INVESTIGATING ROBOT LEARNING OF QUADRUPEDAL LOCOMOTION ON DEFORMABLE TERRAIN

A Thesis approved by the Faculty of Mechanical Engineering at Rheinisch-Westfälischen Technischen Hochschule Aachen to obtain the Master of Science Degree in Robotic Systems Engineering

Submitted by

Mohammed Azharudeeen Farook Deen

Matr.no: 429189

This present work was submitted to Institute for Data Science in Mechanical Engineering(DSME) and to the Chair of Geotechnical Engineering(GUT).

Univ.-Prof. Dr. Raul Fuentes (GUT)
Dr. Ömer Kemal Adak (GUT)
Univ.-Prof. Dr. Sebastian Trimpe (DSME)
Emma Cramer (DSME)

Abstract

Animals exhibit an extraordinary capability for precise and agile locomotion across a diverse array of natural terrains. Replicating this remarkable adaptability and efficiency in robotic systems, particularly quadrupedal robots, presents a formidable challenge that sits at the forefront of robotics research. The overarching objective of our study is to develop and refine a fast, efficient, and robust controller capable of guiding quadruped robots through a wide range of natural environments. These environments encompass a variety of challenging conditions, including uneven terrains, steep inclinations, slippery surfaces, and deformable substrates such as soft soil, loose gravel, sand, dust, water, and thick vegetation. To achieve this, our research delves into different simulation platforms to simulate the intricacies of realistic and complex deformable terrains that closely mimic the natural world. A significant part of our effort focuses on evaluating and implementing a Position-Based Dynamics (PBD) model within Isaac Sim, a cutting-edge simulation platform. This model offers a promising approach to accurately simulate the physical interactions between the quadruped robots and deformable terrain, offering insights into the mechanics of locomotion on such surfaces. Through the application of reinforcement learning (RL), we train a sophisticated controller capable to instill the quadruped with the ability to follow a unified policy across a spectrum of challenging conditions. We culminate with rigorous testing and optimization to validate the effectiveness of our developed control system, emphasizing its robustness and adaptability.

Zusammenfassung

Tiere verfügen über eine außergewöhnliche Fähigkeit zur präzisen und wendigen Fortbewegung in einer Vielzahl von natürlichen Umgebungen. Diese bemerkenswerte Anpassungsfähigkeit und Effizienz in Robotersystemen, insbesondere in vierbeinigen Robotern, nachzubilden, stellt eine große Herausforderung dar, die an der Spitze der Robotikforschung steht. Das übergeordnete Ziel unserer Studie ist die Entwicklung und Verfeinerung eines schnellen, effizienten und robusten Controllers, der in der Lage ist, vierbeinige Roboter durch ein breites Spektrum an natürlichen Umgebungen zu führen. Diese Umgebungen umfassen eine Vielzahl anspruchsvoller Bedingungen, darunter unebenes Gelände, starke Steigungen, rutschige Oberflächen und verformbare Substrate wie weicher Boden, loser Kies, Sand, Staub, Wasser und dichte Vegetation. Um dies zu erreichen, befasst sich unsere Forschung mit verschiedenen Simulationsplattformen, um die Feinheiten realistischer und komplexer verformbarer Terrains zu simulieren, die der natürlichen Welt sehr nahe kommen. Ein wesentlicher Teil unserer Bemühungen konzentriert sich auf die Evaluierung und Implementierung eines PBD-Modells (Position-Based Dynamics) in Isaac Sim, einer hochmodernen Simulationsplattform. Dieses Modell bietet einen vielversprechenden Ansatz, um die physikalischen Interaktionen zwischen den vierbeinigen Robotern und dem deformierbaren Terrain genau zu simulieren und Einblicke in die Mechanik der Fortbewegung auf solchen Oberflächen zu geben. Durch die Anwendung von Verstärkungslernen (Reinforcement Learning, RL) trainieren wir einen hochentwickelten Controller, der dem Vierbeiner die Fähigkeit verleiht, eine einheitliche Strategie über ein Spektrum von schwierigen Bedingungen hinweg zu verfolgen. Wir schließen mit rigorosen Tests und Optimierungen ab, um die Effektivität des von uns entwickelten Steuerungssystems zu validieren und seine Robustheit und Anpassungsfähigkeit zu betonen.

Statutory Declaration In Lieu Of An Oath

I, Mohammed Azharudeen Farook Deen, hereby declare in lieu of an oath that I have completed the present Master thesis entitled “Investigating Robot Learning of Quadrupedal Locomotion on Deformable Terrain” independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar form.

F. Azhar

Aachen, 14.03.2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The marvel of quadrupedal animals' locomotion - their agility, endurance, and speed - has long captivated human imagination. From the swift chase of a cheetah to the graceful leap of a deer, these natural athletes embody a level of physical prowess that has inspired scientists and engineers alike. In understanding the secrets behind the remarkable capabilities of these creatures, we not only unlock new avenues in biological sciences but also pave the way for revolutionary advancements in robotics. This project is motivated by the desire to bridge the gap between biological locomotion and robotic capability, particularly in the context of navigating complex, deformable terrains that pose a significant challenge for current robotic systems.

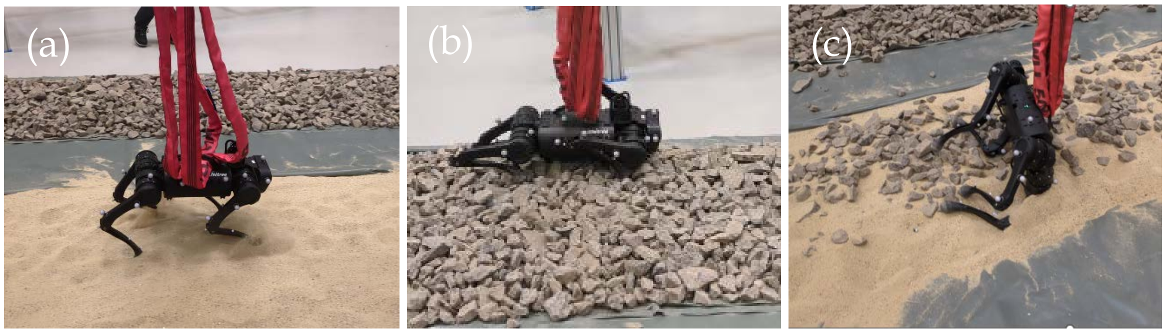

Figure 1.1: The Unitree A1 robot on various terrains: (a) successful traversal on sand, (b) failure on gravel, and (c) failure on a gravel-sand mixture.

Our motivation is further reinforced by preliminary experiments conducted with the Unitree A1 robot, as depicted in Fig 1.1. These trials highlighted the robot's difficulties in traversing terrains composed of gravel and sand, underscoring the need for enhanced locomotion strategies.

1.2 Objectives and Overview

This thesis endeavors to address the significant challenges faced by quadruped robots when traversing on deformable terrains such as sand, gravel, and mud. These challenges stem from the unpredictable and highly variable nature of deformable terrains, which can dramatically affect the stability, mobility, and energy efficiency of robotic locomotion. The objectives of this research are to:

- To identify and understand the challenges quadrupeds face when transitioning from rigid to deformable terrains. This involves analyzing the impact of terrain deformability on locomotion dynamics, control complexity, and energy utilization.
- To create a detailed simulation environment that can accurately model the interaction between the quadruped robots and a variety of deformable terrains. This includes integrating advanced physical modeling techniques to simulate the complex behaviors of granular materials.
- To develop adaptable and robust control strategies that must account for the unpredictability of substrate responses and aim to maintain stability and traction in the face of shifting and sinking surfaces.
- To assess the performance of these control strategies in simulated environments to varying terrain conditions.

1.3 Contribution

The primary contribution of this thesis lies in the successful implementation of a practical and effective simulation environment utilizing Position-Based Dynamics (PBD) within Isaac Sim. This environment was specifically tailored for the development and evaluation of RL algorithms aimed at optimizing quadrupedal locomotion on deformable terrains. The adoption of PBD allowed for a more accurate and computationally efficient simulation of granular interactions, facilitating the real-time training, and testing of RL policies. Our investigation spans various simulation software like Isaac Gym, Isaac Sim and NVIDIA Warp and methodologies including Position-Based Dynamics (PBD), Discrete Element Method (DEM), and Finite Element Method (FEM), to model the intricate interactions between robots and granular materials like sand and gravel. These exploratory phases were crucial for understanding the limitations and advantages of different simulation environments in capturing the complex dynamics of deformable terrains.

In addition to the development of the simulation environment, we designed a comprehensive RL framework that employs accurate input representations and policy architectures, utilizes terrain generation and domain randomization techniques to expose the learning algorithms to a wide range of environmental conditions. It integrates a tailored reward structure that encourages stable and adaptable locomotion strategies suitable for navigating both rigid and deformable terrains.

By blending advanced simulation techniques with cutting-edge learning algorithms, this thesis aims to lay the groundwork for future advancements in robotic locomotion, enabling robots to go where no robot has gone before.

Chapter 2

Background

2.1 Quadruped Robot Locomotion – Overview

The quest to emulate the locomotion of quadruped animals has been a significant area of research in robotics. Quadruped robot locomotion draws inspiration from the natural world, where animals exhibit a remarkable range of movements to navigate diverse and challenging terrains. The study of legged locomotion dynamics has been pivotal in understanding the mechanical principles governing movement and stability.

Traditional methods for navigating rough terrain with legged robots often involve complex and intricate control systems[1], [2], [3], [4], [5]. These methods generally use detailed state machines to manage various movements and reactions, relying on specific estimates like when a foot makes contact with the ground or slips[6], [7]. However, these estimates can be unreliable when dealing with unpredictable elements like mud or snow[8], [9], [10]. Some designs also use sensors on the robot's feet for direct contact detection, but these too can fail in challenging outdoor conditions. As a result, these traditional approaches become increasingly complex and hard to manage, especially as they try to accommodate more varied scenarios, making them difficult to develop and maintain while also being prone to failure in unexpected situations.

Also, a significant portion of Earth's terrestrial environment comprises these non-rigid terrains, presenting a compelling argument for the necessity of adaptable locomotion strategies in legged robots. Recent advancements have led to the development of more versatile quadruped robots capable of traversing a wide range of terrains[11], [12], [13], [14], from rigid surfaces to challenging, deformable terrains like mud and snow[11], [15]. The inherent advantages of legged over wheeled mobility, particularly in handling sinkage and slippage[16], have been demonstrated through advancements in robotic design, enabling traversal across a broader spectrum of terrains. However, the challenge of accurately predicting terrain dynamics[17], [18] due to varying compliance and deformability has limited extensive application outside controlled laboratory settings[19], [20]. These studies lay the foundation for developing control algorithms that can mimic the efficiency and adaptability observed in natural locomotion. However, the dynamic interaction with non-rigid surfaces remains a complex challenge, as traditional control strategies primarily designed for rigid terrains often fail when confronted with the unpredictable nature of deformable terrains. Consequently, modern research efforts to incorporate terrain models into legged robotic control have largely focused on robots with basic morphology, such as the one-dimensional (1D) hopper[19], [21], [22], due to the complexity of more intricate designs.

2.2 Reinforcement Learning for Locomotion

Deep Reinforcement Learning (DRL) has emerged as a powerful tool for automating the learning process of complex locomotion tasks[23], [24], [25]. This approach has streamlined controller design, automated design processes, and enabled the learning of previously un-engineered behaviors[23], [24], [25], [26]. By interacting with a simulated environment, an RL agent iteratively improves its policy, aiming to maximize a cumulative reward signal. Recent advancements in DRL enable robots to learn effective locomotion strategies through trial and error, progressively optimizing their behavior based on experience.

The application of RL in quadruped locomotion has seen substantial interest due to its potential to discover efficient and novel movement strategies without explicit programming. Previous works[11], [12], [13], [15], [23], [24], [27], [28] demonstrate that RL can enable robots to learn locomotion skills that are robust across a variety of surfaces, including those not encountered during training. Agarwal et al.[29] have explored how egocentric vision can inform foot placement and terrain

interaction. These studies underscore the value of RL in automating the design of complex locomotive behaviors that would be challenging to craft manually.
The control strategies for such robots have evolved from simple, statically stable walking patterns to dynamic, agile maneuvers that leverage the robot's natural dynamics. Researchers like Hutter et al. and Lee et al. [11], [15], [23], [30] have showcased quadruped robots traversing steep slopes and rugged terrains, highlighting the importance of sophisticated control algorithms that can adapt to changing environmental conditions. Curriculum learning[31], [32], [33], [34] simplifies training by starting with simple tasks and gradually increasing complexity, a technique previously utilized in robotic systems training. Control strategies for enabling terrain adaptation in legged robots have evolved from rigid models to more flexible, learning-based approaches.
A significant challenge in applying RL to legged locomotion is bridging the gap between simulation and reality, as discrepancies between simulated and real-world conditions can lead to degraded performance outside the laboratory[23], [24], [25]. Strategies such as dynamics randomization[35] and domain adaptation[28], [36] in the form of privileged learning frameworks[37] have been explored to enhance the transferability of learned policies. Privileged Learning involves training a "teacher" policy with access to simulation-only states, from which a "student" policy learns to mimic the teacher’s performance without requiring such privileged information. Domain Randomization extends the diversity of training scenarios, enhancing the robot's ability to adapt to new and varied real-world environments.

2.3 Simulation Environment

A crucial factor for successful DRL application in this domain is the availability of high-fidelity simulators. Simulations play a pivotal role in the development and testing of locomotion strategies, offering a safe and controllable environment for iterative experimentation. Over the years, a myriad of simulation tools has been developed, each with its own set of features, physics engines, and levels of realism.

Leading simulators like MuJoCo[38] (Multi-Joint dynamics with Contact) and Gazebo have been widely used in the robotics community for their robust physics simulation capabilities and flexibility in designing complex environments. MuJoCo, known for its speed and accuracy in simulating the dynamics of multi-jointed robots with contact dynamics, has been instrumental in advancing research in legged locomotion and manipulation. PyBullet[39], an open-source robotics simulator which has gained popularity for its ease of use and integration with Python, provides realistic physics simulation and graphical visualization. These tools have been particularly useful for researchers focusing on complex manipulation tasks and locomotion challenges in dynamic environments.

More recently, NVIDIA's Isaac Gym[40] has emerged as a powerful tool for training robotic systems using DRL. It leverages GPU acceleration to simulate thousands of environments in parallel, dramatically speeding up the training process of DRL algorithms. This capability allows for the exploration of vast parameter spaces and the training of more sophisticated policies that can generalize across a wider range of scenarios. Isaac Gym's integration with NVIDIA's PhysX engine ensures high-fidelity physical simulations, crucial for tasks requiring precise interaction with the environment.

Accurate simulation of deformable terrains poses significant computational challenges, with granular media (GM) simulation being a key area of focus. Despite advancements, the simulation of deformable terrains poses a substantial challenge, given the complexity and variability of real-world surfaces. Several challenges lie at the forefront of this research domain. One is the high computational cost associated with realistic deformable terrain simulation. Another is the development of techniques for effective terrain characterization – enabling the robot to quickly identify properties of the ground it's traversing (e.g., softness, slipperiness) and adapt its locomotion accordingly. Additionally, ensuring safe training and deployment of RL-based controllers is critical, as these algorithms may explore unsafe actions during their learning process.

Kolvenbach et al.'s work[16] on navigating steep and granular Martian analog slopes with a dynamic quadrupedal robot marks a significant milestone in the field, demonstrating the potential of advanced simulation tools to prepare robots for extraterrestrial exploration. The discrete element method[41], [42] (DEM), has been identified as a promising approach for modeling the interactions between robotic limbs and granular media, offering insights into the mechanics of locomotion on soft grounds. However, the computational intensity of DEM limits its applicability in real-time or large-scale simulations, necessitating the exploration of more efficient simulation methodologies for RL-based training. To address this gap, we chose NVIDIA's Isaac Sim for its support of Position-Based Dynamics[43], [44] (PBD), which provides a viable platform for simulating complex interactions with deformable terrains within a RL framework. By leveraging Isaac Sim’s PBD capabilities, our thesis aims to create a simulation environment that closely mimics the challenges of navigating deformable terrains, providing a valuable platform for developing and testing the control policies of quadruped robots. This endeavor not only advances our understanding of robot-environment interactions but also moves us closer to realizing the deployment of autonomous quadrupeds in real-world applications ranging from search and rescue missions to planetary exploration.

Chapter 3

Methods

This chapter delineates the comprehensive methodology employed in developing and training a RL policy for a quadruped robot navigating deformable terrain. The approach begins with training in Phase 1 on rigid terrains, utilizing modifications from a benchmark model. Building upon the foundational skills acquired, the training progresses to Phase 2, where the robots are exposed to dynamic, particle-enriched terrains. This section details the objectives, strategies, and outcomes associated with each phase.

3.1 Quadruped Dynamics

The notion of this section is to elucidate the mathematical foundation underlying quadrupedal dynamics, focusing on their representation through generalized coordinates and the application of Euler-Lagrange equations in describing system behaviors.

To encapsulate the motion of quadruped robots, especially when dealing with floating base systems, we employ generalized coordinates. This approach facilitates a comprehensive representation of the system's kinematics and dynamics. The generalized coordinate vector q is partitioned into two segments: q_b and q_j, representing the unactuated floating base coordinates and the actuated joint coordinates, respectively. Formally, q is defined as:

$$q = \begin{bmatrix} q_b & q_j \end{bmatrix}$$

Where $q_b=[x, y, z, \varphi, \Theta, \psi]^T$ delineates the position and orientation of the robot's center of mass (CoM) with respect to the inertial frame, employing Euler angles (roll ϕ, pitch Θ, and yaw ψ) for base orientation representation.

The equations of motion for the quadruped system are derived from the Euler-Lagrange equation, incorporating the effects of contact forces. It can be expressed as:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = S^T\tau + J_c(q)^T F_c$$

where M(q) is the inertia matrix, $C(q,\dot{q})$ represents Coriolis and centrifugal forces, $G(q)$ is the gravitational force vector, F_c denotes the vector of contact forces, J_c (q) is the contact Jacobian, and S is a selection matrix isolating actuated joints. The system's dynamics hinge on these variables, alongside the gravitational, Coriolis, and contact forces.

The contact forces, crucial for interaction with the environment, are calculated as follows:

$$F_c = (J_c M^{-1} J_c^T)^{-1} \left(J_c M^{-1}(C + G - S^T\tau) - \dot{J_c}\dot{q}\right)$$

This formulation facilitates the estimation of contact forces without direct force measurement, leveraging system dynamics.

3.2 Reinforcement Learning Framework

Reinforcement Learning (RL) is a machine learning paradigm that enables an agent to learn optimal behaviors through interaction with an environment. As shown in Fig.3.1, the process hinges on the agent's ability to evaluate its actions based on feedback, encapsulated as rewards or penalties, thus guiding it towards optimal behavior.

The mathematical foundation of RL is the Markov Decision Process (MDP), characterized by a continuous state space S, a continuous action space A, a state transition function T(s,a,s'), and a reward function R(s,a). Here, T(s,a,s') denotes the probability density of transitioning to the next state s' upon taking an action a in the current state s, while R(s,a) specifies the immediate reward for taking action a in state s.

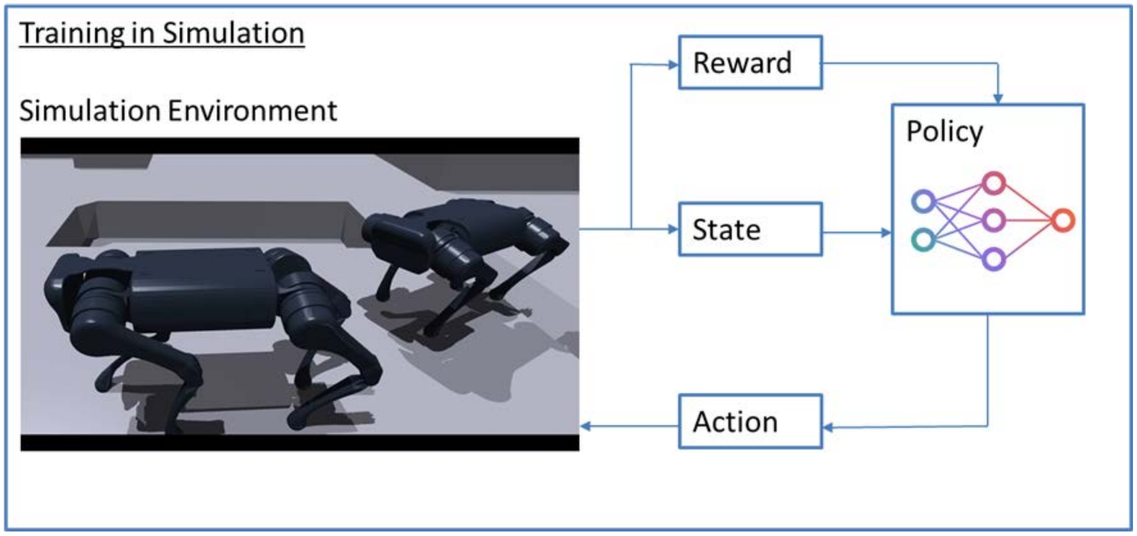

Figure 3.1: Diagram illustrating the cyclical process of reinforcement learning process, including agent, environment, actions, rewards, and policy.

A policy Π(a ∣ s) is a strategic plan that maps states to actions. The objective in RL is to discover a policy that maximizes the expected return, defined as the sum of discounted rewards over a horizon T, mathematically expressed as:

$$J(\pi) = E\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right]$$

where $\gamma \in [0,1]$ is the discount factor moderating the importance of future rewards.

3.2.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO) represents a class of policy gradient algorithms that iteratively improve the policy by estimating the gradient of the expected return with respect to the policy parameters. PPO is distinguished by its use of a clipped surrogate objective function to mitigate excessively large policy updates. The modified objective incorporates clipped probability ratios, forming a conservative estimate of the policy's advantage:

$$L(\theta) = \widehat{E_t}\left[\min\left(r_t(\theta)\widehat{A_t}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}\right)\right]$$

where ϵ is a hyperparameter dictating the clipping range, $r_t(\theta) = \frac{\pi_{\theta_{old}}(a_t|s_t)}{\pi_\theta(a_t|s_t)}$

represents the probability ratio reflecting the disparity between the current and preceding policy, and $\widehat{A_t}$ is an estimator of the advantage function at time t. This formulation encourages moderate adjustments to the policy, fostering stable and effective learning.

3.2.2 Input Representation

The RL model relies on a rich set of inputs to accurately predict the optimal actions for the quadruped robot. Collectively, these components of the input representation form a comprehensive state space that captures both the internal state of the robot and its interaction with the environment. This rich input enables the RL policy to make nuanced decisions that optimize the robot's locomotion in diverse and challenging terrains.

Base Linear Velocities: This component of the state space represents the velocity of the robot's base in three dimensions: forward/backward, left/right, and up/down.

Base Angular Velocities: These velocities indicate the rate of rotation of the robot's base around three perpendicular axes: roll, pitch, and yaw.

Measurement of the Gravity Vector: The gravity vector measurements in the robot's body frame provide information about the orientation of the robot relative to the gravitational force. This helps in distinguishing between the robot's inclination and the slope of the terrain it traverses, aiding in stability and posture adjustment strategies.

Command Vector: The command vector includes desired directions or velocities that the robot should follow, usually provided by a higher-level controller. It represents a target velocity in the forward direction x, lateral movement y, and a rotational velocity around the vertical z axis(yaw), guiding the RL policy towards achieving specific locomotion objectives.

Joint Position: This refers to the current angular positions of each of the robot's twelve joints (assuming three joints per leg for a quadruped)

Joint Velocity: Joint velocities provide the rate of change of the angular positions of the robot's joints.

Previous Actions: Including the history of previous actions allows the policy to incorporate temporal dependencies into its decision-making process. It helps in smoothing the control commands over time and avoiding abrupt changes that could destabilize the robot or result in inefficient movements.

Measurements of the Terrain Sampled from a Grid Around the Robot's Base: This aspect of the input representation consists of a set of distance measurements from the terrain surface to the robot's base, sampled from a grid surrounding the robot. These measurements provide detailed information about the terrain's geometry near the robot, enabling the policy to adapt the robot's movements to various terrain features.

3.2.3 Action Space

The action space in RL defines the set of all possible actions that the learning agent can take in response to the current state. For our quadruped robot with three joints per leg, the action space consists of twelve values of desired joint position targets for each of the robot's twelve joints, which the robot's control system strives to achieve. These targets are not absolute joint angles but are intended to be offsets from a nominal or 'default' joint configuration, which represents the robot's standard posture for efficient movement. The desired joint position targets are then scaled and applied within the robot's control loop to determine the actual motor commands issued to the joints.

The control loop employs a Proportional-Derivative (PD) controller, where the proportional term (denoted by K_p) ensures that the motor command is proportional to the difference between the current joint position and the desired target position, accounting for a scaling factor. The derivative term (denoted by K_d) counters the joint velocity, effectively damping the motion and preventing oscillations that could lead to instability. The resulting torques are clipped to a maximum and minimum value of 80.0 Nm for Anymal Robot, preventing the application of excessive torques that could damage the robot's actuators or cause unstable behavior.

3.2.4 Policy Architecture

The policy architecture encapsulates the computational framework and design of the neural network used by the agent to learn and make decisions. For the RL agent used in this study, the A2C (Advantage Actor-Critic) algorithm was implemented with continuous action spaces. The architecture is designed to separate the policy (actor) and value (critic) functions, each parameterized by a neural network that learns to predict the most beneficial actions and estimate future rewards, respectively. For the action space configuration, the architecture specifies a continuous output with no activation function applied to the mean and standard deviation. The standard deviation is initialized to zero and fixed, implying a deterministic policy at initialization which evolves during training.

The network is structured as a Multi-Layer Perceptron with three hidden layers containing 512, 256, and 128 units respectively. The activation function for all layers is the Exponential Linear Unit (ELU), chosen for its ability to handle the vanishing gradient problem.

3.2.5 Training Configuration

The agent is trained utilizing a Proximal Policy Optimization (PPO) approach. Input normalization is applied across observations, values, and advantages to stabilize training, which is a common practice to aid in the learning process. Other significant hyperparameters include:

Discount factor: for future rewards, set to 0.99, which balances the immediate and future reward trade-off.

Smoothing coefficient: for the advantage calculation, set to 0.95.

Entropy Coefficient: a small value of 0.001, which encourages exploration by penalizing certainty in action selection.

Learning Rate: An adaptive learning rate beginning at 3.e-4, with adjustments based on the Kullback-Leibler (KL)-divergence threshold of 0.008.

Gradient Clipping: The model applies gradient clipping with a norm threshold of 1 to prevent the exploding gradient problem.

Minibatch and Epoch Configuration: The model uses large minibatches of size 16384, over a horizon length of 24 steps, each minibatch is processed for 5 mini-epochs, with a batch size $\approx$ 50,000.

Max Epochs and Evaluation: The learning process is set to a maximum of 2000 epochs, with a performance evaluation criterion (score to win) of 20000.

Hardware Utilization: The experiment is configured to run on specified devices (cpu, gpu), ensuring optimal hardware utilization for training efficiency. For rigid terrains, leveraging the GPU pipeline is recommended for its performance benefits. However, due to the current limitations in Isaac Sim's support for particle simulation views on the GPU pipeline, it is necessary to switch to the CPU pipeline during phase 2 involving particles. This transition ensures that the particles behave as expected within the physics scene, moving, and interacting in response to the robot's actions.

3.2.6 Rewards

The reward function is the guiding force in RL, providing a measure of success for the agent's actions at every time step. The benchmark model's reward function, derived from the foundational paper, consists of a weighted sum of multiple terms, each reflecting a different aspect of the desired behavior:

Linear and Angular Velocity Tracking:

- **Linear Velocity Tracking**: The robot is rewarded for the accuracy with which it matches the target base velocities in the horizontal xy-plane.
- **Angular Velocity Tracking**: Similarly, the robot is rewarded for tracking the target angular velocity around the z-axis, incentivizing the agent to maintain a desired rotational motion.

Penalties for Undesired Velocities and Joint Dynamics:

To discourage undesired behaviors, penalties are included in the reward function:

- **Linear Velocity Penalty**: To penalize any undesired base velocity along the vertical (z) axis to discourage jumping or sinking motions.
- **Angular Velocity Penalty**: Penalize any undesired base angular velocities in the horizontal plane (x,y) to maintain directional stability
- **Joint Motion**: Encouraging smooth and controlled joint movements, penalties are assigned for excessive joint accelerations and velocities.
- **Joint Torques**: To promote energy efficiency and prevent mechanical stress, the reward function penalizes large joint torques.
- **Action Rate**: The policy is penalized for abrupt changes in the desired joint positions, fostering smooth transitions between actions.

To ensure safe interaction with the environment and promote agility:

- **Collisions**: A negative reward is given for each collision, urging the agent to avoid unsafe contact with obstacles.
- **Feet Airtime**: Longer steps are generally more visually appealing and potentially more effective. The reward function thus includes a term that rewards the agent when the feet are in the air, past a certain threshold duration.

Weights and Time Scaling: Each of these reward and penalty terms is scaled by a predefined weight and multiplied by the physics scene time step dt, which normalizes them to maintain consistency across different simulation step sizes.

Reward Terms	Definition	Weight
Linear Velocity Tracking	$\emptyset\left(v_{b,xy}^{*} - v_{b,xy}\right)$	1 dt
Angular velocity Tracking	$\emptyset\left(\omega_{b,xy}^{*} - \omega_{b,xy}\right)$	0.5 dt
Linear Velocity penalty	$-v_{b,z}^{2}$	4 dt
Angular velocity Penalty	$-\left\|\omega_{b,xy}\right\|^{2}$	0.05 dt

Joint motion	$-\|\ddot{q}_j\|^2 - \|\dot{q}_j\|^2$	0.001 dt
Joint torques	$-\|\tau_j\|^2$	0.00002 dt
Action rate	$-\|q_j^*\|^2$	0.25 dt
Collisions	$-n_{collision}$	0.001 dt
Feet airtime	$\sum_{f=0}^{4}(t_{\mathrm{air},f} - 0.5)$	2 dt

Table 3.1: Definition of reward terms, for benchmark model with $\phi(x) := \exp(-4. |x|^2)$

Adaptations for Phase 1 and 2 Training

Transitioning to Phase 1 and 2 of our training, which focuses on navigating granular terrains, required us to revisit and adapt the reward function to better suit the challenges inherent to these environments. We identified the necessity to refine the reward function by the addition of two reward terms to the stumble and feet contact forces, alongside the decision to exclude the airtime reward under certain training conditions.

1. **Stumble Prevention Reward**: Navigating granular terrains, such as sand or gravel, presents unique challenges for quadruped robots, including the risk of stumbling due to uneven or shifting surfaces. To address this, we incorporated a stumble prevention reward[45], which penalizes the robot for instances where the lateral forces on its feet exceed a predefined stumbling threshold force, without sufficient vertical force to counterbalance. The stumble condition is true when the norm of lateral forces on the robot's feet surpasses the stumbling threshold, and the vertical forces are below the vertical force threshold.

2. **Peak Contact Forces Reward**: Another aspect critical to terrain navigation is managing the forces exerted by and on the robot's feet upon contact with the ground. In our case, contact force exceeding 500 N could indicate harsh or unstable landings, potentially leading to slippage or damage and thus we introduced a reward term[46] that penalizes excessive contact force.

3. **Exclusion of Airtime Reward:** In the benchmarked model, the airtime

reward was employed to encourage longer, visually appealing strides. However, this reward term was observed to lower the base height of the robot, inadvertently increasing the risk of failure upon contact with granular terrain due to insufficient clearance. Consequently, to adapt to the specific challenges posed by granular terrain, the airtime reward was removed from the reward function for Phase 1 and 2 of our training. This decision was supported by the rationale that in granular terrain, maintaining stability and controlled contact with the ground takes precedence over the aesthetic or potential efficiency gains attributed to increased airtime.

Reward Terms	Definition	Weight
Linear Velocity Tracking	$\emptyset\left(v_{b,xy}^{*} - v_{b,xy}\right)$	1 dt
Angular velocity Tracking	$\emptyset\left(\omega_{b,xy}^{*} - \omega_{b,xy}\right)$	0.5 dt
Linear Velocity penalty	$-v_{b,z}^{2}$	4 dt
Angular velocity Penalty	$-\left\|\omega_{b,xy}\right\|^{2}$	0.05 dt
Joint motion	$-\left\|\ddot{q}_{j}\right\|^{2} - \left\|\dot{q}_{j}\right\|^{2}$	0.001 dt
Joint torques	$-\left\|\tau_{j}\right\|^{2}$	0.00002 dt
Action rate	$-\left\|q_{j}^{*}\right\|^{2}$	0.25 dt
Collisions	$-n_{collision}$	0.001 dt
Stumble	$\sum(\lvert f_{xy}\rvert > 5) \wedge (\lvert f_{z}\rvert < 1)$	0.5 dt
Feet Contact Force	$\sum\left(\left(\lvert f_{xyz}\rvert - 500\right)_{+}\right)$	0.01 dt

Table 3.2: Definition of reward terms, for Phase 1 model with $\phi(x) := \exp(-4. \lvert x\rvert^{2})$

3.3 Terrain Generation

The development of robust locomotion policies for robots requires training within diverse and challenging environments. To facilitate this, a procedural terrain generation system coupled with an automated curriculum is employed. The terrain generation framework is designed to create a variety of terrains with adjustable complexity levels. These terrains serve as the training grounds for agents, where they are exposed to different conditions and challenges, simulating real-world scenarios.

3.3.1 Phase 1 Terrain Generation

In Phase 1, the curriculum and terrain generation were configured to offer a broad spectrum of environmental challenges across a wide array of terrains. This phase was characterized by a large number of 20 terrain columns and 10 levels in each terrain, creating a vast and diverse training landscape. A diverse set of terrain types, including smooth slope, rough slope, stairs up, stairs down, and discrete obstacles as shown in Fig 3.2. This diversity was aimed at developing versatile navigation strategies across different topographies. A larger map size (map Length: 8. and map Width: 8.), providing ample space for complex terrain configurations and navigation tasks.

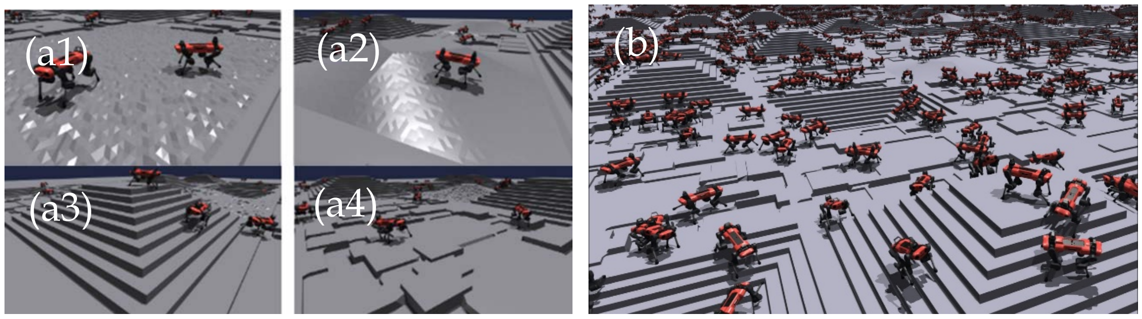

Figure 3.2: Terrain types used for training[34]: (a1) Randomly rough terrain with variations of 0.1 m. (a2) Sloped terrain with an inclination of 25 deg. (a3) Stairs with a width of 0.3 m and height of 0.2 m. (a4) Randomized, discrete obstacles with heights of up to ±0.2 m. (b) Parallel AnymalC robots progressing through various terrains with automatic curriculum.

3.3.2 Phase 2 Terrain Generation

The primary goal of Phase 2 is to train quadruped robots to adeptly traverse granular terrain, with a specific focus on gravel. This objective is underpinned by the realization that navigation through granular media presents unique challenges, including variable resistance and shifting footholds, which are not typically encountered on more stable, rigid terrains. To achieve this goal, we introduce Central Depression Terrain, a flat terrain with central indented region of 4*4 size which is specifically designed to hold particles, simulating the granular texture and behavior of gravel. This terrain serves as the training environment for the robots to adapt their locomotion strategies to the unstable and unpredictable nature of granular media.

To facilitate this specialized training, the map size (mapLength: 6. and mapWidth: 6.) and the number of terrains (numTerrains: 3) and levels (numLevels: 3) are strategically reduced as shown in Fig 3.3. This adjustment is made to match the maximum number of particles that the training environment can realistically accommodate, ensuring an optimal and focused training setup for granular terrain navigation. The reduction in environmental complexity allows for a concentrated effort on mastering traversal over granular media, within the computational and physical constraints of the simulation environment.

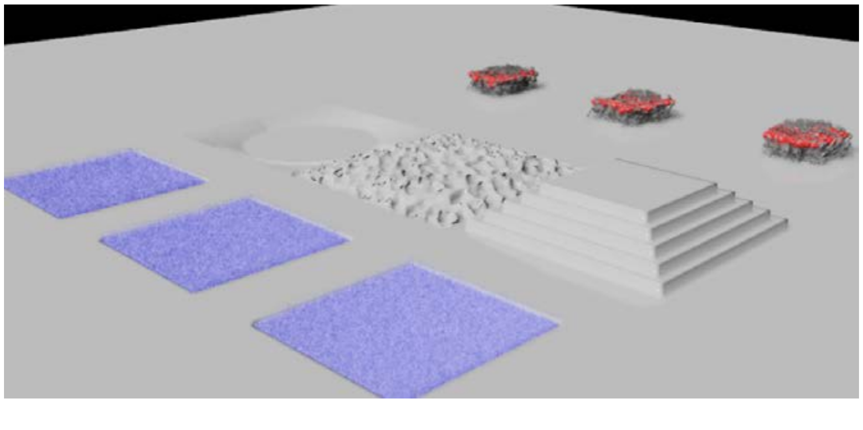

Figure 3.3: A depiction of the simplified training environment used for Phase 2 training on granular terrain. The environment is configured with a map size of 6x6 units and with the central depression terrain filled with particles visualized by blue color, also featuring three distinct terrains and levels to optimize particle count within computational limits.

3.4 Terrain Curriculum Design

One aspect of our training methodology is the incorporation of an automatic curriculum, inspired by game design principles. This curriculum progressively increases the difficulty of the terrain as agents improve their locomotion capabilities. The curriculum is structured around the concept of levels, where each level represents a terrain with a specific set of challenges. Agents start training on less challenging terrains, and as they demonstrate proficiency by traversing these terrains successfully, they are promoted to higher levels with more complex terrains. The curriculum adapts to the performance of each agent individually, ensuring that the difficulty level of the terrain is always matched to the agent's current capabilities.

3.4.1 Curriculum Dynamics

Difficulty Gradation: Terrains are categorized into different levels of difficulty, with parameters such as slope inclination, step height, and obstacle complexity gradually increasing.

Performance Feedback: The curriculum continuously evaluates the performance of robots based on the distance moved by an agent relative to its target velocity and dynamically adjusts the difficulty levels of terrains. Robots that successfully traverse challenging terrains progress to more difficult levels, while those struggling may regress to simpler terrains for reinforcement learning.

Termination Condition: The termination condition for training the robot is triggered when a timeout occurs after reaching the maximum episode length or if the robot falls, indicated by excessive contact forces on the base or knees.

Looping Mechanism: To prevent stagnation and encourage exploration, robots reaching the highest difficulty level are looped back to intermediate levels, promoting diversity and continuous improvement.

3.4.2 Phase 2 Curriculum Progression

The curriculum is meticulously structured to expose robots to an ascending level of terrain difficulty. Initially, robots encounter flat terrain, providing a baseline navigational challenge. Subsequently, they are introduced to three different types of rigid terrains in each column. The rigid terrains introduced at the second level of the curriculum are configured to present the maximum difficulty. This deliberate choice ensures that the robots are adequately challenged to apply and extend their previously acquired knowledge and skills from Phase 1, reinforcing their capabilities.

Finally, the curriculum culminates with the granular terrain, representing the peak challenge of Phase 2 challenging the robots to apply and adapt their learned skills to navigate effectively through the unstable and unpredictable gravel terrain.

3.5 Adapting Terrain Sampling in Particle-Filled Environments

In simulations involving rigid terrains, the height samples crucial for informing the robot's neural network are directly sampled from the ground terrain mesh. This method ensures that the robot receives accurate representations of the terrain it navigates. However, the introduction of terrains filled with particles in a central depression complicates this process. The distribution of particles on top of terrain can lead to incorrect inputs if sampled directly from the terrain mesh, potentially impairing the robot's ability to learn effective locomotion strategies.

Simplified Approach for Particle Terrain Sampling: Initially, methods were considered for extracting detailed particle information, filtering out particles to identify the top layer, and averaging these data points to construct an accurate representation of the terrain height. While theoretically sound, these approaches significantly increase the complexity and computational demands of the simulation, which can hinder real-time simulation performance for effective robot training. Also, even with advanced filtering and averaging techniques, the variability in particle distribution can lead to artifacts and inconsistencies in the height data fed into the network, potentially impairing the robot's learning process. To address this challenge, we adopted a simplified method that involves adjusting the sampled height values in depression gap to match the height level filled with particles.

Monitoring and Maintaining Particle Density: An additional layer of complexity is introduced by the dynamic nature of particle simulations, where the number of particles within a designated area can fluctuate due to the robot's interactions with the terrain. To ensure consistency in the terrain representation, we implemented a function to check the particle density within the depressions at intervals of 15 seconds. If the number of particles falls below 80 percent of the initial value, indicating a significant dispersion of particles, we reset the particles to their initial state. This process also involves identifying robots within the affected grid and resetting them to avoid collision while resetting.

3.6 Sim-to-Real Transfer Additions

To address these challenges and facilitate a more seamless sim-to-real transfer, several specific measures are implemented:

- **Friction Coefficient Randomization:** Each robot's interaction with the ground is subject to variability, with friction coefficients sampled uniformly from the range [0.5,1.25]. This randomization introduces variability in traction, simulating a wider array of real-world surfaces and conditions the robots might encounter, from slippery floors to rough terrains.

- **Observation Noise Incorporation:** Real-world sensor data invariably includes noise, which can significantly affect a robot's ability to perceive its environment accurately. To acclimate the robots to these conditions, noise is added to the observations within the simulation, based on actual measurements from real robots. This practice ensures that the trained policies are not overly reliant on pristine, noise-free data and can maintain their performance in the presence of sensor inaccuracies.

Joint Positions	$\pm 0.01\ rad$
Joint velocities	$\pm 1.5\ rad/s$
Base linear velocity	$\pm 0.01\ m/s$
Base angular velocity	$\pm 0.2\ rad/s$
Projected gravity	$\pm 0.05\ rad/s^2$
Measured terrain heights	$\pm 0.01\ m$

Table 3.3: For every element within the observations, a noise value is drawn from a uniform distribution of the specified scale and then added to the corresponding observation component to introduce variability.

- **Random External Pushes:** To further prepare the robots for unexpected disturbances, they are subjected to random pushes during training episodes. These disturbances occur every 10 seconds, with the robots' bases being accelerated up to ±1 m/s in both the x and y directions. This intervention teaches the robots to maintain or quickly regain stability, a critical skill for real-world deployment where unexpected interactions with the environment or with humans can occur.

- **Dynamic Particle Parameter Randomization:** In addition, our simulation includes environments with dynamically adjustable gravel particle parameters during Phase 2 training. We dynamically randomize several PBD parameters from Table 3.4 around selected nominal values every 20 seconds which includes the density, friction, adhesion, and particle friction scale of the granular particles.

Density	[2500, 3500]
Friction	[0.3,0.4]
Particle Friction Scale	[0.75, 1.25]
Adhesion	[0.0001, 0.0003]

Particle Adhesion Scale	[1, 5]

Table 3.4: The Ranges for dynamic randomization of particle parameters in simulation environments. These parameters are adjusted every 20 seconds during Phase 2 training to enhance the quadruped's adaptability to varying granular terrain properties.

3.7 Warp: A Differentiable Physics Simulator

We started with an initial attempt to model deformable interactions using traditional Finite Element Method (FEM) technique within NVIDIA's Isaac Sim, which resulted in suboptimal and unnatural behaviors in the quadruped simulation. The outcomes may diverge significantly from real-world scenarios as they do not adequately reflect the hydrodynamic-like properties of granular media.

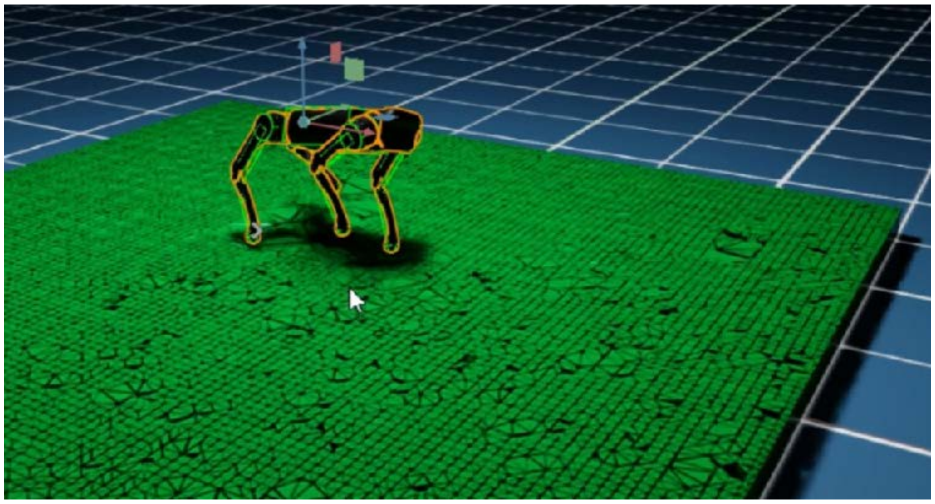

Figure 3.4: Simulation demonstrating the results of an attempt to apply deformable body simulation[47] using Finite Element Method (FEM) approaches within NVIDIA's PhysX and Isaac Sim. The depicted quadruped robot exhibits instability, characterized by excessive jittering of the mesh object, persistent sliding, and unnatural jumping behavior despite parameter adjustments, leading to non-realistic motion patterns.

In the progression of this study, the Warp framework emerged as an initial tool of interest for simulating complex physical interactions within the Isaac Sim environment. Warp's proficiency in handling differentiable physics made it a suitable candidate for exploring deformable terrain dynamics. However, the study's focus transitioned from Warp to a PBD approach as the research unfolded. This chapter delves into the initial forays with Warp, articulating the reasons for its selection and the subsequent pivot to PBD within Isaac Sim. It discusses the challenges encountered with Warp simulations, such as scalability and the integration complexity within the Universal Scene Description (USD)-centric environment of Isaac Sim. The limitations that led to the discontinuation of Warp in favor of a PBD-based methodology are analyzed, providing insights into the decision-making process behind simulation framework selection.

3.7.1 Synergy Between Warp and Isaac Sim

Warp[48] is a Python framework designed for high-performance GPU simulations in real time applications, particularly for scenarios where differentiable physics is required in robotics, game development, and virtual reality. While Warp is not a replacement for fully featured physics engines like Isaac Sim or PhysX, it is designed to complement existing physics engines by focusing on scenarios that benefit from differentiable physics. Its capability to handle complex physical interactions, especially with deformable objects and terrains, positions Warp as a valuable tool for simulations requiring gradient-based optimization.

The integration of Warp's deformable terrain simulation capabilities within the Isaac Sim environment offers a powerful platform for this study. The Universal Scene Description (USD) framework serves as a bridge between Warp and Isaac Sim, enabling the incorporation of Warp-generated deformable terrains into the Isaac Sim environment. By exporting Warp-simulated terrains as USD files, we create portable and interoperable scene representations that can be readily imported into Isaac Sim. After simulating a particle grid in Warp, the scene is exported as a USD file. This file encapsulates the geometric and dynamic properties of the terrain, including its deformability attributes. Within Isaac Sim, the USD file is imported to create a new scene or augment an existing simulation framework. Then, we establish a communication channel with the help of Action Graphs to allow the Isaac Sim environment to interact dynamically with the imported particle objects.

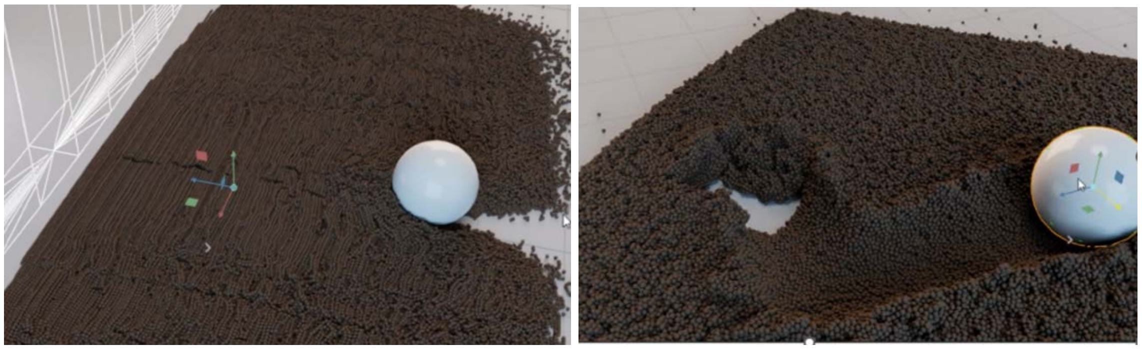

Figure 3.5: Interaction of the collider mesh within the DEM particle implementation in Warp, showing the effects of mesh collision with the granular material.

Our approach was inspired by NVIDIA's showcase of Discrete Element Method (DEM) example scenes within Isaac Sim. By studying these examples, we gained insights into the effective simulation of complex physical interactions between robots and their environments. Starting from these foundational examples, we expanded our framework to include Warp-generated deformable terrains, further enhancing the realism and challenge of our training scenarios.

3.7.2 Action Graph Integration

Action graphs in Isaac Sim provide a robust mechanism for orchestrating complex simulations, enabling the control and coordination of various simulation elements, including those imported from Warp. Omnigraph is a graph-based programming framework used within Isaac Sim. It provides the infrastructure for the creation and management of complex simulations through a network of interconnected nodes. This system enables the integration of disparate simulation components and the management of complex data flow within the simulation environment. Several nodes come to play to enable this interaction.

The "OgnParticlesFromMesh" node is designed to generate particles within a 3D mesh within the Omnigraph framework. It takes the input mesh and transforms its point positions into world space. It then uses a custom kernel function to fill a volume defined by the mesh with particles. This kernel evaluates each point within a grid overlaid on the mesh, determining if the point falls within a specified range from the mesh surface. Points that meet the criteria are jittered randomly to prevent uniformity and then stored as particle positions. Once the valid particle positions are identified, the node initializes particle properties such as velocity, mass, and radius based on the input parameters that are uniformly set across all particles. The node packages the resulting set of particles into a bundle that can be used by the next node.

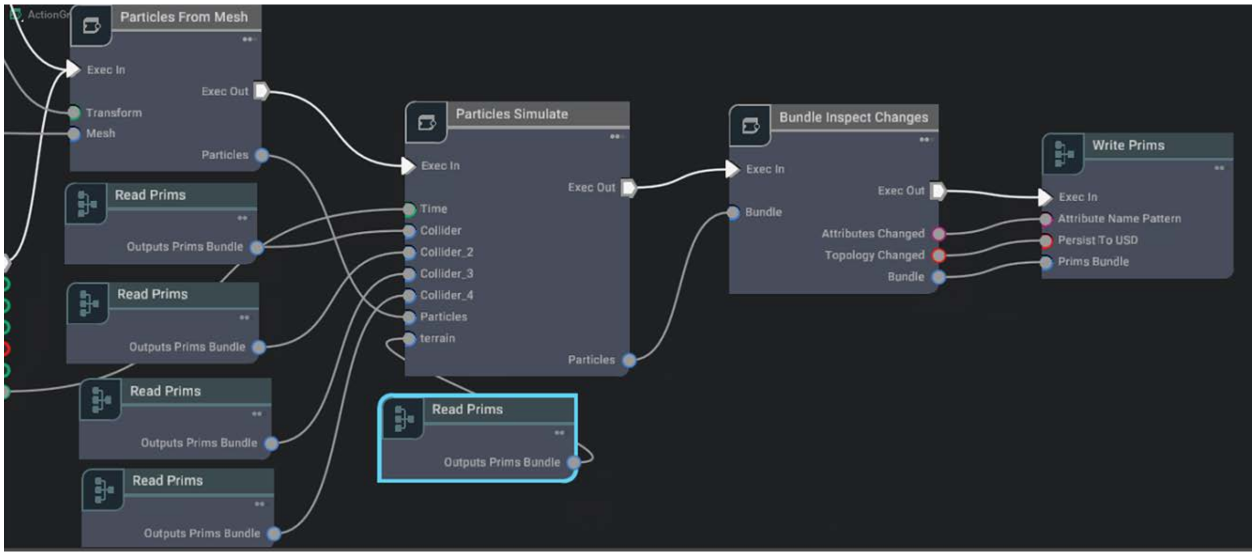

Figure 3.6: An action graph visualizing the data flow and processing sequence for particle simulation, highlighting nodes for reading, simulating, inspecting, and writing primitives.

OgnParticlesSimulateNode's primary function is to simulate particle dynamics within Isaac Sim. It interacts with other components, such as rigid body colliders managed by PhysX, handles the transformation of particle positions to and from world space, enabling the simulation to reflect changes in the environment accurately and updates particle velocities based on collisions and interactions, providing a realistic simulation of particle movement and behavior.

Upon receiving input from a particle simulation node, the "OgnBundleInspectChanges" node monitors these particle attributes for changes. It utilizes the Omniverse Graph change tracking system to record which attributes have been altered since the last evaluation. This is achieved through a comparison of the current state of attributes against their previous state. Once changes are detected and recorded, the node outputs a bundle that reflects the current state of the simulation entities, along with a string of changed attribute names and a flag indicating whether topology changes occurred. It then activates the execution for downstream node, called "Write Prims" node, enabling them to reflect the changes made to the particles in the simulation environment.

3.7.3 Particle Dynamics

Particle-Particle Dynamics:

In the simulation, each particle is subject to forces arising from contact with other particles. The force acting on a particle i due to another particle j is given by the sum of the normal force and the frictional force. The normal force (F_n) is computed based on the contact stiffness (k_n) and damping (k_d), while the frictional force (F_t) is calculated using a friction coefficient (μ) and the frictional stiffness (k_f).

The normal force is given by:

$$\mathrm{F_n} = -\mathrm{k_n}.\, \mathrm{c} - \min(\mathrm{v_n}, 0)\,.\, \mathrm{k_d}$$

where c is the penetration depth between the two particles and v_n is the relative normal velocity.

The frictional force is governed by Coulomb's friction law:

$$F_t = -\min\left(k_f v_s, \mu |F_n|\right)$$

where v_s is the magnitude of the relative tangential velocity. The direction of the frictional force is opposite to the relative tangential velocity.

The total force (F) acting on the particle is the combination of the normal and frictional forces:

$$F = F_n n + F_t t$$

where n is the normal vector and t is the tangential vector at the point of contact.

Particle-Collider Dynamics:

The interaction between a particle and a collider is modeled by calculating the forces due to contact, damping, and friction. The contact force f_n is the elastic response due to the compression c between the particle and the collider, and is calculated as:

$$f_n = n \cdot c \cdot k_e$$

where n is the contact normal, c is the penetration depth, and k_e is the combined stiffness constant derived from both the particle and the collider material properties.

The damping force f_d counteracts the relative normal velocity v_n at the point of contact, and is given by:

$$f_d = n \cdot \min(v_n, 0) \cdot k_d$$

where k_d is the combined damping constant.

The friction force f_t opposes the relative tangential motion. Assuming Coulomb friction and a smooth friction model for stability around $|\boldsymbol{v_t}| = 0,$ the friction force is:

$$f_t = \frac{v_t}{|v_t|} \cdot \min\left(k_f |v_t|, |\mu c k_e|\right)$$

where v_t is the tangential component of the relative velocity, k_f is the combined friction coefficient, and μ is the combined dynamic friction coefficient.

The total force f_{total} applied to the particle is the sum of the contact, damping, and friction forces:

$$f_{total} = f_n + f_d + f_t$$

The torque t_{total} exerted on the collider due to the force at the point of contact is given by:

$$t_{total} = r \times f_{total}$$

where r is the vector from the collider's center of mass to the point of contact.

3.7.4 Integration Hurdles

Staging Together:

Warp and PhysX operate with distinct APIs and data models. While Warp provides a powerful set of tools for particle simulation, direct code-level integration with Isaac Sim can pose significant challenges due to the differences in the data models, the need for synchronization of state across various systems, and the complexities involved in managing dependencies and interactions.

Universal Scene Description (USD) emerges as a solution to these challenges by providing a coherent and consistent way to represent the state of the simulation across all components. The USD format, being a comprehensive scene description framework, allows for the encapsulation of a wide variety of scene data including geometry, shaders, and rigging, as well as dynamic simulation states. It serves as an intermediary layer between Warp, PhysX, and other components within Isaac Sim. With USD, simulations can scale more efficiently by leveraging its capabilities for referencing and instancing. Within the action graph, dedicated nodes are used to read from and write to the USD stage, enabling the interaction of Warp particle dynamics with the PhysX-based physical world. By staging particle dynamics through USD, the action graph facilitates a smooth data flow between Warp and PhysX. Actions can be defined to trigger updates in the particle system based on physical events or changes in the environment, and vice versa.

Model Scale and Parameter Normalization:

A challenge encountered in the integration of Warp particle dynamics with PhysX in Isaac Sim revolved around the issue of model scale and selection of appropriate simulation parameters. Specifically, existing Warp particle scene was scaled at 100 times larger than typical setups which lead to discrepancies in integration, making realistic simulations hard to achieve. Most particle parameters were declared dimensionless, hence, all parameters of the simulation, including particle sizes, distances, and physical force, were carefully tuned to easily map to corresponding simulation entities, ensuring that all components adhered to the adjusted scale.

The transformation of the coordinate system to align Warp's Y-axis up configuration with the standard Z-axis up orientation in Isaac Sim ensured consistent orientation across simulations.

Multiple Collider Handling:

The original OgnParticlesSimulateNode was designed to handle a single collider, which limited its applicability for complex models such as quadrupeds. The first step was to adapt the node's internal data structures to accommodate multiple colliders. This involved redefining the data structure to hold an array of collider entities instead of a single entity. Each entity in this array represents a separate collider mesh, allowing for individual configuration and update routines. Extending the collider handling system to manage multiple colliders enabled the assignment of individual colliders to each leg of a quadruped model.

Ground Terrain Integration:

In the development of our simulation framework, we encountered the limitation of Warp's default ground collider, which was represented as a flat plane. This simplistic representation was insufficient for simulating the complex interactions between particles and varied terrain surfaces, which are crucial for accurate environmental simulations. To address this, we modified the OgnParticlesSimulateNode to integrate a terrain mesh as a collider, thereby replacing the default flat ground collider. Warp's default flat ground collider was disabled, ensuring that the simulation engine does not automatically generate the flat ground collider at runtime. To ensure efficient collision detection with the complex terrain mesh, spatial indexing structures, such as bounding volume hierarchies (BVH), were employed. These structures significantly improve the performance of collision detection algorithms by reducing the number of collision checks required.

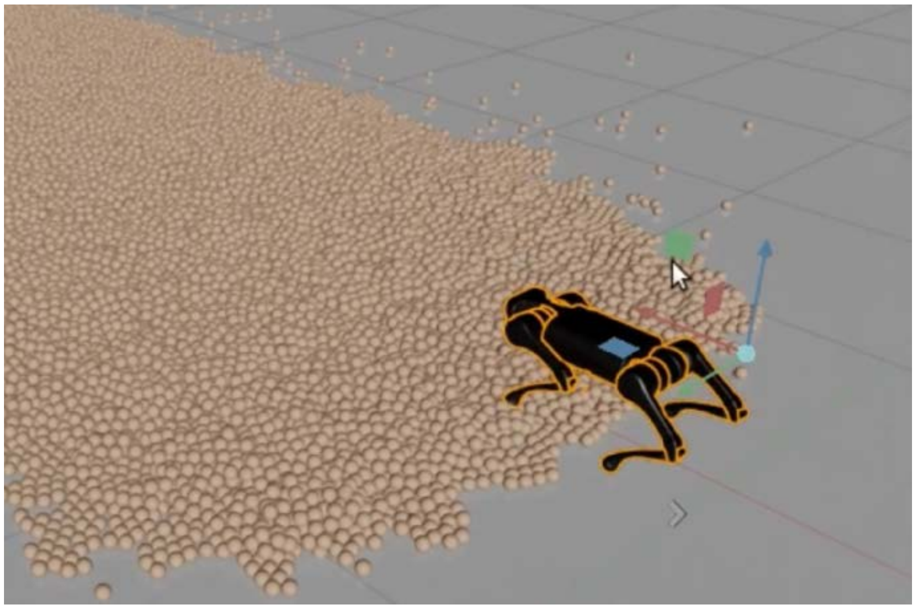
Figure 3.7: Integration of a quadruped robot with a discrete element method (DEM) particle system USD file from Warp in Isaac Sim, noting the slower performance due to the complexity of the simulation.

3.7.5 Persisting Challenges

Mesh Dynamics and Sphere Primitives

Warp operates by creating high-fidelity simulations of physical interactions, offering the mesh class to manage triangle mesh data. When integrating Warp with Isaac Sim, collider meshes from Isaac Sim are input into Warp, where an instance of the mesh is created within Warp's model builder. This process involves translating mesh data—points, indices, and velocities—into Warp's format and utilizing this data to simulate forces, collisions, and other physical interactions. The Mesh.refit() function plays a crucial role in ensuring the mesh accurately reflects dynamic changes, such as deformations or displacements, by rebuilding the bounding volume hierarchy (BVH) structure. However, this integration revealed limitations when dealing with simpler geometric representations, such as sphere primitives. Sphere primitives, defined by a center point and radius, offer efficient collision detection in many physics engines by simplifying calculations to radius comparisons. Unfortunately, Warp does not support the option to refit sphere primitives, rendering this efficient collision model incompatible with dynamic simulations in Warp.

Computational Efficiency

While Warp is designed for high-speed physics simulations, creating instances of collider meshes within its environment proved slow and inefficient for our purposes. The need to create instances within Warp's environment for each collider mesh and subsequently refit the BVH structure for dynamic updates significantly hampers the simulation's performance. This inefficiency stems primarily from the need to instantiate complex mesh data within Warp, separate from Isaac Sim's native handling of physics through PhysX.

Challenges with Excluding Interactions

A significant challenge arose from the necessity to exclude collisions between the quadruped's body and the ground within the Warp simulation, alongside excluding particle-body interactions in the PhysX environment of Isaac Sim. This exclusion was required to avoid redundant or conflicting collision processing between the two systems. However, it introduced substantial complexity in ensuring accurate and realistic simulation outcomes, particularly in scenarios where the dynamic feedback between the robot and the deformable terrain was critical for training fidelity.

One-Directional Interaction Limitations

Our simulation framework was initially designed with colliders treated as kinematic entities—meaning they could influence particles yet remained unaffected by them. This design simplifies collision detection and response calculations but fails to capture the reciprocal nature of physical interactions, such as the impact forces particles exert on colliders. Real-world scenarios involve bidirectional dynamics, where particles not only respond to colliders but also influence their state, a critical aspect in simulations aiming for high fidelity and realism.

To enable Bidirectional Interactions, we aimed an approach to extract the calculated body force vector on collider from particles within Warp. This vector was intended to aggregate the influence of all interacting particles on a given collider, thereby enabling the simulation to reflect the cumulative effect of particle impacts on the collider's state. The calculated force vector would then be applied externally to the PhysX solver, allowing for the dynamic adjustment of collider properties in response to particle interactions.

However, inbuilt support within Isaac Sim's action graphs for managing external force applications from one simulation solver (Warp) to another (PhysX) was absent. In response, we contemplated developing a custom node designed to bridge this gap by facilitating the application of Warp-calculated forces onto PhysX colliders. However, the integration complexities between two distinct solvers quickly became apparent. The lack of direct access and control over PhysX's internal mechanics, combined with the anticipated synchronization and state management issues, presented formidable obstacles.

Anticipated Integration Problems

Given the distinct computational models and data structures employed by Warp and PhysX, seamless integration was fraught with challenges. Synchronizing the state between solvers, ensuring consistent and accurate force applications, and managing the computational overhead of such interactions were among the anticipated problems. These challenges highlighted the significant integration efforts required to achieve bidirectional interactions, efforts that could potentially outweigh the benefits in our specific simulation context.

3.7.6 Discussion

The pursuit of bidirectional interactions between particles and colliders revealed critical limitations in our simulation framework's architecture and underscored the complexities of integrating disparate solvers. Despite the conceptual appeal of reflecting the reciprocal dynamics between particles and colliders, practical implementation hurdles—stemming from a lack of inbuilt support, access restrictions, and anticipated integration challenges—led to the decision against pursuing this feature further within the scope of our project. This experience, however, has provided valuable insights into the intricacies of simulation framework design and the importance of solver compatibility for future enhancements.

Given these challenges—specifically, the limitations in refitting simple primitives, computational inefficiencies, and difficulties in achieving accurate two-way interactions—we opted to pivot towards leveraging Isaac Sim's embedded PBD particle dynamics for simulating deformable terrains, eliminating the need for complex integrations between different systems.

3.8 Position-Based Dynamics in Isaac Sim

Although models with NVIDIA's Warp were extensively explored, Position-Based Dynamics (PBD) was ultimately chosen for our simulations in Isaac Sim due to its efficiency in real-time applications. PBD is a physics simulation technique for simulating complex interactions between particles in real-time applications. Unlike traditional rigid body dynamics, PBD operates on the principle of position-based constraints, allowing for more flexible and efficient simulations of deformable bodies, fluids, and granular materials. By leveraging the NVIDIA Omniverse's physX engine, Isaac Sim efficiently computes the interactions between millions of particles, making it possible to simulate complex terrains and interactions that RL quadrupeds can train on.

3.8.1 Theoretical Framework of PBD

The core of PBD lies in its iterative solver that enforces constraints to model physical behaviors. The basic equation governing PBD is derived from the principle of position correction based on constraints. For a particle i, the position $\boldsymbol{p}_i$ is updated based on a constraint function $C(\boldsymbol{p}1, \boldsymbol{p}2, \dots, \boldsymbol{p}n) = 0$, which describes the physical condition to be met. The constraints can represent anything from particle incompressibility, frictional contact, to cohesion and adhesion effects between particles. The position update equation can be formulated as:

$$\boldsymbol{p}_i' = \boldsymbol{p}_i + \Delta\boldsymbol{p}_i$$

where $\Delta\boldsymbol{p}_i$ is the position correction calculated to satisfy the constraint C. The correction $\Delta\boldsymbol{p}_i$ is computed as:

$$\Delta\boldsymbol{p}_i = -\lambda\nabla_{\boldsymbol{p}_i}C(\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_n)$$

with λ being a Lagrange multiplier determined by the constraint's compliance and $\nabla\boldsymbol{p}_i C$ representing the gradient of the constraint with respect to particle i's position.

Advantages: While DEM is another popular method for simulating granular materials, it differs from PBD in its approach to force computation and integration. DEM calculates forces based on collisions and contacts between discrete elements and integrates Newton's equations of motion to determine particle trajectories. This method is highly accurate for simulating granular flow and force transmission but can be computationally intensive, especially for large numbers of particles.

PBD, on the other hand, offers a more efficient alternative for real-time applications by directly manipulating particle positions to satisfy constraints, thereby avoiding the direct computation of forces. This efficiency makes PBD particularly appealing for training RL quadrupeds in simulated environments, where computational resources are a limiting factor.

3.8.2 Creating Particle Systems

The first step in simulating granular terrain involves creating a particle system within the stage of Isaac Sim. A physics scene is defined with gravity set to Earth's standard gravitational acceleration (9.81 m/s^2), which sets the foundational physics parameters for the particle simulation. Subsequently, a particle system is created and configured with specific properties such as rest offset, contact offset, and maximum velocity, which dictate the behavior and interaction of particles within the system.

Key configurations for the particle system include:

Solid Rest Offset: Determines the size of solid particles or the proximity at which two solid particles are in contact. Two solid particles are touching when their centers are two times Solid Rest Offset apart.

Particle Contact Offset: Defines the neighborhood radius for each particle. Particles within twice this offset from one another are considered neighbors, affecting constraint evaluations during simulations.

Contact Offset (Particle-Non-Particle Interaction): This parameter specifies how close a particle must be to a collider before contact constraints are generated and processed by the solver.

Rest Offset (Particle-Non-Particle Interaction): Represents the effective contact distance from a collider's surface for both solid particles and non-particle objects.

Collision Offset: A buffer distance from a collider's surface where contact begins to be generated, particularly useful for preventing fast-moving objects from tunneling through colliders.

MaxNeighborhood: Caps the number of neighbors a particle can have, with the default setting being 96. This parameter is vital for managing the computational load during constraint evaluations. Set to 200.

Max Velocity: Dictates the upper limit on the speed at which particles within the simulation can move, which is set as 100.

Time Step: Our initial configuration used a time step of 200, providing us with satisfactory results for our baseline simulations. However, in pursuit of enhanced simulation performance, modifications to the time stepping for specific elements within the simulation were considered, particularly for the particle system. The idea was to reduce the particle system's time step, effectively halving it in an alternative physics scene, to achieve faster simulation of the particles without compromising performance. While Isaac Sim supports the creation of multiple physics scenes with varying time steps for rigid bodies, enabling a hierarchical priority system for dynamics calculations and collision handling, it does not extend this functionality to particle systems. Consequently, our efforts to employ a separate time stepping for particles were impeded by the current capabilities of Isaac Sim.

It is generally understood that maximizing the simulation time step is crucial for achieving maximum throughput. Policy steps, conducted at 50 Hz in our case, necessitate several actuator and simulator steps to ensure stability. These additional computation steps are inversely proportional to the overall efficiency of the simulation; thus, reducing them is of utmost importance. It has been documented that reducing the time step below 0.005 seconds (corresponds to four simulation steps per policy step) is not viable, as it leads to instability in the actuator network designed to mimic a discrete-time PD-controller. This instability is not a limitation of the simulation's physics engine but a consequence of the actuator network's design constraints.

Unlike the aforementioned systems, our quadruped simulation utilizes a straightforward PD controller rather than a separate actuator network for joint control. This allows for a more direct and less computationally intensive approach. With the intention of scaling the overall physics step time from 0.005 s to 0.01 s, we aimed to halve the computational load. Unfortunately, this adjustment did not yield a stable gait within the robot, indicating that other parameters tied to the simulation's stability might require recalibration. Notably, rewards are normalized by the time step, and changes to the step size could disrupt this balance. Additionally, the calculation of contact forces, which involves multiplying the impulse by the time step, could also be adversely affected by these changes.

Due to these complexities and the instability observed during preliminary testing, we ultimately decided to maintain a time step of 0.005 s for our physics scene.

Particle Instancing for Granular Terrain

With the particle system and material configured, we proceed to create particle instancers. These instancers are responsible for defining the spatial distribution of particles, effectively shaping the simulated granular terrain. Initially, our strategy involved creation of particles by instancing them from a predefined cube mesh based on the mesh's volume and shape.

However, transitioning to a headless training environment, where graphical interfaces are disabled to optimize computational resources for simulation and learning tasks, resulted in errors. So, we opted for manually initializing particle positions and velocities for each particle within the simulation environment. The dimensions of the particle grid are carefully chosen to match the pit in the central depression terrain, such as the depth and width of a central depression. To fully accommodate the depth of the depression and avoid any voids that could undermine the realism of the simulation, the height of the grid is increased to allow particles to be realistically filled and compacted.

Jitter: The spacing between particles within the grid is chosen to be 2.5 times the radius of the particles. To prevent the uniformity inherent in a grid arrangement and to introduce a degree of randomness that mimics natural terrain more closely, a jitter factor of 20% to particle spacing is applied to the positions of particles.

Visualization and Material Binding: A visual representation with blue color is assigned to the particles, and physical materials are bound to the particle system to render the simulated terrain visually and interactively plausible.

3.8.3 Particle Material Configuration

Dimensionless Parameters: One of the challenges encountered during the simulation of granular materials in Isaac Sim was the dimensionless nature of many parameters within the PBD framework. Unlike traditional simulation parameters that are directly tied to physical quantities (e.g., mass in kilograms, velocity in meters per second), PBD parameters in Isaac Sim, such as particle friction scale and adhesion offset scale, do not have direct real-world counterparts. This discrepancy introduces complexity in calibrating the simulation to closely mimic the behavior of real granular materials like gravel.

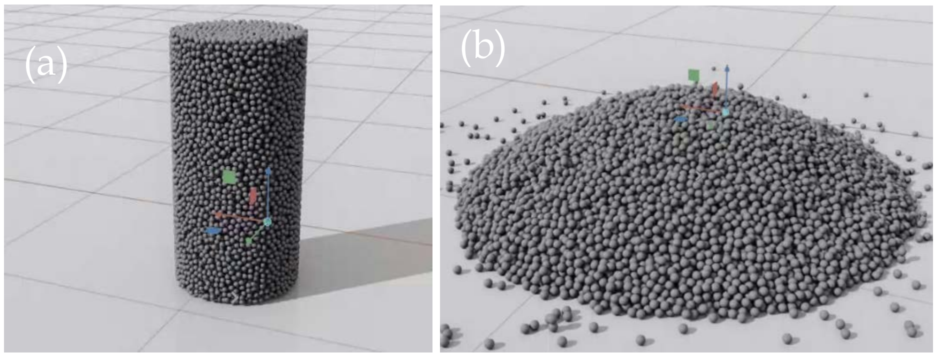

Figure 3.8(a): Initial state of a column drop test showcasing a cylindrical stack of particles (b) Resultant dispersion of particles following the column drop test, illustrating the spread upon impact.

Column Drop (Angle of Repose) Test: To address this challenge and ensure the simulation parameters reflect real-world granular behavior as accurately as possible, we employed empirical testing, specifically the static angle of repose test. The angle of repose is the steepest angle at which a pile of unconsolidated granular material remains stable without any particles sliding down. This angle provides critical insights into the frictional properties and collective behavior of granular materials.

Based on the insights gained from the angle of repose tests, we selected the following parameters for the PBD particles in Isaac Sim:

Friction: Set to 1, providing a middle-ground friction value that approximates the natural resistance of granular materials against sliding over each other.

Particle Friction Scale: Kept at 0.3 to not alter the base friction significantly, ensuring the simulation mirrors the natural frictional behavior observed in the angle of repose tests.

Damping: Set to 10 to prevent excessive particle dispersion upon contact.

Adhesion: Minimally set to 0.0001, with an offset scale of 0.009.

Particle adhesion scale: Set to 1.0. Despite varying the particle adhesion scale values across a wide range, from minimal to extreme values, there was no discernible difference in how the particles adhered to form lumps or exhibited increased cohesion. Ideally, adjusting these parameters should have resulted in particles displaying a tendency to stick together. However, such behavior was not observed.

Gravity Scale: Maintained at 1.0 to simulate the Earth's gravity.

Density: Set to 2500 kg/m^3, aligning with the density of gravel.

3.9 Simulation Environment

Our project embarked on a mission to replicate and extend the findings of the groundbreaking paper, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning[34]," which leveraged the power of GPUs for efficient quadruped locomotion training. The initial benchmarking efforts were guided by parameters and models detailed in the paper, with the original codebase hosted on GitHub by leggedrobotics/RSL under "legged_gym" within the Isaac Gym framework.

3.9.1 Choosing Isaac Sim over Isaac Gym

Despite Isaac Gym's capabilities and the option to use PhysX for physics simulations, several key factors motivated our transition to Isaac Sim:

- **Graphical User Interface (GUI):** Isaac Sim provides a GUI that significantly aids in testing and visualizing particle dynamics, a feature absent in Isaac Gym. The GUI allows for intuitive interaction and visual feedback with the simulation environment, enabling rapid prototyping and debugging.
- **Particle Dynamics and Integration:** No prior model or framework within Isaac Gym explicitly demonstrated the integration or manipulation of particles at the code level. Isaac Sim, conversely, offered built-in demo scenes support for simulating complex particle dynamics, a feature that was not readily accessible or well-documented in Isaac Gym's existing frameworks.
- **Availability of compatible foundational codebase:** We utilized the code from OmniIsaacGymEnvs[49] provided by Nvidia Omniverse, which contained a crude implementation of the model akin to that used in the legged_gym. This repository provided a foundational base tailored for Isaac Sim.

3.9.2 Implementation Differences

Transitioning to Isaac Sim also highlighted the differences in code implementations between the two environments. One of the notable difference was the approach to the Proximal Policy Optimization (PPO) algorithm. The legged_gym[50] codebase for Isaac Gym utilized a custom implementation designed for parallelized training of quadruped locomotion. In contrast, OmniIsaacGymEnvs for Isaac Sim leveraged RLGames, a versatile framework, necessitating adaptations.

Incorporation of Missing Reward Terms: The initial codebase lacked airtime reward and collision avoidance reward terms. Integrating these rewards was essential to align with the benchmarks established in the literature.

Incorporating these rewards requires accurate measurement of foot and knee contact forces. The RigidPrimView class is designed to create a focused view of specific rigid prims (primitive objects). Hence, to collect the forces, addition of a RigidPrimView specifically targeting the robot's shank and knee components was required.

Implementation of a Velocity Curriculum: A direct replication of the model and parameters did not initially yield the expected learning reward curves. It was found that the robots struggled to get to the next levels in learning. To address this, a simple velocity curriculum was introduced, starting the quadrupeds at lower velocities, and gradually increasing the command range as their performance improved.

Specifically, if the average reward associated with a motion command (e.g., linear velocity in the x and y directions, angular velocity around the z-axis) exceeds a predefined threshold (80% in our case) of the maximum possible reward, the range of permissible commands is expanded by 0.5 m/s, otherwise reduced by the same.

Addition of Helper Functions: we incorporated additional helper functions and variables to meticulously collect and organize data during test runs. These functions are integral for post-run analysis, enabling us to extract valuable insights from every episode's performance metrics during the inference phase.

Chapter 4

Results

Our analysis is grounded on benchmarking results obtained from the implementation detailed in the paper "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning[34]." The progression of our research through Phase 1 and Phase 2 training sessions is critically assessed, showcasing the adaptability and performance enhancements achieved with our methodologies.

4.1 Benchmarking Against Prior Work

Our initial benchmarking efforts focused on replicating the performance metrics as reported Rudin, Nikita, et al[34]. By leveraging the parameters and models detailed in the paper, we aimed to establish a solid foundation for our subsequent experiments.

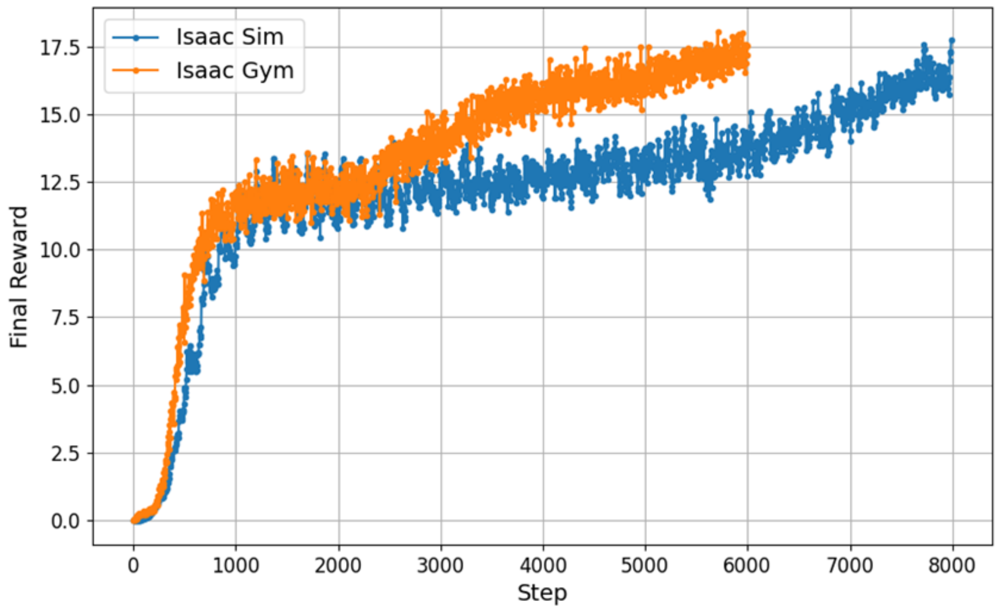

Figure 4.1: Comparison of average episodic total reward curves from Isaac Gym and Isaac Sim, demonstrating the consistency of the benchmark training model across different

simulation platforms.

Hence, we replicated the model described in the aforementioned paper within Isaac Gym, providing a baseline for evaluating our subsequent modifications and training sessions within Isaac Sim. The reward curves obtained from both Isaac Gym and Isaac Sim exhibited a high degree of similarity, affirming the robustness of the training model across different simulation environments.

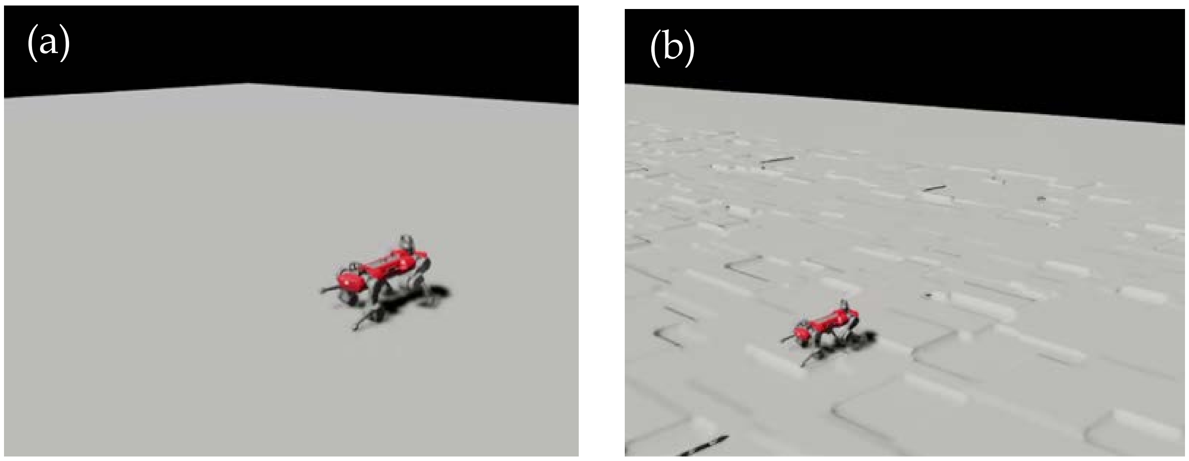

Figure 4.2: Evaluation run with command speed of 1 m/s in both horizontal directions and no angular velocity on a (a) flat terrain simulation (b) terrain with randomized, discrete obstacles.

The top row graphs in Fig. 4.3 and 4.4 show linear and angular velocity tracking. The solid lines (actual velocities) are close to the dashed lines (commanded velocities), indicating the controller is effectively tracking the desired speeds in both linear (x, y) and angular (z) velocities. The middle row shows non commanded velocity components which maintains balance and handles the intricacies of both flat and uneven terrains The bottom row graphs depict joint position tracking for the hip, thigh and knee joints of the left front leg. Again, the actual positions (solid lines) closely follow the target positions (dashed lines), with some expected oscillations due to dynamic interactions between the robot and the terrain.

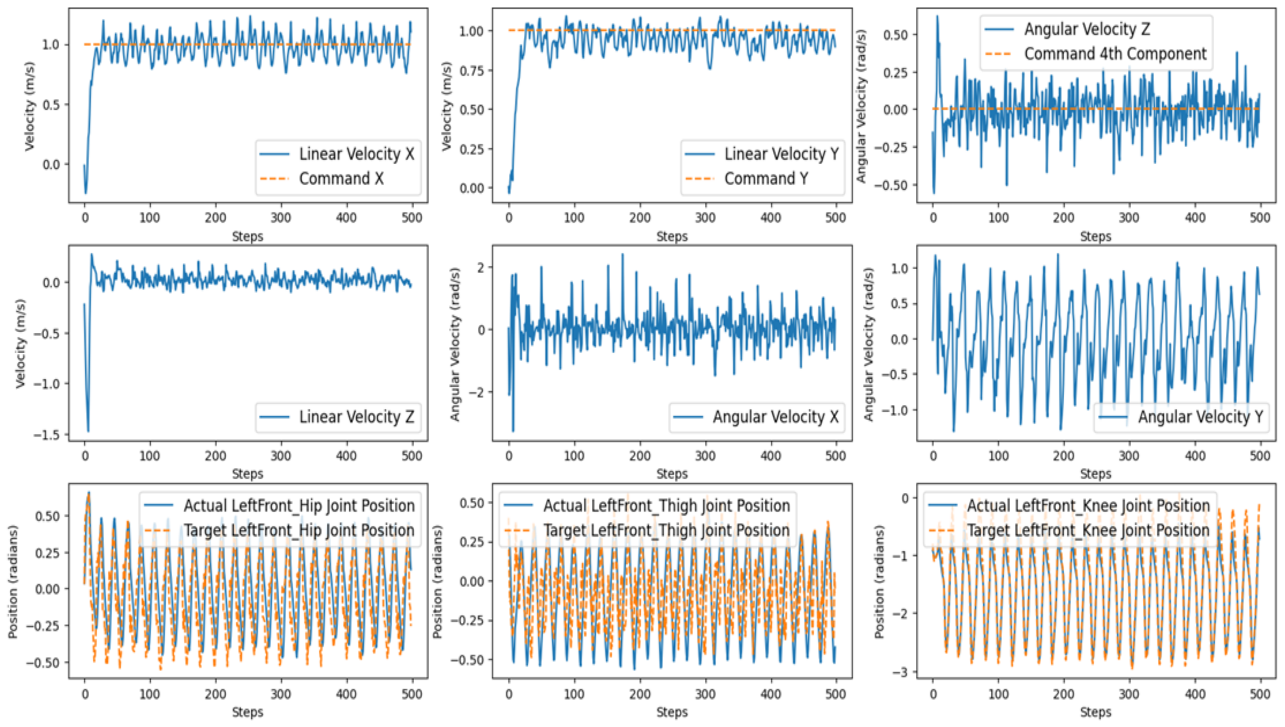

Figure 4.3: Performance metrics for a quadruped robot test run on flat terrain, showing the linear and angular velocities as well as hip, thigh, and knee joint position tracking.

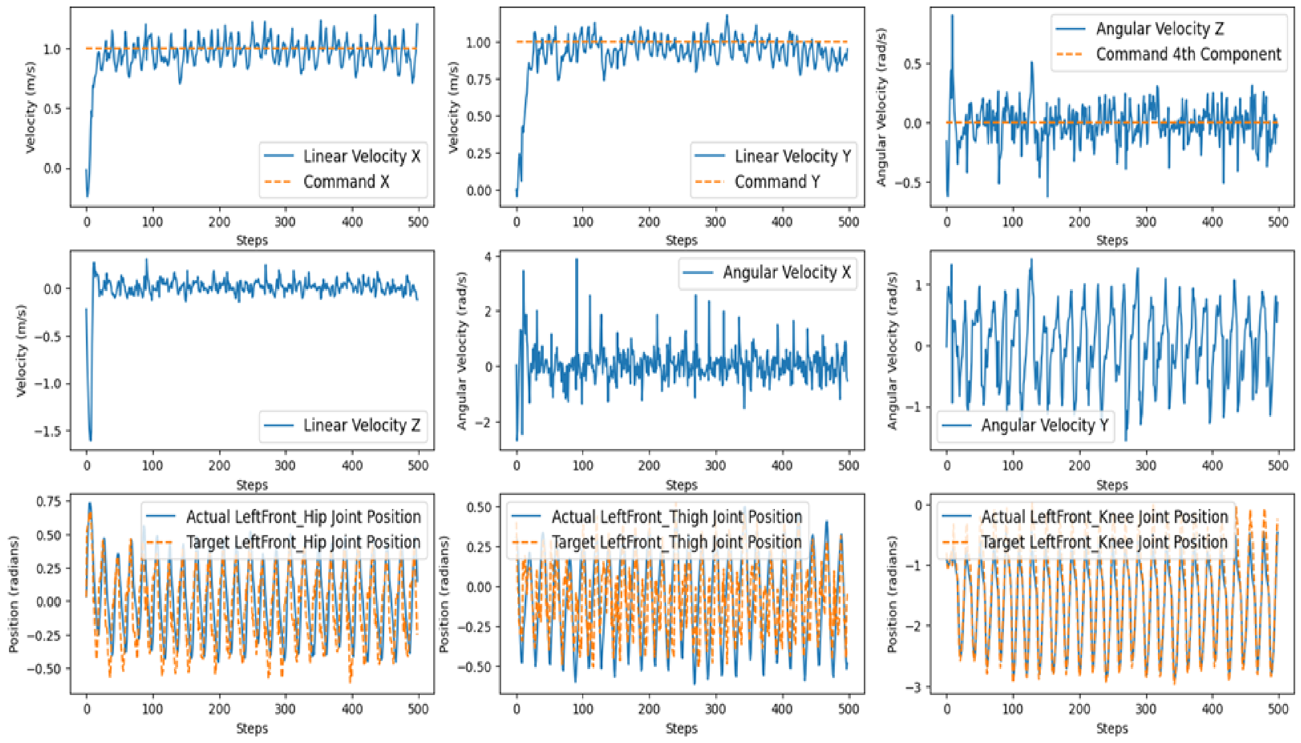

Figure 4.4: Performance metrics for a quadruped robot test run on randomized, discrete obstacles, showing the linear and angular velocities as well as hip,thigh and knee joint position tracking.

4.2 Phase 1 Training: Quadruped Locomotion Without Particles

Building upon the benchmarked model, we proceeded to Phase 1 of our training regimen, where the focus was on training the quadruped model in environments devoid of particle dynamics. This stage involved 2000 epochs of intensive training with an ensemble of 2048 robots.

Inference mode: For inference, we employed a distinct setup from our training configuration. Specifically, we used a 6x6 size grid of particles on a single terrain, accounting for granular interaction forces. The inference was conducted on a single robot and episode length was set to 6 secs considering the size of the particle grid to capture accurate particle interactions. Learning is not active and domain parameters are set constant. Only an x-command velocity of 2 m/s was given for all the following inference runs.

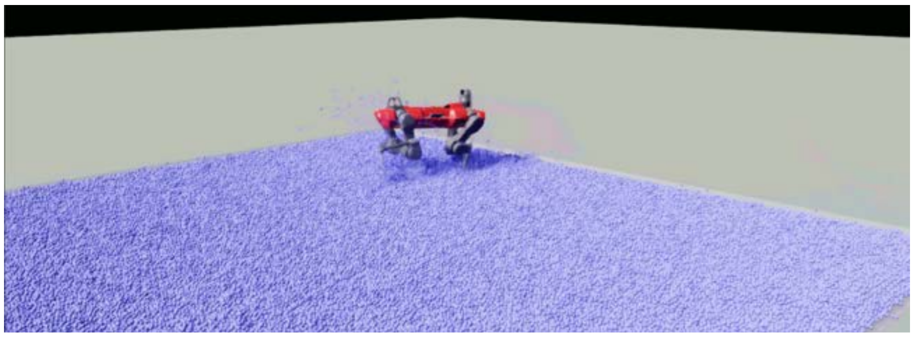

Figure 4.5: Inference environment showcasing a quadruped robot navigating a 6x6 particle grid navigating a dense particle field over a 6-second episode.

Velocity Tracking: The mean values for linear velocity reward in the XY plane and angular velocity reward around the Z-axis are slightly higher in the Phase 1 Model, suggesting improved velocity tracking performance. Also, the linear velocity RMSD plot shows that after an initial peak, both models settle into a pattern, with Phase 1 consistently maintaining a lower RMSD compared to the benchmark model, where the equations for the RMSD can be defined as:

$$\mathrm{RMSD}_{\mathrm{lin\_x}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(v_{\mathrm{base\_x},i} - v_{\mathrm{cmd\_x},i}\right)^2}; \ \mathrm{RMSD}_{\mathrm{ang\_z}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\omega_{\mathrm{base\_z},i} - \omega_{\mathrm{cmd\_z},i}\right)^2}$$

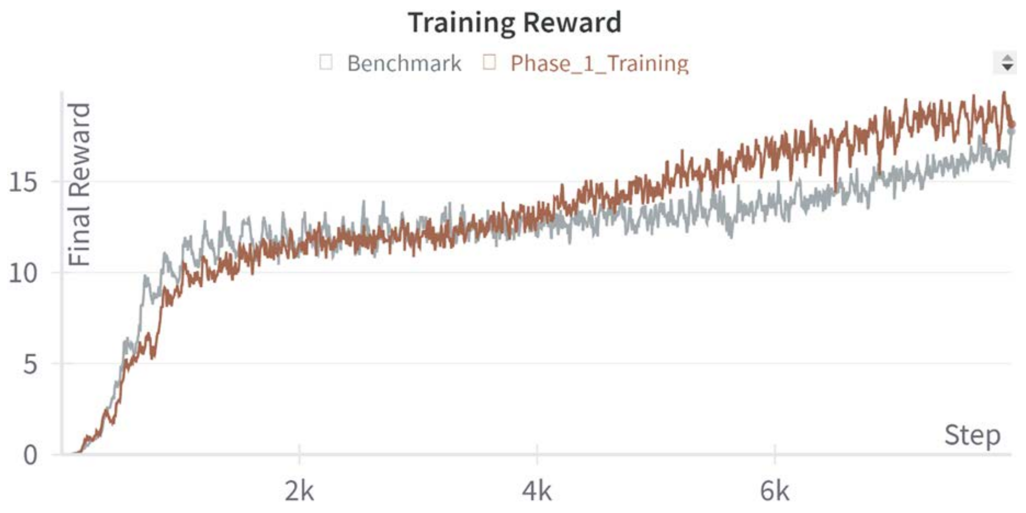

Figure 4.6: Comparison of average episodic total reward curves in training from Benchmark and Phase 1.

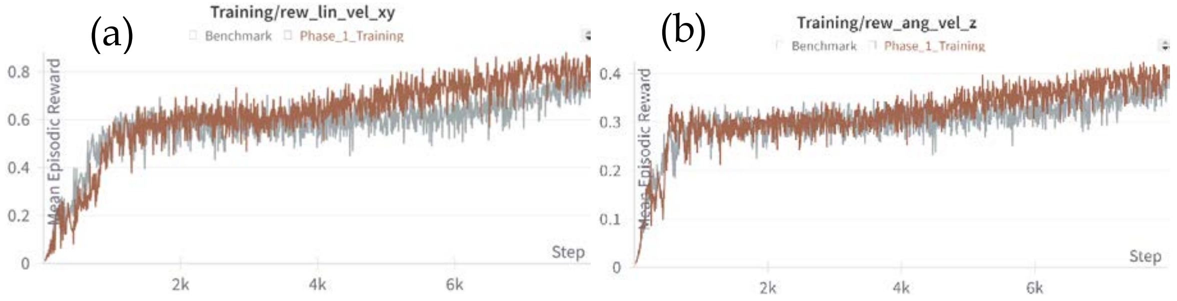

Figure 4.7: Comparison of (a) linear xy velocity and (b) angular z velocity tracking mean episodic rewards in training between a benchmark and Phase 1.

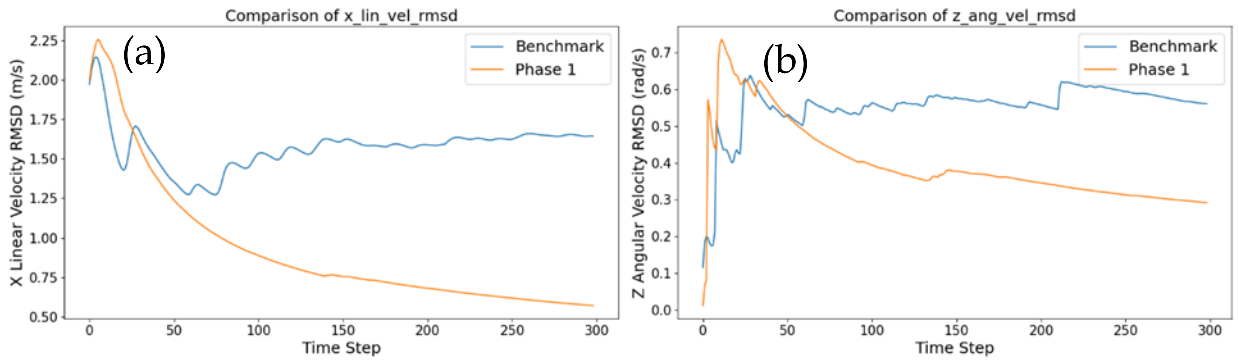

Figure 4.8: Comparative analysis of root-mean-square deviation (RMSD) in (a) X linear and (b) Z angular velocities between a benchmark and Phase 1.

This suggests better performance in tracking linear velocity commands in Phase 1. The angular velocity RMSD plot indicates a similar trend initially, but the Benchmark model exhibits a gradual increase in RMSD over time, suggesting a divergence from desired angular velocities as time progresses.

Energy Consumption and Efficiency:

The most striking difference between the two models is observed in the power consumption and Cost of Transport (CoT). It is a measure often used to assess the energy efficiency of moving an entity from one point to another. It is defined as the power consumption divided by the product of mass (taken as 30 Kg), gravitational acceleration, and velocity. Phase 1 model demonstrated a significant reduction in average power consumption, with a mean of 179.13, compared to the benchmarked model's mean of 309.92.

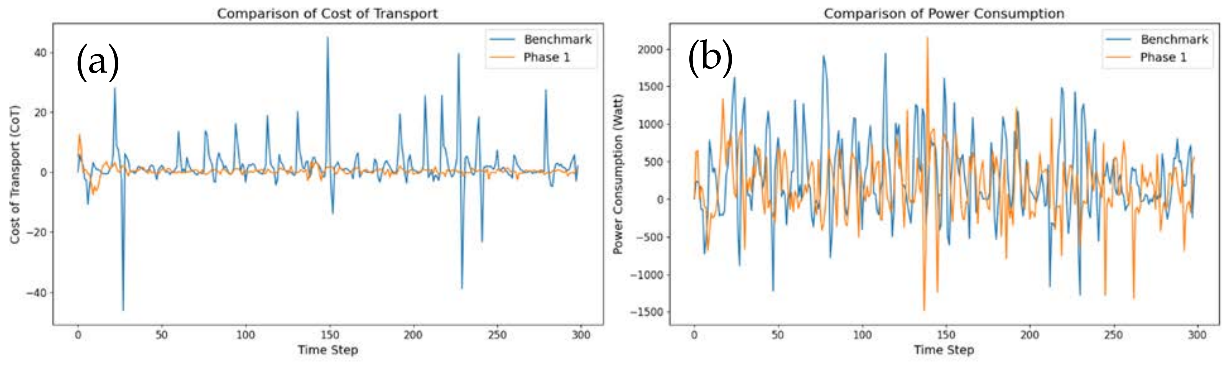

Figure 4.9: Graphs showing a comparison of (a) Cost of Transport and (b)Power Consumption between a benchmark and Phase 1.

This reduction suggests a more energy-efficient locomotion strategy, further supported by the CoT metrics, where the Phase 1 model averaged 0.38 versus the benchmarked model's 2.00. This reduction suggests enhanced efficiency and energy utilization in the Phase 1 Model, likely attributable to the introduced adjustments in its reward structure. The Phase 1 model's adjustments have led to a more disciplined approach towards achieving efficient locomotion, reducing unnecessary movements (evident from lower average torques and joint accelerations).

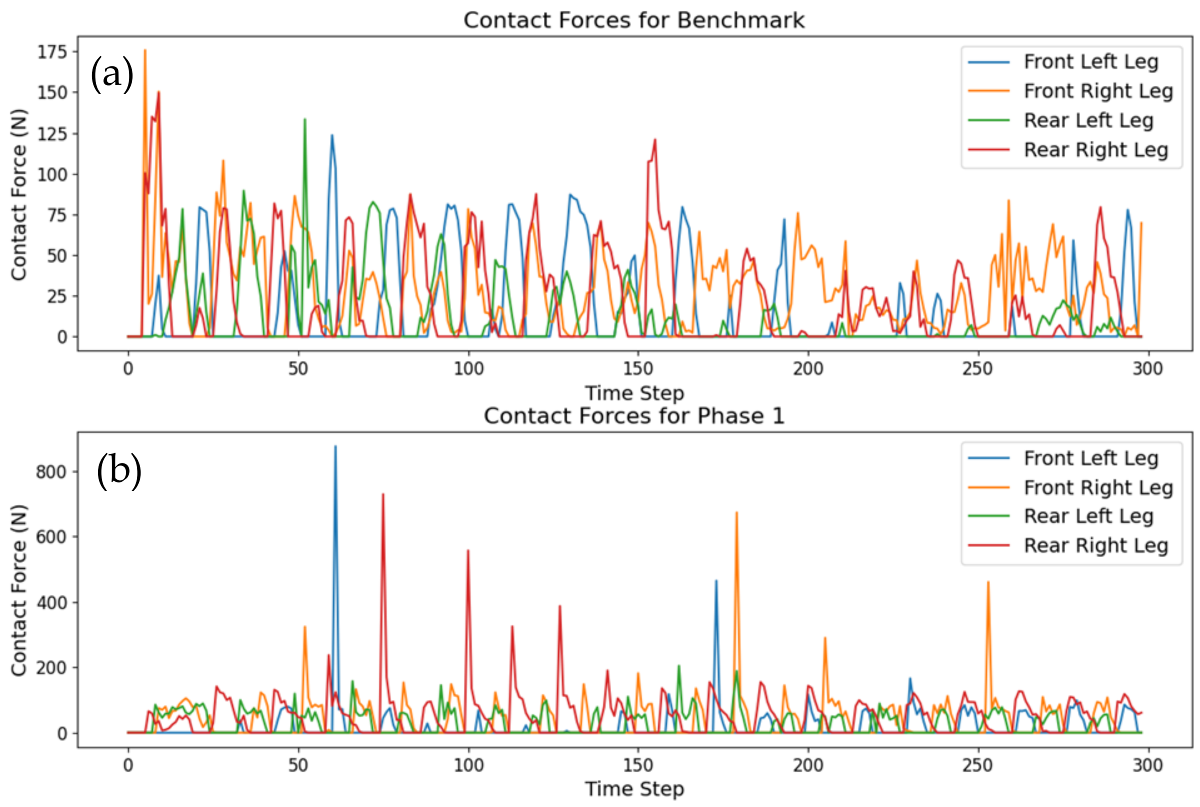

Figure 4.10: Comparative contact force measurements for each leg of a quadruped robot during (a) benchmark test and (b) Phase 1.

Contact Forces:

Interestingly, the contact forces metrics show the Phase 1 model engaging in interactions with the ground that are on average stronger than those of the benchmarked model. Mean contact forces increased from 18.27 (Benchmarked) to 30.87 (Phase 1) which could indicate a more assertive gait pattern encouraged by the new rewards, potentially contributing to the improved energy efficiency by reducing slippage and missteps. A broader range of interactions is observed in the Phase 1 model, suggesting a more dynamic engagement with the environment.

Thus, the introduction of the contact force and stumble rewards in the Phase 1 model, alongside the removal of the airtime reward, showcases a targeted approach to enhance stability and energy efficiency during locomotion.

4.3 Phase 2 Training: Incorporating Particles for Granular Terrain Simulation

To accommodate the intricate dynamics of particle interactions and the reduced size of the training terrains, a few adjustments were made:

- **Episode Length Adjustment:** Recognizing the reduced length and width of the terrains in Phase 2, we modified the episode length from 20 to 10 seconds for maximum interaction with particles.
- **Reduction in Robot Count:** Corresponding to the decrease in the number of terrains available for training, the total number of robots engaged in parallel training sessions was reduced from 2048 to 256. In alignment with this reduction, the mini-batch size for training was also decreased by 1/8th.
- **Training Epochs:** On top of the 2000 epochs completed in Phase 1, the model underwent an additional 1000 epochs of training in Phase 2.

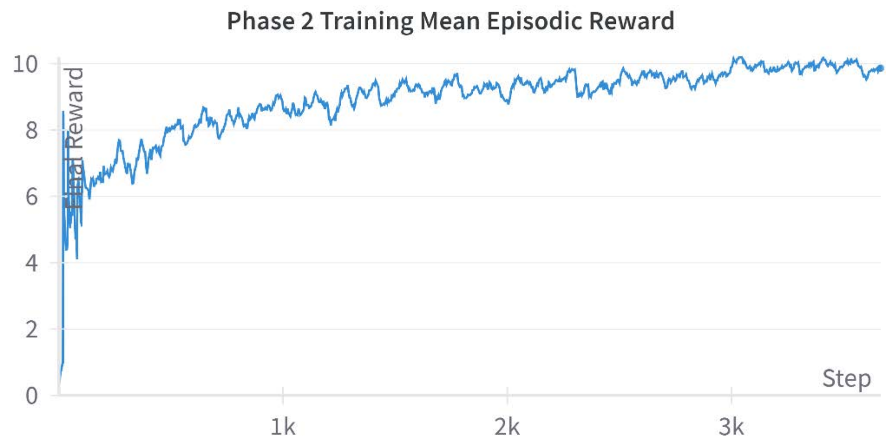

Figure 4.11: Average episodic total reward curve for Phase 2. Note that direct comparison with Phase 1 is not feasible due to the episode length being halved from 20 secs to 10 secs.

Contact Forces: Phase 2 shows a slight decrease in the mean contact forces compared to Phase 1. This could be due to the particle-filled terrain's nature, where the interaction dynamics between the robot's feet and the ground are more complex and unpredictable. The varied contact forces suggest that Phase 2 adapts to maintaining stability and traction in a more challenging environment.

Power Consumption and Cost of Transport (CoT): While Phase 2 maintains similar levels of performance in velocity control and actuator efficiency, The power consumption and CoT are slightly higher with a mean of 199.89 in Phase 2 than in Phase 1, indicating a potential increase in energy expenditure, suggesting the inherent challenge in navigate the particle-filled terrain.

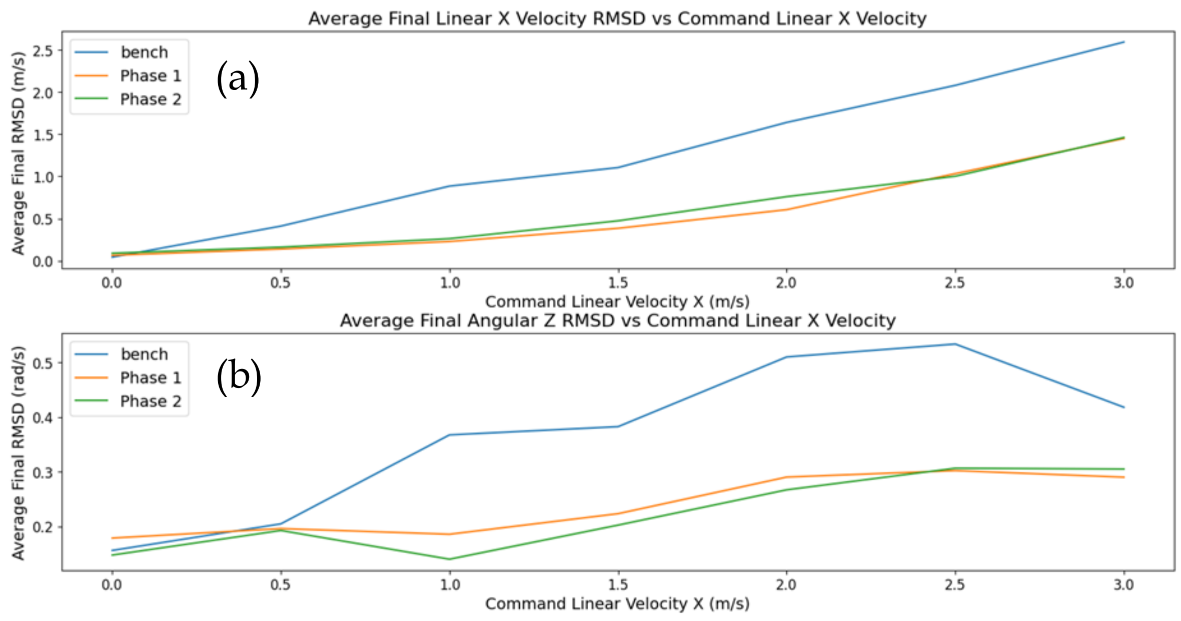

Figure 4.12: Plots depicting the relationship between the (a)average final linear X velocity RMSD and commanded linear X velocity, and (b)average final angular Z RMSD versus commanded linear X velocity across benchmark, Phase 1, and Phase 2. The commanded linear X velocity was incrementally increased from 0 to 3 m/s in steps of 0.5, and the commanded yaw (angular Z) velocity varied from -0.6 to 0.6 rad/s in increments of 0.2.

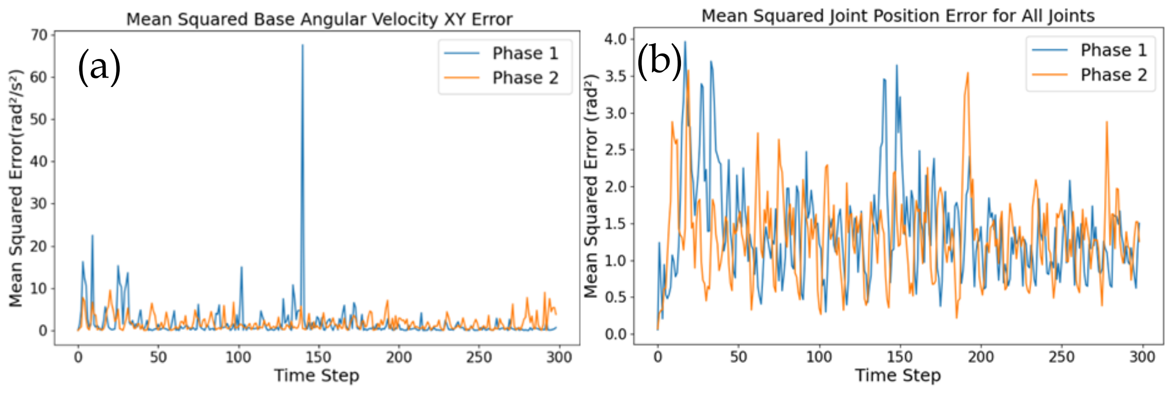

Figure 4.13: Graph illustrating (a)the mean squared error of base angular velocity in the XY plane (b) mean squared error in joint position for all joints for a quadruped robot across Phase 1 and Phase 2.

Fig. 4.12 plots depict the relationship between the command linear velocity in the x-direction and the average final root mean square deviation (RMSD) for both linear and angular velocities, comparing the benchmarked model with Phase 1 and Phase 2 models.

The transition from rigid to particle-filled terrain in Phase 2 training does not show a significant improvement in the overall velocity tracking metrics compared to Phase 1. From Fig. 4.12(a), as the command linear velocity increases, the average final linear RMSD also increases for all models. This trend suggests that as robots are commanded to move faster, the deviation from the desired trajectory increases, which is expected due to the increased difficulty in maintaining precise control at higher speeds. A similar trend is observed for angular RMSD in Fig. 4.12(b), where an increase in command linear velocity correlates with an increase in angular RMSD. This indicates that faster linear movements are associated with greater challenges in maintaining angular stability.

The Phase 1 model demonstrates a substantially flatter error curve compared to the benchmark, indicating that the RMSE remains relatively constant regardless of changes in commanded linear velocity. This stability in the RMSE suggests that the modifications in Phase 1, such as the removal of the airtime reward and the addition of contact force and stumble rewards, may have contributed to a more consistent and robust control over the robot's linear velocity. Phase 2 model maintains a similarly low and stable RMSE across all commanded velocities, closely mirroring the performance of Phase 1.

However, a notable trend in Fig 4.13 is observed in the mean squared sum errors of the base angular velocity in the XY plane. In Phase 1, the mean squared sum error stands at 1.7539 with a high standard deviation of 4.7669, indicating a considerable variance in performance across different episodes. However, in Phase 2, not only does the mean squared sum error decrease to 1.6875, but the standard deviation also significantly reduces to 1.7753. Similarly, when analyzing the mean squared error for all joints (Degrees of Freedom or DOFs), a similar improvement is seen from Phase 1 to Phase 2. The mean squared error reduces from 1.3894 in Phase 1 to

1.3297 in Phase 2, accompanied by a decrease in standard deviation from 0.6691 to 0.5849. The reduction in DOF Positions Squared Error from Phase 1 to Phase 2 indicates an improvement in the robot's ability to accurately achieve target joint positions. The improvement seen in Angular Velocity XY Squared Errors signifies better control over the robot's orientation and turning movements. Efficient control over angular velocity is vital for maintaining balance, performing sharp turns, and correcting orientations in response to dynamic environmental changes.

Even though the linear velocity RMSD did not show marked improvement, the enhancements in joint position accuracy and angular velocity control contribute to the overall locomotion stability and robustness of the robot. These improvements facilitate smoother transitions between movements, reduce the risk of falls or missteps, and improve the robot's ability to maintain a desired trajectory or orientation under varying conditions. In essence, the robot becomes more adaptable and reliable, qualities that are crucial for real-world applications.

4.4 Discussion:

The transition to Phase 2 does not yield significant velocity tracking improvements over Phase 1, where several factors could be at play. The balance between computational efficiency and learning effectiveness is critical. The reductions in robots, minibatch size, environment length, and the number of terrains, while beneficial for computational manageability, might have constrained the diversity and complexity of the learning scenarios encountered by the robot. Additionally, the shortened episode length could constrain the learning of long-term locomotion strategies. While Phase 2 aims to make the training process more computationally manageable, it is essential to ensure that these optimizations do not significantly hinder the model's ability to learn and adapt to complex locomotion challenges.

As the model becomes more optimized in Phase 1, additional gains may become increasingly difficult to achieve, resulting in smaller improvements during subsequent training phases. With a more concise and potentially less diverse training environment, there is a risk that the model may become overfit to the specific scenarios it was exposed to, potentially reducing its generalization to new terrains or conditions.

Also, the inherent difficulty of adapting to highly variable and unpredictable terrains demands for more specialized training strategies, or the limitations of the current reward structure in capturing the nuances of terrain interaction. The adjustments in rewards between the phases were critical in directing the learning focus. However, the balancing of rewards, particularly with the introduction of new metrics like contact forces and stumble prevention, requires fine-tuning to ensure that each aspect contributes optimally to the overall learning goal. Future work could explore alternative training methodologies, reward structures, or model architectures to better equip the robot for efficient locomotion in complex terrains.

Chapter 6

Conclusion

This thesis has provided an extensive examination of the challenges and potentials in the domain of quadruped robot locomotion within simulated environments, with a focus on particle-based dynamics. A comprehensive approach was adopted, starting with the motivation and fundamental objectives, leading up to a detailed exploration of methodologies involving RL and dynamic terrain generation within simulation platforms like Isaac Sim and Warp.

Through rigorous training phases and methodological curriculum design, the ability of quadruped robots to navigate complex terrains has been enhanced. Benchmarking against existing work, we've delineated clear improvements and acknowledged the inherent challenges.

6.1 Future Work

The research paves the way for a number of exciting avenues for future investigation.

Domain Adaptation: Implementing privileged learning through student-teacher policies could bridge the gap between simulation and real-world application more efficiently.

Multi-Modal Locomotion Strategies: Diversifying the gaits and movement strategies, such as walking, trotting, and galloping, can equip robots with a versatile toolkit for navigating diverse environments.

Reward Function Exploration: Exploring additional reward terms and fine-tuning existing ones could further enhance the training process, potentially leading to more natural and efficient locomotion.

Training with Particles from the Start: Initiating training in particle-based environments may lead to more grounded behaviors that account for complex interactions from the outset.

Efficient Algorithms: Investigating other algorithms like Soft Actor-Critic (SAC) could improve sample efficiency, particularly when the number of robots or computational resources is limited.

Real-World Transfer: Transferring our simulation-trained models to physical hardware will be a critical step in assessing the viability of our methods in real-world scenarios.

Hardware Iteration: The feedback loop between simulation findings and hardware design can lead to more resilient and capable robots, capable of handling the unpredictability of real-world interactions.

The overarching goal will remain to build robots that can assist in a variety of tasks, ranging from search and rescue operations to planetary exploration. The adaptability and resilience of such machines are crucial, and the work presented here lays a foundational framework that future research can build upon to realize these ambitious goals.

A. Bibliography

[1] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, “Dynamic Locomotion on Slippery Ground,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4170–4176, Oct. 2019, doi: 10.1109/LRA.2019.2931284.

[2] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim, “Contact Model Fusion for Event-Based Locomotion in Unstructured Terrains,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD: IEEE, May 2018, pp. 4399–4406. doi: 10.1109/ICRA.2018.8460904.

[3] M. Focchi *et al.*, “Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality,” vol. 132, 2020, pp. 165–209. doi: 10.1007/978-3-030-22327-4_9.

[4] J. Reher, W.-L. Ma, and A. D. Ames, “Dynamic Walking with Compliance on a Cassie Bipedal Robot.” arXiv, Apr. 24, 2019. doi: 10.48550/arXiv.1904.11104.

[5] Y. Gong *et al.*, “Feedback Control of a Cassie Bipedal Robot: Walking, Standing, and Riding a Segway.” arXiv, Sep. 19, 2018. doi: 10.48550/arXiv.1809.07279.

[6] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter, “Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon: IEEE, Oct. 2016, pp. 3872–3878. doi: 10.1109/IROS.2016.7759570.

[7] M. Focchi, V. Barasuol, M. Frigerio, D. G. Caldwell, and C. Semini, “Slip Detection and Recovery for Quadruped Robots,” in *Robotics Research: Volume 2*, A. Bicchi and W. Burgard, Eds., in Springer Proceedings in Advanced Robotics. , Cham: Springer International Publishing, 2018, pp. 185–199. doi: 10.1007/978-3-319-60916-4_11.

[8] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, “State estimation for legged robots on unstable and slippery terrain,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 6058–6064. doi: 10.1109/IROS.2013.6697236.

[9] C. Gehring *et al.*, “Dynamic trotting on slopes for quadrupedal robots,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany: IEEE, Sep. 2015, pp. 5129–5135. doi: 10.1109/IROS.2015.7354099.

[10] R. Hartley *et al.*, “Legged Robot State-Estimation Through Combined Forward Kinematic and Preintegrated Contact Factors,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD: IEEE, May 2018, pp. 4422–4429. doi: 10.1109/ICRA.2018.8460748.

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Sci. Robot.*, vol. 5, no. 47, p. eabc5986, Oct. 2020, doi: 10.1126/scirobotics.abc5986.

[12] A. Kumar, Z. Fu, D. Pathak, and J. Malik, “RMA: Rapid Motor Adaptation for

Legged Robots." arXiv, Jul. 08, 2021. Accessed: Sep. 25, 2023. [Online]. Available: http://arxiv.org/abs/2107.04034

[13] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent Training of a Control Policy and a State Estimator for Dynamic and Robust Legged Locomotion," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4630–4637, Apr. 2022, doi: 10.1109/LRA.2022.3151396.

[14] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the MIT Cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, Feb. 2017, doi: 10.1177/0278364917694244.

[15] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, p. eabk2822, Jan. 2022, doi: 10.1126/scirobotics.abk2822.

[16] H. Kolvenbach *et al.*, "Traversing Steep and Granular Martian Analog Slopes With a Dynamic Quadrupedal Robot." arXiv, Jun. 03, 2021. Accessed: Mar. 11, 2024. [Online]. Available: http://arxiv.org/abs/2106.01974

[17] N. Mazouchova, N. Gravish, A. Savu, and D. I. Goldman, "Utilization of granular solidification during terrestrial locomotion of hatchling sea turtles," *Biol Lett*, vol. 6, no. 3, pp. 398–401, Jun. 2010, doi: 10.1098/rsbl.2009.1041.

[18] "Sidewinding with minimal slip: Snake and robot ascent of sandy slopes | Science." Accessed: Mar. 11, 2024. [Online]. Available: https://www.science.org/doi/10.1126/science.1255718

[19] "The Soft-Landing Problem: Minimizing Energy Loss by a Legged Robot Impacting Yielding Terrain | IEEE Journals & Magazine | IEEE Xplore." Accessed: Mar. 11, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9018262

[20] V. Vasilopoulos, I. S. Paraskevas, and E. G. Papadopoulos, "Compliant terrain legged locomotion using a viscoplastic approach," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA: IEEE, Sep. 2014, pp. 4849–4854. doi: 10.1109/IROS.2014.6943251.

[21] C. M. Hubicki, J. J. Aguilar, D. I. Goldman, and A. D. Ames, "Tractable terrain-aware motion planning on granular media: An impulsive jumping study," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 3887–3892. doi: 10.1109/IROS.2016.7759572.

[22] A. H. Chang, C. M. Hubicki, J. J. Aguilar, D. I. Goldman, A. D. Ames, and P. A. Vela, "Learning Terrain Dynamics: A Gaussian Process Modeling and Optimal Control Adaptation Framework Applied to Robotic Jumping," *IEEE Trans. Contr. Syst. Technol.*, vol. 29, no. 4, pp. 1581–1596, Jul. 2021, doi: 10.1109/TCST.2020.3009636.

[23] "Learning agile and dynamic motor skills for legged robots." Accessed: Mar. 11, 2024. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.aau5872

[24] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to

Walk via Deep Reinforcement Learning." arXiv, Jun. 19, 2019. doi: 10.48550/arXiv.1812.11103.

[25] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning Locomotion Skills for Cassie: Iterative Design and Sim-to-Real," in *Proceedings of the Conference on Robot Learning*, PMLR, May 2020, pp. 317–329. Accessed: Mar. 11, 2024. [Online]. Available: https://proceedings.mlr.press/v100/xie20a.html

[26] J. Lee, J. Hwangbo, and M. Hutter, "Robust Recovery Controller for a Quadrupedal Robot using Deep Reinforcement Learning." arXiv, Jan. 22, 2019. doi: 10.48550/arXiv.1901.07517.

[27] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning." arXiv, May 18, 2021. doi: 10.48550/arXiv.2105.08328.

[28] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning Fast Adaptation with Meta Strategy Optimization." arXiv, Feb. 15, 2020. doi: 10.48550/arXiv.1909.12995.

[29] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged Locomotion in Challenging Terrains using Egocentric Vision." arXiv, Nov. 14, 2022. Accessed: Apr. 08, 2023. [Online]. Available: http://arxiv.org/abs/2211.07638

[30] "Per-Contact Iteration Method for Solving Contact Dynamics | IEEE Journals & Magazine | IEEE Xplore." Accessed: Mar. 11, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8255551

[31] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, in ICML '09. New York, NY, USA: Association for Computing Machinery, Jun. 2009, pp. 41–48. doi: 10.1145/1553374.1553380.

[32] R. Li, A. Jabri, T. Darrell, and P. Agrawal, "Towards Practical Multi-Object Manipulation using Relational Reinforcement Learning." arXiv, Dec. 23, 2019. doi: 10.48550/arXiv.1912.11032.

[33] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacher–Student Curriculum Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3732–3740, Sep. 2020, doi: 10.1109/TNNLS.2019.2934906.

[34] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning." arXiv, Aug. 19, 2022. Accessed: Sep. 10, 2023. [Online]. Available: http://arxiv.org/abs/2109.11978

[35] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 3803–3810. doi: 10.1109/ICRA.2018.8460528.

[36] R. Kaushik, T. Anne, and J.-B. Mouret, "Fast Online Adaptation in Robotics through Meta-Learning Embeddings of Simulated Priors." arXiv, Jan. 06, 2021. doi: 10.48550/arXiv.2003.04663.

[37] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by Cheating."

arXiv, Dec. 27, 2019. doi: 10.48550/arXiv.1912.12294.
[38] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 5026–5033. doi: 10.1109/IROS.2012.6386109.
[39] "Bullet Real-Time Physics Simulation | Home of Bullet and PyBullet: physics simulation for games, visual effects, robotics and reinforcement learning." Accessed: Mar. 14, 2024. [Online]. Available: https://pybullet.org/wordpress/
[40] V. Makoviychuk *et al.*, "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning." arXiv, Aug. 25, 2021. doi: 10.48550/arXiv.2108.10470.
[41] P. A. Cundall and O. D. L. Strack, "A discrete numerical model for granular assemblies," *Géotechnique*, vol. 29, no. 1, pp. 47–65, Mar. 1979, doi: 10.1680/geot.1979.29.1.47.
[42] H. He, J. Zheng, Y. Chen, and Y. Ning, "Physics engine based simulation of shear behavior of granular soils using hard and soft contact models," *Journal of Computational Science*, vol. 56, p. 101504, Nov. 2021, doi: 10.1016/j.jocs.2021.101504.
[43] J. Bender, M. Müller, and M. Macklin, "A Survey on Position Based Dynamics, 2017," 2017.
[44] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, Apr. 2007, doi: 10.1016/j.jvcir.2007.01.005.
[45] "ZSL-RPPO: Zero-Shot Learning for Quadrupedal Locomotion in Challenging Terrains using Recurrent Proximal Policy Optimization." Accessed: Mar. 14, 2024. [Online]. Available: https://arxiv.org/html/2403.01928v1
[46] M. Shafiee, G. Bellegarda, and A. Ijspeert, "DeepTransition: Viability Leads to the Emergence of Gait Transitions in Learning Anticipatory Quadrupedal Locomotion Skills." arXiv, Jun. 14, 2023. Accessed: Mar. 14, 2024. [Online]. Available: http://arxiv.org/abs/2306.07419
[47] "Deformable-Body Simulation — Omniverse Extensions latest documentation." Accessed: Mar. 14, 2024. [Online]. Available: https://docs.omniverse.nvidia.com/extensions/latest/ext_physics/deformable-bodies.html
[48] "Warp (Preview) — Omniverse Extensions latest documentation." Accessed: Mar. 14, 2024. [Online]. Available: https://docs.omniverse.nvidia.com/extensions/latest/ext_warp.html
[49] "NVIDIA-Omniverse/OmniIsaacGymEnvs: Reinforcement Learning Environments for Omniverse Isaac Gym." Accessed: Mar. 14, 2024. [Online]. Available: https://github.com/NVIDIA-Omniverse/OmniIsaacGymEnvs
[50] "leggedrobotics/legged_gym: Isaac Gym Environments for Legged Robots." Accessed: Mar. 14, 2024. [Online]. Available: https://github.com/leggedrobotics/legged_gym