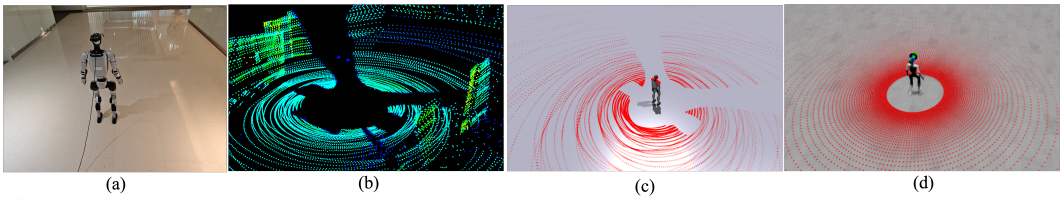
$$r_{rays} = \sum_{i=1}^{n} \frac{\hat{d}_{t,i}}{n \cdot d_{\max}}$$

Domain Randomization. This was applied during training to enhance sim-to-real transfer. This included randomizing robot physical parameters [7] and LiDAR perception characteristics (masking, noise). See Appendix 7 for details.

4 Experiments

4.1 LiDAR fidelity Evaluation

As illustrated in Figure. 4, we validate our LiDAR simulator's fidelity. We compare its output to real data from a Unitree G1 robot using a Livox Mid-360 sensor and Isaac Sim LiDAR using the same horizontal angle and vertical angle. Our simulation includes the sensor's non-repetitive scan pattern and robot self-occlusion. The simulated and real point distributions and structures appeared similar. This result indicates high simulator fidelity, which helps reduce the sim2real perception gap for LiDAR policies.You can find more lidar model scan patterns in the Appendix.**??**



(a) (b) (c) (d)

Figure 4: Comparison of simulated and real point cloud for the Unitree G1 robot. (a) The physical Unitree G1 robot setup. (b) Real-world LiDAR scan captured by the onboard Livox Mid-360 sensor. (c) **(Ours)** Point cloud generated using our Livox Mid-360 sensor model within the Isaac Gym. (d) Point cloud using the official sensor within the Isaac Sim. Ours captures the self-occlusion effect as in the real-world LiDAR.

.

Table 1: Rendering time (ms) for static scenes across configurations. Ours is much more efficient than Isaac Sim.

Lines	Number of Environments									
	256		512		1024		2048		4096	
	Ours	IsaacSim	Ours	IsaacSim	Ours	IsaacSim	Ours	IsaacSim	Ours	IsaacSim
1k	**2.5**$_{\pm 0.3}$	21$_{\pm 0.7}$	**2.1**$_{\pm 0.4}$	19$_{\pm 1.7}$	**2.2**$_{\pm 0.1}$	32$_{\pm 0.8}$	**3.4**$_{\pm 0.2}$	40$_{\pm 1.8}$	**3.3**$_{\pm 0.3}$	95$_{\pm 2.1}$
4k	**3.3**$_{\pm 0.4}$	28$_{\pm 0.6}$	**5.7**$_{\pm 0.3}$	39$_{\pm 0.3}$	**16.2**$_{\pm 1.1}$	81$_{\pm 1.1}$	**49.9**$_{\pm 2.3}$	146$_{\pm 4.8}$	**117.9**$_{\pm 0.7}$	308$_{\pm 5.8}$
16k	**3.7**$_{\pm 0.1}$	81$_{\pm 3.1}$	**11.2**$_{\pm 0.8}$	142$_{\pm 1.4}$	**30.4**$_{\pm 2.8}$	317$_{\pm 17.3}$	**118.3**$_{\pm 4.7}$	600$_{\pm 8.8}$	**220.0**$_{\pm 1.1}$	OOM
32k	**5.8**$_{\pm 0.1}$	148$_{\pm 2.3}$	**22.9**$_{\pm 2.6}$	286$_{\pm 2.7}$	**47.9**$_{\pm 0.8}$	569$_{\pm 14.9}$	**133.5**$_{\pm 8.4}$	1133$_{\pm 32.3}$	**273.0**$_{\pm 0.9}$	OOM

4.2 Comparison with Existing Simulators and Efficiency Analysis

While numerous robotics simulators exist, official support for LiDAR simulation is relatively limited, with Gazebo and NVIDIA Isaac Sim being prominent examples offering built-in capabilities. We compare our LiDAR toolkit features against these established platforms based on our evaluation and publicly available documentation. Table 2 summarizes the key LiDAR simulation capabilities across these platforms. Table 1 shows the rendering

Table 2: Comparison of Capabilities Across Platforms.

LiDAR Simulation Feature	Ours	Isaac Sim	Gazebo
Rotating LiDAR Support	✓	✓	✓
Solid-State LiDAR Support	✓	✓	✓
Hybrid Solid-State LiDAR *(Non-repetitive scan)*	✓	×	✓
Static Irregular Objects	✓	✓	✓
Dynamic Irregular Objects	✓	×	×
Self-Occlusion	✓	×	×
Cross-Platform Support	✓	×	✓
Massively Parallel Execution	✓	✓	×

speed of our liDAR implementation relative to Isaac Sim. Our implementation aims to combine broad feature support with a high-performance parallel execution environment, like Isaac Gym and Genesis.You can find more detail in the Appendix. E.

4.3 PD-RiskNet Ablation Study

We have tested four ways to process the LiDAR point cloud to see if our PD-RiskNet is effective. We compare four approaches for extracting the same dimensional point cloud features in an Isaac Gym simulation with 6 dynamic obstacles (1.5 m/s speed) over 30 trials each. As shown in Fig. 6, the task is to reach the end of a 12m x 4m area. Table. 3 shows the direct method using an MLP failed due to Out of Memory(OOM) (24GB). Methods using FPS downsampling, FPS+MLP, and FPS+GRU suffer very low success rates, mostly failing due to timeouts or getting stuck rather than colliding.

Table 3: PD-RiskNet Ablation Results (30 Trials)

Method	Success Rate (%)	Collision Rate (%)
Direct MLP	OOM*	OOM*
FPS + MLP	33.3%	93.3%
FPS + GRU	30.0%	70.0%
Ours	**76.7%**	**56.7%**

4.4 Real-World Deployment

We evaluate the omnidirectional obstacle avoidance effectiveness and its reactive behaviors in various scenarios, as shown in Fig. 1. Furthermore, we test its performance under several extreme conditions. As depicted in Fig. 5, the robot successfully crosses through rocky terrain within dense grass, even while experiencing interference from moving humans. It also demonstrates adaptability to rapidly approaching aerial obstacles and frequent human obstructions. A quantitative comparison of success rates against the native Unitree system across different obstacle types over 30 trials is presented in Table 4.

Figure 5: Robot obstacle avoidance performance was assessed in varied scenarios, including complex terrain and dynamic human interference.

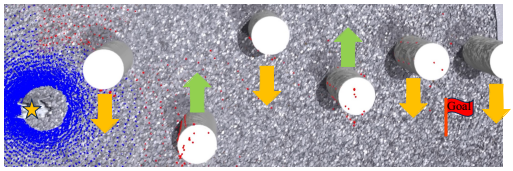


Figure 6: Simulation ablation experiment scene setting.

Scenario	Omni-Perception	Unitree System
Static obstacles	**30/30 (100%)**	30/30 (100%)
Aerial obstacles	**21/30 (70%)**	0/30 (0%)
Small obstacles	25/30 (83%)	**30/30 (100%)**
Moving humans	**27/30 (90%)**	0/30 (0%)

Table 4: Real-world performance comparison between Omni-Perception and the native Unitree system. Success rates are reported based on 30 trials for each scenario involving different types of obstacles. Bold values indicate the higher success rate for that scenario.

5 Conclusion

In conclusion, we present Omni-Perception, an end-to-end reinforcement learning framework that successfully enables robust, omnidirectional collision avoidance for legged robots by directly leveraging raw LiDAR point cloud data. Through PD-RiskNet architecture and training in a high-fidelity lidar simulation environment, our approach integrates perception and control, allowing robots to move in complex environments with diverse static and dynamic obstacles while maintaining desired velocity commands. Real-world experiments validated the system's ability to perform agile and safe locomotion in challenging, dynamic 3D settings.

6 Limitations

Environmental Geometry: Performance can degrade in environments dominated by highly unstructured vegetation, such as dense grass, where LiDAR struggles to extract reliable geometric features essential for locomotion. This poses a challenge similar to sensor limitations noted in visually complex scenes.Introducing semantic segmentation into the framework is a potential solution.

Sim2Real Fidelity Trade-off: the current approach utilizes sampling strategies within a high-fidelity simulator to facilitate sim-to-real transfer. While effective, this involves an inherent trade-off, potentially reducing the level of fine-grained geometric detail perceived by the robot compared to the raw sensor output. This simplification might become a limiting factor in scenarios demanding exceptionally precise navigation around complex small obstacles, potentially leading to suboptimal paths or collisions. Further research could focus on improving simulation fidelity, developing more advanced domain randomization or adaptation techniques, or incorporating online learning to fine-tune perception upon deployment.

6.1 Failure Case

extremely unstructured environments: Although we have successful examples in this environment, the effectiveness of the robot's locomotion strategy will be greatly reduced. The robot identified the grass on the left side of the picture as a dangerous obstacle, and because it was close, it quickly moved to the left.

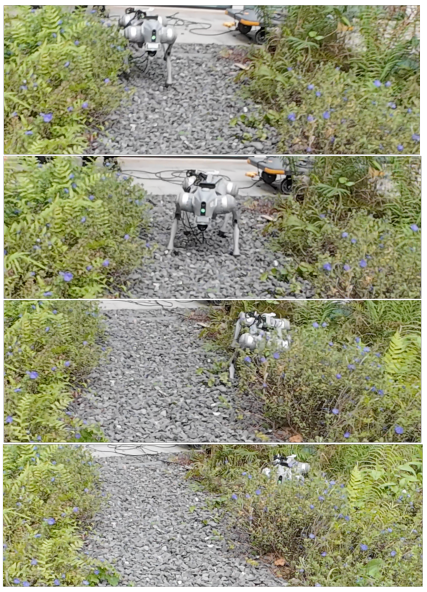


Figure 7: dense grass.

Because the entire passage was narrow, the robot was forced to enter the grass on the right side of the picture. After entering the grass, the robot's surroundings were perceived as a dangerous area, which led to mission failure.

Objects that are too small and sparse:

Since we will average the distant point clouds, the features of very small objects will be destroyed, making it impossible to respond correctly to such obstacles.

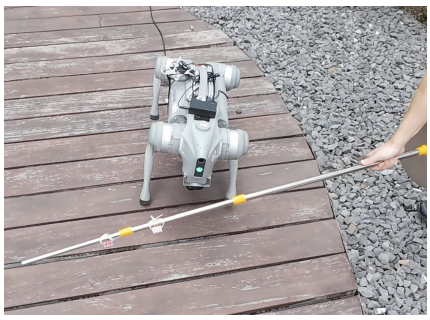


Figure 8: Thin rod.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62306257) and the Guangzhou-HKUST(GZ) Joint Funding Program (Grant No.2023A03J0008), Education Bureau of Guangzhou Municipality.

References

[1] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22 (6):46–57, 1989.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[3] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi. Agile but safe: Learning collision-free high-speed legged locomotion. In *Robotics: Science and Systems (RSS)*, 2024.

[4] Z. Xu, B. Chen, X. Zhan, Y. Xiu, C. Suzuki, and K. Shimada. A vision-based autonomous uav inspection framework for unknown tunnel construction sites with dynamic obstacles. *IEEE Robotics and Automation Letters*, 8(8):4983–4990, 2023.

[5] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim. Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2464–2470, 2020. doi: 10.1109/ICRA40945.2020.9196777.

[6] T. Dudzik, M. Chignoli, G. Bledt, B. Lim, A. Miller, D. Kim, and S. Kim. Robust autonomous navigation of a small-scale quadruped robot in real-world environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3664–3671. IEEE, 2020.

[7] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. *Conference on Robot Learning*, 2022.

[8] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. In *Robotics: Science and Systems*, 2022.

[9] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. sci. *Robotics*, 4:26, 2019.

[10] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[11] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. 2021.

[12] Z. Wang, Y. Jia, L. Shi, H. Wang, H. Zhao, X. Li, J. Zhou, J. Ma, and G. Zhou. Arm-constrained curriculum learning for loco-manipulation of a wheel-legged robot. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10770–10776. IEEE, 2024.

[13] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[14] S. Kareer, N. Yokoyama, D. Batra, S. Ha, and J. Truong. Vinl: Visual navigation and locomotion over obstacles. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2018–2024. IEEE, 2023.

[15] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023.

[16] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard. Learning predictive terrain models for legged robot locomotion. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3545–3552. IEEE, 2008.

[17] A.-C. Cheng, Y. Ji, Z. Yang, Z. Gongye, X. Zou, J. Kautz, E. Bıyık, H. Yin, S. Liu, and X. Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024.

[18] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter. Elevation mapping for locomotion and navigation using gpu. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2273–2280. IEEE, 2022.

[19] Y. Li and J. Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4): 50–61, 2020.

[20] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2020.

[21] T. Ma, Z. Wang, J. Zhou, M. Wang, and J. Liang. Glover: Generalizable open-vocabulary affordance reasoning for task-oriented grasping. *arXiv preprint arXiv:2411.12286*, 2024.

[22] T. Ma, J. Zheng, Z. Wang, Z. Gao, J. Zhou, and J. Liang. Glover++: Unleashing the potential of affordance learning from human behaviors for robotic manipulation. *arXiv preprint arXiv:2505.11865*, 2025.

[23] T. Ma, J. Zhou, Z. Wang, R. Qiu, and J. Liang. Contrastive imitation learning for language-guided multi-task robotic manipulation. *arXiv preprint arXiv:2406.09738*, 2024.

[24] J. Zhou, K. Ye, J. Liu, T. Ma, Z. Wang, R. Qiu, K.-Y. Lin, Z. Zhao, and J. Liang. Exploring the limits of vision-language-action manipulations in cross-task generalization. *arXiv preprint arXiv:2505.15660*, 2025.

[25] J. Zhou, T. Ma, K.-Y. Lin, Z. Wang, R. Qiu, and J. Liang. Mitigating the human-robot domain discrepancy in visual pre-training for robotic manipulation. *arXiv preprint arXiv:2406.14235*, 2024.

[26] Y. Jia, G. Wang, Y. Dong, J. Wu, Y. Zeng, H. Lin, Z. Wang, H. Ge, W. Gu, K. Ding, et al. Discoverse: Efficient robot simulation in complex high-fidelity environments. *arXiv preprint arXiv:2507.21981*, 2025.

[27] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter. Advances in real-world applications for legged robots. *Journal of Field Robotics*, 35(8):1311–1326, 2018.

[28] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano, et al. The current state and future outlook of rescue robotics. *Journal of Field Robotics*, 36(7):1171–1191, 2019.

[29] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced skills by learning locomotion and local navigation end-to-end. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2497–2503. IEEE, 2022.

[30] D. Wisth, M. Camurri, and M. Fallon. Vilens: Visual, inertial, lidar, and leg odometry for all-terrain legged robots. *IEEE Transactions on Robotics*, 39(1):309–326, 2022.

[31] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[32] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner. Optimization and learning for rough terrain legged locomotion. *The International Journal of Robotics Research*, 30(2):175–191, 2011.

[33] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo. Learning quadrupedal locomotion on deformable terrain. *Science Robotics*, 8(74):eade2256, 2023.

[34] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

[35] J. Lee, M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter. Learning robust autonomous navigation and locomotion for wheeled-legged robots. *Science Robotics*, 9(89): eadi9641, 2024.

[36] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

[37] Z. Zhuang, S. Yao, and H. Zhao. Humanoid parkour learning. *arXiv preprint arXiv:2406.10759*, 2024.

[38] E. Chane-Sane, J. Amigo, T. Flayols, L. Righetti, and N. Mansard. Soloparkour: Constrained reinforcement learning for visual locomotion from privileged experience. In *Conference on Robot Learning (CoRL)*, 2024.

[39] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics*, 7(62):eabk2822, 2022.

[40] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter. Learning a state representation and navigation in cluttered and dynamic environments. *IEEE Robotics and Automation Letters*, 6 (3):5081–5088, 2021.

[41] Y. F. Chen, M. Liu, M. Everett, and J. P. How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[42] P. Arm, G. Waibel, J. Preisig, T. Tuna, R. Zhou, V. Bickel, G. Ligeza, T. Miki, F. Kehl, H. Kolvenbach, et al. Scientific exploration of challenging planetary analog environments with a team of legged robots. *Science robotics*, 8(80):eade9548, 2023.

[43] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter. Navigation planning for legged robots in challenging terrain. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1184–1189. IEEE, 2016.

[44] H. Yin, X. Xu, S. Lu, X. Chen, R. Xiong, S. Shen, C. Stachniss, and Y. Wang. A survey on global lidar localization: Challenges, advances and open problems. *International Journal of Computer Vision*, 132(8):3139–3171, 2024.

[45] S. Haddadin, A. De Luca, and A. Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312, 2017.

[46] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter. Collision-free mpc for legged robots in static and dynamic scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8266–8272. IEEE, 2021.

[47] J.-W. Park, H.-D. Oh, and M.-J. Tahk. Uav collision avoidance based on geometric approach. In *2008 SICE Annual Conference*, pages 2122–2126. IEEE, 2008.

[48] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA conference on mathematics of surfaces*, volume 1, pages 602–608, 1998.

[49] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter. A collision-free mpc for whole-body dynamic locomotion and manipulation. In *2022 international conference on robotics and automation (ICRA)*, pages 4686–4693. IEEE, 2022.

[50] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos. Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles. *IEEE robotics and automation letters*, 5(4):6001–6008, 2020.

[51] Q. Liao, Z. Li, A. Thirugnanam, J. Zeng, and K. Sreenath. Walking in narrow spaces: Safety-critical locomotion control for quadrupedal robots with duality-based optimization, 2023. URL https://arxiv.org/abs/2212.14199.

[52] M. Koptev, N. Figueroa, and A. Billard. Reactive collision-free motion generation in joint space via dynamical systems and sampling-based mpc. *The International Journal of Robotics Research*, 43(13):2049–2069, 2024.

[53] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada. Navrl: Learning safe flight in dynamic environments. *IEEE Robotics and Automation Letters*, 10(4):3668–3675, 2025. doi:10.1109/LRA.2025.3546069.

[54] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza. Learning perception-aware agile flight in cluttered environments, 2022. URL https://arxiv.org/abs/2210.01841.

[55] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[56] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi:10.1109/IROS.2004.1389727.

[57] G. Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

[58] M. Kulkarni, W. Rehberg, and K. Alexis. Aerial gym simulator: A framework for highly parallelized simulation of aerial robots. *IEEE Robotics and Automation Letters*, 2025.

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

[60] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019.

[61] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha. Fast bvh construction on gpus. In *Computer Graphics Forum*, volume 28, pages 375–384. Wiley Online Library, 2009.

[62] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[63] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

A Implementation Details

A.1 Rewards

Term	Equation	Weight
Omni-Perception Rewards		
Velocity Tracking with Avoidance ($r_{vel,avoid}$)	$\exp\{-\beta_{va}\|\|\mathbf{v}_t - (\mathbf{v}_t^{\text{cmd}} + \mathbf{V}_{\text{avoid},t})\|\|^2\}$	*2*
Distance Maximization (r_{rays})	$\sum_{i=1}^{n} \frac{\min(d_{t,i}, d_{\max})}{n \cdot d_{\max}}$	*1.5*
Auxiliary Rewards		
z velocity	v_z^2	-3×10^{-4}
foot stumble	$\|\|Force_{xy}^{\text{foot}}\|\|^2$	-2×10^{-2}
link collision	$\|\|Force_{xy}^{\text{PenltyLink}}\|\|^2$	-0.02
joint limit violation	$\mathbf{1}_{q_i > q_{\max} \| \| q_i < q_{\min}}$	-0.2
joint torques	$\|\|\boldsymbol{\tau}\|\|^2$	-1×10^{-6}
joint velocities	$\|\|\dot{\mathbf{q}}\|\|^2$	-1×10^{-6}
joint accelerations	$\|\|\ddot{\mathbf{q}}\|\|^2$	-2.5×10^{-7}
action smoothing	$\|\|\mathbf{a}_{t-1} - \mathbf{a}_t\|\|^2$	-5×10^{-3}
action smoothing rate	$\|\|\mathbf{a}_{t-2} - 2\mathbf{a}_{t-1} + \mathbf{a}_t\|\|^2$	-5×10^{-3}

Table 5: Reward structure for Omni-Perception

B Network Architecture Details

The Omni-Perception framework utilizes specific neural network architectures for perception (PD-RiskNet) and control (Actor).

B.1 PD-RiskNet Architecture

The PD-RiskNet processes spatio-temporal LiDAR point cloud data. As described in Section 3.3.1 of the main text, the raw point cloud is partitioned into proximal (P_{proximal}) and distal (P_{distal}) subsets.

- **Input Processing:** Both the proximal and distal pathways process a history of point cloud data. The paper mentions using a history of N_{hist} frames (Sec 3.1, Sec 3.3.1). Based on additional details provided, we use $N_{\text{hist}} = 10$, meaning each Gated Recurrent Unit (GRU) processes features derived from 10 consecutive LiDAR scans.
- **Proximal Pathway:** The proximal point cloud (P_{proximal}) undergoes Farthest Point Sampling (FPS) before being fed into a dedicated GRU. This GRU is supervised using privileged height information during training. The output embedding dimension from the proximal GRU is **187 features**.
- **Distal Pathway:** The distal point cloud (P_{distal}) is processed using Average Downsampling. Features from the current and $N_{\text{hist}} - 1$ preceding frames (forming a sequence of 10 frames) are fed into a separate GRU. The output embedding dimension from the distal GRU is **64 features**.

The embeddings from both the proximal and distal GRUs are concatenated with proprioceptive history and the command vector before being passed to the Actor network.

B.2 Actor Network Architecture

The locomotion policy (Actor) is implemented as a Multi-Layer Perceptron (MLP).

- **Input:** Concatenated features from PD-RiskNet (187 + 64 features), processed proprioceptive history, and the current velocity command.

- **Hidden Layers:** The MLP consists of sequential fully connected layers with the following hidden dimensions: **[1024, 512, 256, 128]**. Appropriate activation functions ELU are typically used between layers.
- **Output Layer:** The final layer outputs the target joint positions for the robot's actuators. The output dimension is **12**.

C PPO Hyperparameters

The policy was trained using the Proximal Policy Optimization (PPO) algorithm [63]. Key hyperparameters used during training are listed in Table 6.

Table 6: PPO Hyperparameters

Parameter	Value
PPO clip parameter (ϵ)	0.2
GAE λ	0.95
Reward discount factor (γ)	0.99
Learning rate	1×10^{-3}
Learning rate schedule	adaptive
Value loss coefficient	1.0
Use clipped value loss	True
Entropy coefficient	0.01
Desired KL divergence	0.01
Max gradient norm	1.0
Number of environments	4096
Number of env steps per batch	24
Learning epochs per batch	5
Number of mini-batches per epoch	4

D Domain Randomization Details

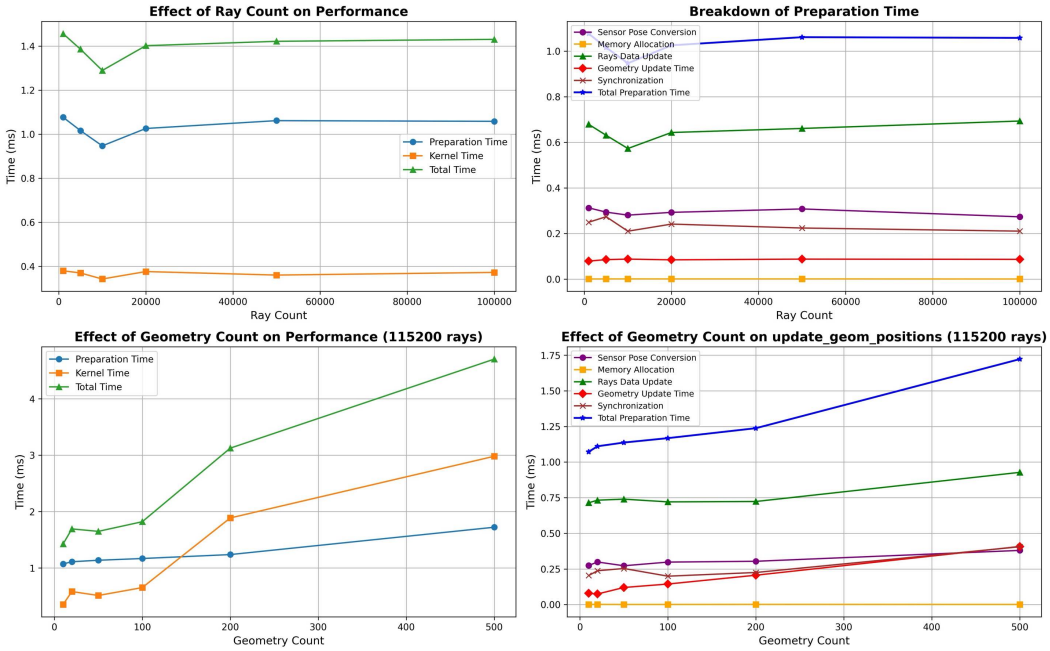
To improve sim-to-real transfer, we applied domain randomization to various simulation parameters during training. We followed previous work [7] for randomizing the robot's physical attributes. For LiDAR perception, we introduced specific randomizations: random masks were applied to 10% of the point cloud, assigning these points small distance values uniformly sampled from [0, 0.3]. Additionally, 10% noise was randomly added to the measured LiDAR distances.

The specific parameters and their randomization ranges are detailed in Table 7. All attributes listed were uniformly sampled across all 4096 parallel environments during reinforcement learning.

Table 7: Parameters and Ranges for Domain Randomization

Parameter	Range	
	Min	Max
LiDAR Point Masking Ratio	10% (Values $\in [0, 0.3]$)	
LiDAR Distance Noise Ratio	10%	
Added Mass (kg)	-1.0	5.0
Payload Mass (kg)	-1.0	3.0
Center of Mass x (m)	-0.1	0.1
Center of Mass y (m)	-0.15	0.15
Center of Mass z (m)	-0.2	0.2
Ground Friction Coefficient	0.40	1.00
Ground Restitution	0.00	1.00
Motor Strength (Scale Factor)	0.8	1.2
Joint Calibration Offset (rad)	-0.02	0.02
Gravity Offset (m/s^2)	-1.0	1.0
Proprioception Latency (s)	0.005	0.045

E Taichi Lidar Efficiency

We tested the performance of Taichi Version on three different computers, including a MacBook.Our program is cross-platform like MuJoCo,Genesis,Gazebo,Isaac sim/gym.

In scenes with fewer geoms (lower 200), simulating with 115,200 rays can achieve 500Hz+ simulation efficiency, which is really fast! Most of the time is spent in the preparation process, with a large proportion (more than60%)