GaitNet: Greedy, Acyclic Quadruped Gait Generation

Owen Sullivan

Supervisors:
Prof. Mahdi Agheli

Worcester Polytechnic Institute

November 30, 2025

Abstract

Quadruped locomotion across unstructured terrain remains a longstanding challenge in robotics, requiring controllers that can adapt dynamically to complex and unpredictable environments. Traditional approaches rely on model predictive control or pre-defined gait schedules, offering stability but limited adaptability. Conversely, end-to-end learning methods enable agile and versatile behaviors but often sacrifice interpretability, safety, and real-time feasibility. This thesis introduces *GaitNet*, a hybrid control framework that integrates a greedy, neural network-based gait planner with a traditional model-based controller to generate dynamic, acyclic locomotion for quadruped robots.

The proposed system consists of two primary components: a *Footstep Evaluation Network* that learns terrain-aware footstep cost maps, and *GaitNet*, a reinforcement learning-based gait selector that ranks and executes feasible footstep actions in real time. Together, these modules enable a balance between the adaptability of learning-based control and the robustness of analytical motion planning. Trained in NVIDIA Isaac Lab using parallel GPU simulation, the system is evaluated across a range of terrain difficulties and commanded velocities.

Experimental results show that *GaitNet* achieves a 69.4% mean survival rate in challenging terrain, outperforming a single-leg motion planner baseline by more than a factor of two. Ablation studies further reveal that dynamic swing duration offers limited benefit in static environments, while removing pre-computed footstep costs leads to improved performance and more efficient gait patterns. These findings highlight the advantages of direct policy learning for coordinated, multi-leg motion without overreliance on heuristic priors.

Overall, this work demonstrates that greedy, neural network-based planning can produce dynamic, non-gaited quadruped motion. The presented framework bridges the gap between reactive learning and structured control, providing a foundation for future research toward fully autonomous, terrain-adaptive legged locomotion in real-world settings.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
DQN	**D**eep **Q** **N**etwork
MCTS	**M**onte **C**arlo **T**ree **S**earch
ML	**M**achine **L**earning
MLP	**M**ulti-**L**ayer **P**erceptron
MPC	**M**odel **P**redictive **C**ontrol
PPO	**P**roximal **P**olicy **O**ptimization
RL	**R**einforcement **L**earning

List of Symbols

Symbol[2]	Description	Units[3]
$\mathbf{f}_c$	Footstep candidate Vector of *leg index* (or -1 for no leg), *dx*, *dy*, and *cost*. With *dx* and *dy* being offsets from the nominal foot position in the base frame.	$\begin{bmatrix}\text{-} & \text{m} & \text{m} & \text{-}\end{bmatrix}^T$
$\mathbf{f}_a$	Footstep action Vector of *leg index* (or -1 for no leg), *dx*, *dy*, and *swing duration*. With *dx* and *dy* being offsets from the nominal foot position in the base frame.	$\begin{bmatrix}\text{-} & \text{m} & \text{m} & \text{s}\end{bmatrix}^T$
$\mathbf{r}_w$	COM position in world frame	m
$\mathbf{q}_{rw}$	Root orientation (quaternion) in world frame	-
$\mathbf{v}_b$	Root linear velocity in base frame	$\text{m}\,\text{s}^{-1}$
ω_b	Root angular velocity in base frame	$\text{rad}\,\text{s}^{-1}$
$\mathbf{p}_{ib}$	End-effector i position in base frame	m
$\mathbf{g}$	Gravity vector in base frame	$\text{m}\,\text{s}^{-2}$
$\mathbf{c}_i$	End-effector i contact state	-
$\mathbf{q}$	Joint positions	-
$\dot{\mathbf{q}}$	Joint velocities	-
$\ddot{\mathbf{q}}$	Joint accelerations	-
τ	Joint torques	$\text{N}\,\text{m}$
$\mathbf{u}$	Control Input Vector of v_x, v_y, and ω_z	$\begin{bmatrix}\text{m}\,\text{s}^{-1} & \text{m}\,\text{s}^{-1} & \text{rad}\,\text{s}^{-1}\end{bmatrix}^T$

[2] A symbol defined with a subscript i, but displayed without it indicates the concatenation of all i elements, e.g. $\mathbf{c} = \begin{bmatrix}\mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & \mathbf{c}_4\end{bmatrix}^T$

[3] Units marked with "-" indicate either no unit or multiple units

Terminology

Within legged robotics, there are many conflicting or ambiguous terms. For clarity, the exact meanings of specific terms used in this paper are described below.

Term	Definition
Footstep planning	The process of selecting when and where to step.
Greedy	In the context of footstep planning, a planner which only provides instantaneous actions.
Dynamic locomotion/gait	The opposite of quasi-static locomotion; where balance is maintained through active control and dynamic interactions, not slow and stable placements. Sometimes as an adjective, dynamic (as in the title).
Acyclic gait	Sometimes "free-gait" or "non-gaited". Refers to a footstep plan which cannot be represented with phase cycles or transitions between gait primitives. The opposite of gaited or gait-based locomotion.

1 Introduction

Quadruped locomotion in unstructured environments remains a significant challenge, particularly when traditional gait cycles prove inadequate [3]. While many existing systems rely on periodic gait patterns or centralized planners [3], recent advances in learning-based methods have enabled more adaptive, non-gaited approaches [4]. This study proposes a novel method for generating non-gaited, dynamic footstep plans using a neural network-based greedy planner, aiming to achieve a balance between computational efficiency and dynamic performance.

1.1 Motivation and Vision

Legged robots hold significant promise for enabling mobility in environments that are inaccessible to wheeled and tracked systems. Their ability to traverse uneven, unstructured, and unpredictable terrains—such as disaster zones, staircases, dense vegetation, and rocky landscapes—positions them as key enablers for future exploration, inspection, and rescue missions.

Realizing this potential requires a robot to continuously and intelligently decide where and when to place its feet—a process known as *contact planning*. Unlike periodic gait patterns suitable for flat or predictable surfaces, real-world locomotion often demands dynamic, acyclic, and adaptive footstep sequences. Even minor errors in contact selection can lead to instability or failure, whereas well-chosen footholds enable efficient, agile, and robust movement.

The generation of such contact plans in real time remains a formidable challenge. It requires the system to perceive its surroundings, anticipate the outcomes of possible actions, and select feasible contact points within tight temporal constraints. Existing methods often fall into one of three categories, fully using traditional control—model predictive controllers, PD controllers, optimization—using neural networks, or a mix of the two: hybrid methods. These three methods offer different trade offs between deliberative, computationally intensive planning and fast, reactive control that lacks stability or generalization.

The vision of this research is to develop a hybrid control architecture that optimizes these trade offs. Specifically, this work aims to design a system capable of producing non-gaited, dynamic footstep plans that retain both the responsiveness of learning-based approaches and the reliability of model-based control. The objective is to provide quadruped robots with the decision-making agility necessary for navigating complex terrains while maintaining the stability and robustness demanded by real-world operation.

Through this approach, the thesis seeks to contribute toward a new class of hybrid locomotion frameworks that enable real-time, adaptive, and physically consistent quadruped motion—closing the gap between high-level perception and low-level control in legged robotics.

1.2 Research Gap

Quadruped control pipelines can be categorized into several approaches, many of which can be broadly classified as traditional, neural network-based, or hybrid methods.

Traditional approaches typically rely on model predictive control (MPC) frameworks for motion optimization, supported by lower-level controllers to track desired trajectories [5]. Many aspects of quadruped locomotion—such as joint torques and ground reaction forces—are smooth and well-suited to continuous optimization techniques [6]. However, these methods become less tractable when discrete contact states

are introduced into the optimization problem [7]. To manage this complexity, most implementations either assume pre-defined gait sequences [3, 4] or encode discrete contact decisions within larger optimization structures [8]. While effective in structured environments, these strategies constrain the robot's ability to exploit the full versatility of legged locomotion by reducing the dimensionality of expressable states for the system.

Neural network-based methods have emerged as an alternative, particularly with advances in deep reinforcement learning. These approaches often replace the MPC component of traditional control pipelines, learning to directly map observed states to joint torques or motion commands [9]. In doing so, they inherently address the mixed-integer nature of contact planning, allowing the network to learn both continuous and discrete aspects of locomotion simultaneously [6]. Despite their flexibility, such methods generally lack formal guarantees of stability, require substantial training data, and often struggle to generalize beyond their training distribution [9].

Hybrid approaches, which integrate learning-based modules within model-based control frameworks, have recently gained attention as a potential middle ground. In these systems, neural networks are typically employed to solve specific subproblems—such as dynamically switching between fixed gaits or greedily selecting individual footholds—while the overall motion execution remains governed by a traditional controller [10, 11]. This division of responsibilities enables learning to focus on the non-smooth or combinatorial components of locomotion, while preserving the interpretability, safety, and robustness inherent to model-based systems.

Looking at how each of these categories handles the contact planning problem, hybrid control schemes have demonstrated promise, the challenge of contact and footstep planning continues to limit their performance. Traditional optimization-based planners struggle with the discrete nature of contact transitions, resulting in high computational costs that preclude real-time operation [8]. Alternatively, many rely on pre-specified gait patterns [12, 13, 14, 15], constraining adaptability to irregular or unpredictable terrain. Even more advanced approaches, such as Monte Carlo Tree Search (MCTS)-based planners, have achieved impressive dynamic behaviors but remain computationally demanding [16, 17]. Finally, end to end learning methods solve these problems implicitly, but at the cost of control theory guarantees and interpretability.

Despite recent progress, a clear research gap remains. Current hybrid methods do not yet achieve the balance between adaptability and stability that would enable fully dynamic, acyclic locomotion in complex environments. In particular, there is a lack of systems that can match the contact planning agility of neural network-based approaches while retaining the formal guarantees and reliability of traditional control frameworks [18, 6]. Recent work, such as ContactNet [1], has demonstrated the potential of machine learning for footstep planning; however, its design—restricted to single-leg motions with fixed timing—still falls short of the dynamic, non-gaited behaviors exhibited by end-to-end learning systems.

This motivates a central research question:

> Can a hybrid control pipeline, which integrates a greedy, neural-network-based planner with a traditional model-based controller, generate dynamic, acyclic gaits for robust quadruped locomotion in challenging environments?

1.3 Hypothesis

To address the identified research gap, we hypothesize that a hybrid control pipeline, incorporating a greedy, neural network-based footstep planner and a traditional model-based control framework, can effectively generate dynamic, non-gaited locomotion for quadruped robots in challenging environments. By leveraging the advancements introduced by ContactNet and extending its capabilities to support multi-leg motion with dynamic swing durations, we believe that such a planner can produce footstep sequences that enable the robot to traverse complex terrains while maintaining both stability and efficiency.

1.4 Approach

The proposed approach employs a modified implementation of ContactNet's architecture, adapted specifically to generate footstep candidates. These candidates will be evaluated and ranked by our novel planner, *GaitNet*, which selects the most appropriate action at each time step. By integrating this planner within a traditional

control framework, we aim to harness the advantages of both learning-based and model-based methods, yielding a robust and adaptable quadruped locomotion system.

1.5 Contributions

Quadruped locomotion in unstructured environments demands footstep planners that combine adaptability, computational efficiency, and physical consistency. This thesis addresses the challenge of generating dynamic, acyclic contact sequences by exploring a hybrid control framework that integrates a greedy, neural network-based footstep planner—GaitNet—within a traditional model-based controller. Building on insights from recent learning-driven approaches while maintaining the reliability of structured control, this work aims to close the gap between flexible footstep selection and stable trajectory execution. The following contributions summarize the central advances introduced in this thesis:

- Development of a novel footstep planner utilizing greedy planning with a neural network, for use in hybrid control of quadruped robots.
- Demonstration that the proposed planner can generate dynamic, acyclic gaits in challenging environments.
- Empirical evaluation showing that the planner achieves comparable diverse expressiveness with fast inference times.

1.6 Thesis Structure

The remaining chapters are organized as follows.

Chapter 2 presents the background necessary to contextualize this work, covering fundamental concepts in quadruped locomotion, gait generation, and learning-based control methods. This chapter establishes the theoretical foundations on which the proposed approach is built.

Chapter 3 describes the methodology, detailing the design and implementation of the CNN-based greedy planner, GaitNet, including the underlying network architecture and training pipeline. It explains how this module integrates with a model-based control framework to enable dynamic footstep planning.

Chapter 4 reports the experimental results and evaluates the performance of the proposed planner across a range of challenging scenarios. This chapter also discusses the strengths and limitations of the method, examining how the system behaves under diverse conditions.

Chapter 5 concludes the thesis by summarizing the key findings and reflecting on the broader implications for quadruped locomotion research. It also identifies potential directions for future work, particularly avenues for improving adaptability, stability, and real-world deployment.

2 Background

Recent advances in legged locomotion have largely focused on either optimization-based motion planning or learning-based policy generation. The former, categorized as traditional control approaches, offers formal guarantees but faces significant computational hurdles when dealing with dynamic, non-gaited footstep selection. The latter, end-to-end neural networks, provides adaptability and implicitly solves the discrete contact problem, but lacks control theory guarantees and interpretability.

The most promising efforts to bridge this gap are hybrid methods, which integrate learning modules within model-based frameworks. Our proposed approach seeks to enhance this hybrid strategy by developing a computationally efficient, non-gaited footstep planner. This chapter reviews the foundations upon which our method builds: the limitations of traditional control and end-to-end Neural Networks, and the state-of-the-art in hybrid methods, including learned gait policies and explicit footstep planners. This review establishes the necessity of our novel hybrid architecture.

2.1 Traditional Control Approaches

For the purpose of this thesis, we define traditional control approaches as those that rely on explicit optimization of footstep positions, contact sequences, and timing, often through mixed-integer programming or search-based methods. These approaches provide guarantees associated with control theory but often suffer from high computational costs and limited real-time applicability, especially for methods which attack the non-gaited footstep planning problem.

Deits and Tedrake [19] formulate footstep placement as a mixed-integer quadratic program over convex terrain regions, enabling robust bipedal foothold selection. However, their method does not optimize contact timing and is limited to sequential foot placement, restricting agility and generalization to more complex contact sequences.

Extensions to this framework incorporate contact scheduling into the optimization itself. Winkler et al. [8] fully optimize contact duration and ordering, allowing multiple steps to be planned simultaneously rather than sequentially. While this increases flexibility, the resulting computation is too slow for real-time control. Similarly, Aceituno-Cabezas et al. [20] integrate footstep positions and gait timings into a mixed-integer convex program based on centroidal dynamics, achieving simultaneous contact and motion planning. Taouil et al. [**?**] further explore non-gaited footstep planning via MCTS, generating dynamic multi-step sequences in real time, though computation remains high due to the branching factor. Akizhanov et al. [21] improve MCTS efficiency using a learned classifier to prune contact configurations and a target adjustment network to compensate for low-level control errors, but the approach is still computationally intensive.

These methods provide precise, globally consistent locomotion plans, but high computational costs and limited real-time applicability restrict their use in dynamic, acyclic gait scenarios.

2.2 End-to-End Neural Networks

End-to-end neural network approaches for legged locomotion directly map sensory inputs to control outputs, bypassing explicit modeling of dynamics or footstep planning.

Zhang et al. [22] propose an end-to-end RL approach mapping proprioceptive input directly to joint targets, reaching 2.5 m/s and generalizing across terrains via a terrain curriculum and curiosity rewards.

However, such models require slow training, and re-training for new environments. Zhang et al. [23] address multi-gait transitions using heuristically defined tables between similar gaits, improving flexibility but still constraining behaviors to predefined families.

While these models achieve impressive feats, they often require extensive training and struggle to generalize to unseen terrains or tasks. This motivates approaches that combine the adaptability of learning-based methods with the structure and guarantees of traditional control, enabling fast, terrain-adaptive footstep planning without the need for extensive retraining.

2.3 Hybrid Methods

Hybrid methods for legged locomotion combine elements of traditional control (e.g., optimization or Model Predictive Control, MPC) and learning-based approaches to achieve fast, terrain-adaptive footstep generation. These methods often solve high-level planning problems with learned models, while delegating low-level control to traditional, high-frequency solvers. This division of labor enables real-time performance without sacrificing the expressiveness offered by deep learning.

2.3.1 Learned Policies for Gait and Trajectory Modulation

A major class of hybrid methods utilizes deep learning to generate agile, robust locomotion policies, often focusing on body motion and gait modulation rather than explicit, step-by-step foot placement. These approaches typically learn a mapping from sensor data to control commands, relying on fixed or implicit gait patterns.

Deep learning has shown strong potential for generating such robust policies. Shi et al. [24] modulate trajectory generator parameters in real time for energy-efficient walking, while Xie et al. [12] train Reinforcement Learning (RL) policies on centroidal dynamics models to output body accelerations under fixed gait heuristics. Other methods focus on blind, proprioceptive-based control: Duan et al. [25] learn step-to-step transitions using proprioception, and Siekmann et al. [26] achieve blind stair climbing through LSTM-based proprioceptive policies. Lee et al. [14] even infer terrain structure from proprioceptive history using a temporal CNN and automated curriculum learning.

Subsequent works extend these ideas through explicit high-level gait selection. Da et al. [27] use a Deep Q-Network (DQN) to choose among predefined gait primitives executed by a low-level controller, while Yang et al. [28] learn policies that output gait parameters—frequency, swing ratio, and phase offsets—interpreted by a phase integrator. Both enable efficient gait modulation but generally assume flat terrain and heuristic foot placement. Sun et al. [29] integrate offline gait optimization with contact-implicit trajectory optimization and high-frequency MPC, achieving dynamic control but limiting adaptability to a set of discrete, speed-dependent gaits.

Although effective at generating robust body motions, these methods primarily rely on fixed or implicit gait patterns and largely lack explicit, flexible control over individual footsteps based on local terrain features.

2.3.2 Learning-Based Footstep Planners

In contrast to gait modulation, learning-based footstep planners focus on explicitly learning the footstep planning component. These methods combine perception and control through data-driven models, often coupling the learned components with downstream MPC or search-based planning to enforce dynamic feasibility.

Early methods demonstrated the concept but faced scope limitations. FootstepNet [30] generates heuristic-free plans via deep RL but is limited to bipeds in simple 2D environments. DeepLoco [31] introduces a hierarchical policy for footstep and motion planning, yet its coarse terrain input and biped focus limit dynamic adaptability. Other approaches separate the planning concerns: DeepGait [32] separates footstep planning from motion optimization, enabling modularity but relying on static gaits. RLOC [33] uses a convolutional encoder to pass terrain data embeddings to multiple RL modules before generating joint torques with a traditional whole body controller. One of these RL modules is a footstep planner, but the system uses a fixed gait schedule and the footstep planner is only evaluated once at the beginning of each cycle.

A key challenge is moving beyond fixed gaits while maintaining stability. ContactNet [1] uses a CNN to produce non-gaited footstep sequences with greedy selection, though it is constrained to one-leg-at-a-time

motion and coarse foothold discretization. Omar et al. [34] and Villarreal et al. [15] use CNNs for terrain classification to inform MPC or heuristic-free selection, but both depend on fixed gait sequences.

Recent Advances in Terrain Generalization

More recent advances expand terrain generalization and multi-contact reasoning. Tolomei et al. [**?**] created a framework generalized to 3D terrain and variable foot shapes, ranking footholds based on terrain curvature, but still follow a fixed gait. Chen et al. [35] apply a DQN to a hexapod, simultaneously selecting both next foot positions and gait, with fallback to predefined behaviors outside expected states, but motion is constrained to three-leg-at-a-time gaits. Similarly, Yao et al. [36] output both swing leg motions and desired robot pose from a deep RL policy but remain limited to a trotting gait. Meduri et al. [37] use a DQN to learn 3D foothold cost maps, but retain fixed biped gait timing.

Overall, these approaches highlight trade-offs between expressive, non-gaited footstep planning and computational efficiency. CNN-based and RL methods improve terrain generalization, but fixed gaits or sequential leg motions remain common constraints for achieving real-time performance and maintaining stability.

2.3.3 Greedy Planning and Learned Foothold Classification

To complement learning-based approaches, other hybrid methods focus on improving the efficiency of the planning step itself through greedy search or by leveraging learned terrain models for safe foothold identification.

Greedy Planners for Efficiency

Greedy planners offer computationally efficient alternatives to exhaustive search, often prioritizing goal-directed heuristics or local stability metrics. Gao et al. [38] propose GH-QP, a greedy-heuristic hybrid that incorporates expected robot speed into a footstep planning heuristic, though it is limited to bipedal systems and lacks support for dynamic footstep plans. Zucker et al. [39] use A* with a learned terrain cost and Dubins car heuristic for footstep planning, enabling effective search in SE(2) but restricted to static, gated locomotion. Kalakrishnan et al. [40] combine greedy terrain-cost search with a coarse-to-fine pose optimization pipeline, but rely on fixed gait patterns.

It is worth noting that greedy elements are already incorporated into several learning-based methods discussed previously, such as ContactNet [1], DeepLoco [31], and Villarreal et al. [15]. While these works demonstrate the potential of greedy methods for efficiency, none explicitly support both dynamically unstable and non-gaited footstep generation.

Learned Foothold Classification

A related line of research focuses solely on determining safe stepping regions (foothold classification) using learned terrain models to support downstream planners that enforce safety constraints. Asselmeier et al. [41] generate steppability maps from visual inputs to guide a trajectory-optimization-based planner. Omar et al. [42] propose a two-stage classifier that first identifies steppable regions before selecting footholds, emphasizing safety. Similarly, Omar et al. [34] use a CNN to identify safe footholds, which are then passed to an MPC for real-time planning, but the approach relies on fixed gait sequences.

These methods provide robust terrain understanding but are typically designed as standalone perception modules, separate from the full dynamic footstep generation process, and thus are not directly applicable to dynamic or non-gaited motion planning without significant integration.

2.4 Synthesis

The preceding review of quadruped locomotion control, encompassing traditional optimization-based methods, end-to-end neural networks, and hybrid control schemes, reveals a fundamental trade-off: control theory guarantees and robustness versus computational efficiency and adaptability to unstructured, non-gaited motion. Traditional methods provide stability but suffer from high computational cost, particularly when

attempting to solve the mixed-integer programming problem inherent in non-gaited contact planning [8, **?**]. Conversely, end-to-end learning approaches achieve impressive dynamic, acyclic behaviors but lack formal guarantees and generalizability beyond their training distribution [22].

Hybrid control architectures have emerged as the promising middle ground, integrating learned modules within model-based frameworks to balance agility and stability. However, existing hybrid footstep planners, such as ContactNet [1], are often constrained by enforcing sequential, one-leg-at-a-time motion or relying on fixed gait timing. These constraints fundamentally limit the robot's ability to execute the fully dynamic, acyclic locomotion required for optimal performance in complex terrain.

This thesis addresses the identified gap by proposing a novel hybrid architecture built upon the efficiency of greedy, NN-based selection while overcoming the single-leg and fixed-timing limitations of prior work. The *GaitNet* planner is designed to simultaneously evaluate and select footstep candidates for multiple legs with dynamic swing durations. By delegating the computationally intensive, discrete decision-making of contact selection to this neural network and feeding the resulting plans into a robust, model-based controller, we aim to achieve a system that combines the agility and non-gaited expressiveness of learning-based approaches with the stability and physical consistency of traditional control frameworks. This synthesis validates the necessity of a new hybrid approach capable of supporting dynamic, multi-leg contact planning for robust quadruped locomotion.

3 Methodology

3.1 System Overview

This work presents a control framework for quadruped robots capable of generating dynamic, acyclic gaits in challenging environments. The framework employs a hierarchical architecture that processes robot state and terrain data to produce footstep commands for the robot controller.

The first stage of the framework comprises a footstep evaluation network, similar to ContactNet from [1], which estimates candidate footsteps $\mathbf{f_c}$. The second stage, a gait generation network (GaitNet), receives these candidates $\mathbf{f_c}$, ranks them, and outputs the optimal footstep action $\mathbf{f_a}$ to the robot controller. The robot controller is the MIT Mini-Cheetah Controller [2], implemented in python by [43].

This hierarchical design enables strict control over the range of possible actions the robot can execute. Between the two stages, candidate actions are filtered to ensure that only valid, prescribed movements are permitted.

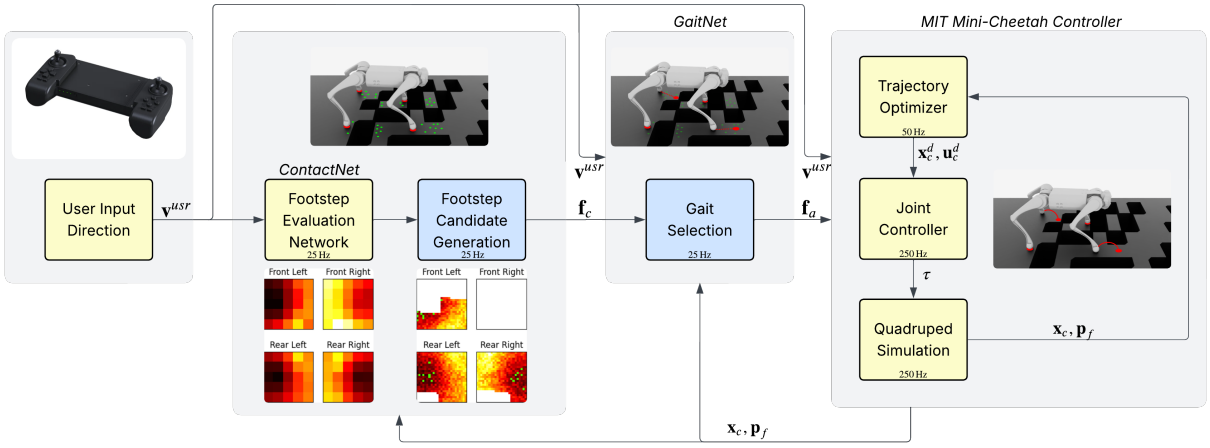


Figure 3.1: Block diagram of the proposed framework. Novel aspects shown in blue. The user defines an input direction $\mathbf{v^{usr}}$, which the *Footstep Evaluation Network* (section 3.3), based on *ContactNet* [1], uses along with the robot state $\mathbf{x_c}$ and the current foot positions $\mathbf{p_f}$ to generate a footstep cost map. The footstep cost map is then processed (subsection 3.3.3) to produce a set of candidate footstep actions $\mathbf{f_c}$. These candidates are passed to the *GaitNet* (section 3.4), which selects one[1]action $\mathbf{f_a}$ to send to the robot controller. The *MIT Mini-Cheetah Controller* [2] receives $\mathbf{f_a}$ and executes the corresponding actions.

[1]As is explained in later sections, GaitNet only outputs one footstep action at a time. The two separate foot target locations shown in the GaitNet visual would be selected sequentially with a 1/25s delay. This visual is meant to highlight the near simultaneous nature of the footstep selection process.

3.2 Simulation Environment

The simulation environment used for this project is NVIDIA Isaac Lab [44]. This framework was selected for several reasons. It is a modern platform with a Python interface designed specifically for reinforcement learning applications and GPU parallelism. The use of GPU parallelism enables significantly faster simulation and data collection, albeit at the cost of increased programming complexity and reduced compatibility with older hardware. Furthermore, the extensive collection of example and community projects provides valuable references for implementing the simulation features required in this work.

Although both simulation and learning processes are executed on the GPU, the robot controllers operate on the CPU. Figure 3.2 illustrates the overall processing flow. The simulation environment runs entirely on the GPU, where multiple robots are simulated in parallel. Meanwhile, the robot model predictive controllers (MPCs) execute on the CPU, with each controller running concurrently. The CPU and GPU communicate at every simulation step to exchange robot states and corresponding control actions.

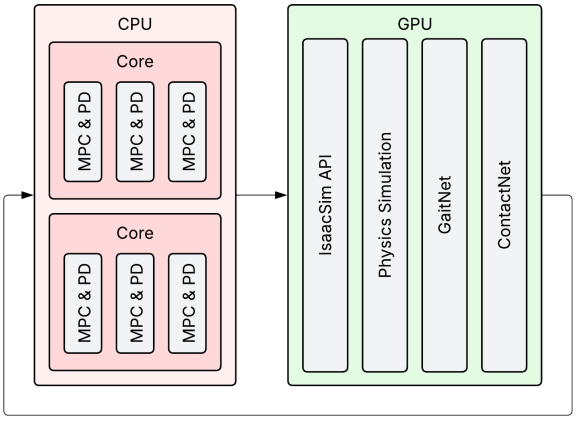


Figure 3.2: Block diagram showing the programming tasks computed on the CPU versus the GPU. The full simulation is run in parallel using NVIDIA Isaac Lab on the GPU, while the robot MPCs are run in parallel on the CPU. Multiple Cores in the CPU and multiple "MPC & PD" in each core show the hierarchy of how the parallel computing workload is assigned.

NVIDIA Isaac Lab uses a declarative system of Python dataclasses to define the simulation environment. For this work, a custom environment is used (Figure 3.3), including:

- Unitree Go 1—Configured to use force control for each joint.
- Terrain raycasts—Measure the height of the terrain at each possible footstep location. This emulates the lidar processing steps of a full vision pipeline.
- Custom terrain—A grid of 12×12 sub-terrains (4×4 m each) with increasing difficulty (Figure 3.4). Each sub-terrain consists of a 12 cm square grid pattern with missing sections. The void density varies from approximately 0% to 40%. The quantity of terrain is limited by GPU memory.

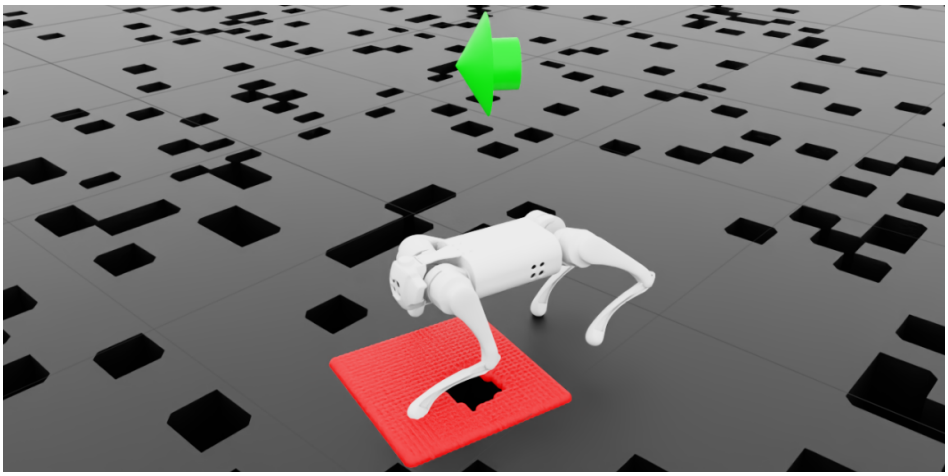


Figure 3.3: Image of a Unitree Go 1 navigating the terrain. Red region shows area converted into heightmap for front left leg. Black regions show voids in the terrain. Green arrow shows desired velocity vector.

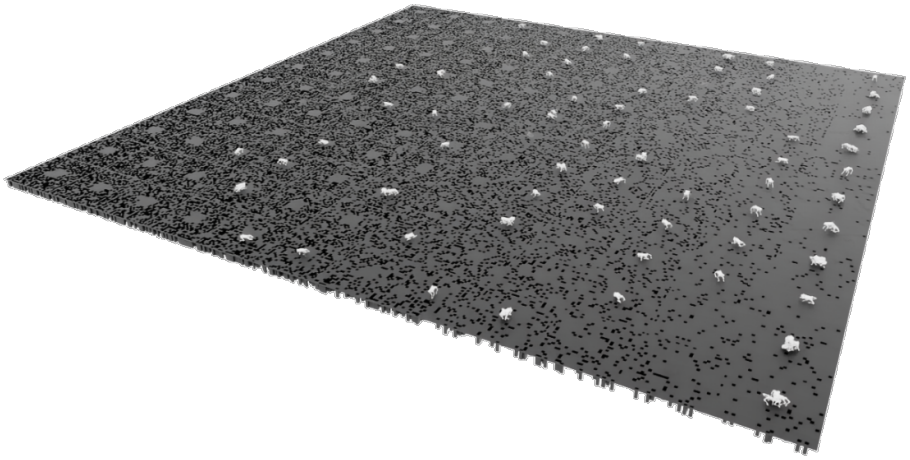


Figure 3.4: Image showing the full terrain used in simulation. Full terrain is a grid of 12×12 sub-terrains with increasing difficulty. Sub-terrains are 4×4 m grids with missing sections.

3.3 Footstep Evaluation Network

The purpose of the footstep evaluation network is to generate footstep candidates for the GaitNet model. These footstep candidates allow the remainder of the system to work in a discrete space, only choosing from a set of these footstep candidates instead of optimizing over the whole space. Because of this, the precise output of this network is not critical; rather, it is essential that the network provides high-quality candidates when sampled. It achieves this by estimating the cost associated with potential footsteps given the current robot state. The network's architecture and training process are largely based on ContactNet [1], with several key modifications described in the following section.

The primary reason for the differences between this Footstep Evaluation Network and ContactNet [1] is because of their slightly different use cases. ContactNet was designed to provide a single action at each evaluation, whereas the Footstep Evaluation Network needs to provide multiple feasible candidate actions to be narrowed down later. Despite this different use case, ContactNet still provides much of the guidance for this implementation.

3.3.1 Architecture

The footstep evaluation network is responsible for generating a set of footstep candidates $\mathbf{f_c}$ based on footstep evaluation network input $\mathbf{x}_c$:

$$\mathbf{x_c} = \begin{bmatrix} \mathbf{p}_{b,xy} \\ \mathbf{r}_{w,z} \\ \mathbf{v}_b \\ \omega_b \\ \mathbf{u} \end{bmatrix} \tag{3.1}$$

Here, $\mathbf{p}_{b,xy}$ represents the x and y positions of all end effectors in the base frame, stacked into a single vector, and $\mathbf{r}_{w,z}$ denotes the height of the robot's center of mass in the world frame, $\mathbf{v}_b$ is linear velocity of the robot in the root frame, ω_b is the angular velocity of the robot in the root frame, and $\mathbf{u}$ is the control input as a vector of $[v_x, v_y, \omega_z]$. The inclusion of ω_b distinguishes this formulation from that in [1]. This modification was intended to improve the model's performance in situations with high rotational velocities.

A set of footstep candidates is referred to as a footstep cost map (Figure 3.5), which stores the expected cost of different actions the robot could take with its feet. These maps are represented as four 5×5 grids, one for each leg. This choice differs from the implementation in [1], a larger grid size was selected to provide more information for each foot, which is important later when sampling footstep candidates. Ground truth footstep cost maps are generated as described in subsection 3.3.2, then the network is trained to replicate the ground truths. After this the network is able to quickly generate a footstep cost map based on $\mathbf{x}_c$.

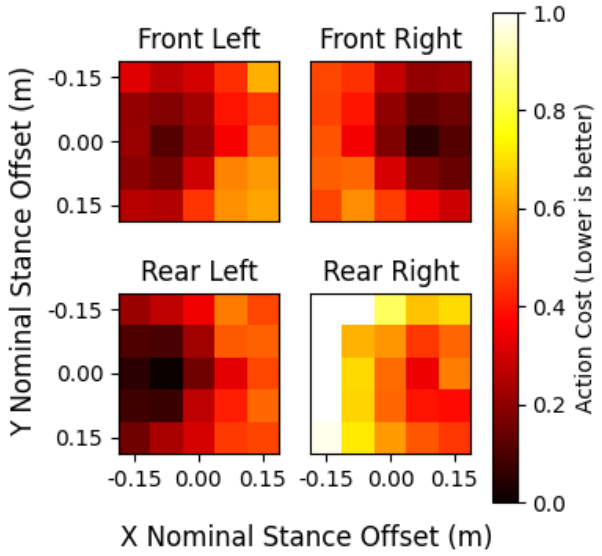


Figure 3.5: Example footstep cost maps for all four legs. Darker colors indicate lower cost (more favorable) footstep positions.

The architecture of the footstep evaluation network is shown in Figure 3.6. It consists of a feedforward neural network that initially maps the input through two fully connected layers of 64 nodes each, both with ReLU activations. The resulting 64-dimensional feature vector is reshaped into an 8×8 spatial representation and processed by a convolutional layer with two output channels, a 3×3 kernel, stride 1, and padding 1, followed by a ReLU nonlinearity. The output is flattened and passed through a final fully connected layer before being reshaped into a 5×5 grid. This specific architecture differs from [1], where a larger network without the convolutional layer was used. These modifications were found to provide better results for the 5×5 output grid.

To produce the four footstep candidate maps, this network is replicated four times—once per leg—with weights not shared between legs. This design allows the network to learn leg-specific behaviors, accommodating potential asymmetries in the robot's dynamics.

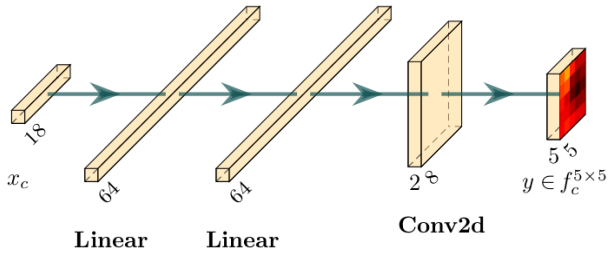


Figure 3.6: Footstep evaluation neural network architecture.

3.3.2 Training

The footstep evaluation network is trained to predict heuristic footstep cost maps, as described in [1]. Training data is generated using the simulation environment outlined in section 3.2, but with planar terrain. As in [1], planar terrain is sufficient for data collection because the cost maps are masked based on terrain after inference (see subsection 3.3.3).

During training, 100 robots are simulated in parallel, divided into four groups of 25. Each group tests a grid of footstep positions for one foot at a time; the front-left group is illustrated in Figure 3.7. An *iteration* is considered complete once all robots have either successfully completed or failed their assigned motions. Importantly, all robots begin each iteration from the same initial state. Figure 3.8 shows an example iteration from start to finish. Note that the rear right leg's position differs between Figure 3.8a and Figure 3.8d because that was the solution chosen to be explored further.

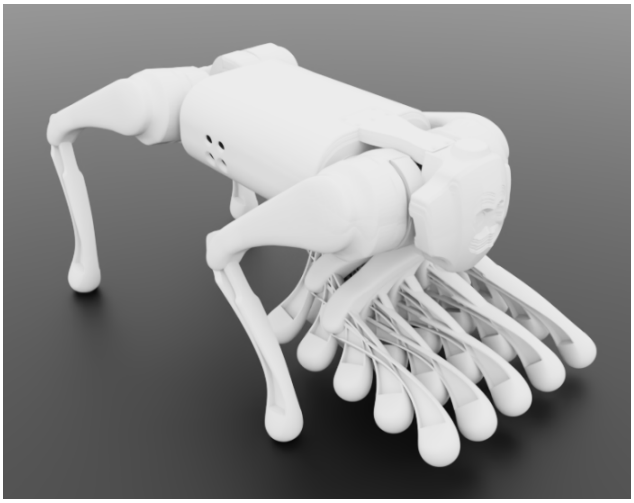
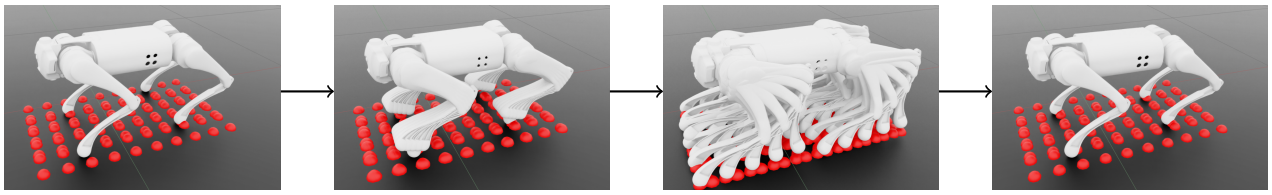


Figure 3.7: Snapshot showing 25 robots testing different footstep positions in parallel. For actual data generation, 100 robots are run in parallel, testing 25 footstep positions for each foot at a time.



(a) Previous best state (b) Beginning of swing (c) End of swing (d) Continue search from a best state

Figure 3.8: Sample iteration of the footstep evaluation network data generation process. Note that all 100 robots are present in all figures, they just occupy the same space. Additionally, each robot only ever moves one leg at a time. (a) shows the previous best state, which is used as the starting point for the iteration. (b) shows the beginning of the swing phase, where the foot is lifted off the ground. (c) shows the end of the swing phase, where the foot is placed in a new position. (d) shows the next iteration, which continues the search from one of the 10 best states found in (c).

Data collection proceeds by chaining multiple iterations together, with control inputs periodically re-sampled from $\mathbf{u} \in (-0.2, 0.2) \times (-0.2, 0.2) \times (-0.4, 0.4)$, where $\times$ denotes the Cartesian product. After each iteration, the top 10 actions are inserted into a tree structure as edges, with the resulting state as the leaf node. The tree is explored by randomly selecting leaves for expansion, and this process continues until a predefined maximum number of iterations is reached. To ensure data quality, iterations in which more than 50% of the robots fall are discarded, along with their two parent nodes in the tree. This approach promotes the collection of diverse, yet successful, training data. Collecting data as described in this section captures the same information as the data collection plan described in [1], with the addition of a more broad range of commanded velocities.

Figure 3.9 illustrates the distribution of foot positions in the training dataset. Darker regions correspond to discrete points on the 5×5 footstep grids; a foot occupies a given position if it was moved there in that iteration. The uniform coverage indicates that the training dataset effectively captures a wide range of states.

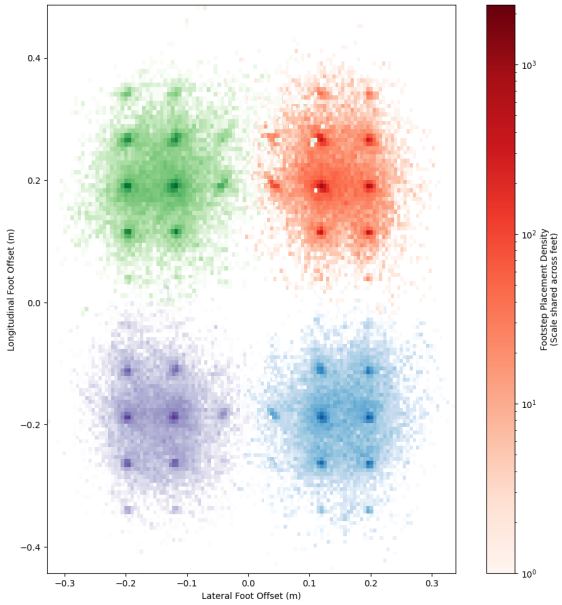


Figure 3.9: Foot placement heatmaps showing distribution of foot positions in the GaitNet training data. Note that the histograms are overlaid in some places, obscuring underlying data.

The purpose of this data collection is to assign a cost to each potential footstep, forming the candidate set $\mathbf{f_c}$. Costs are computed heuristically based on simulation outcomes, balancing stability and efficiency. While similar to the approach in [1], the heuristic employed here differs in several aspects to better suit the output for footstep candidate sampling. Figure 3.11 shows the individual factors used to construct the footstep

candidate maps, and Figure 3.10 displays the resulting combined cost map. These factors are described below, referencing the subfigures in Figure 3.11

The lin_vel_z_l2 factor (Figure 3.11a) penalizes high vertical velocity of the trunk. The ang_vel_xy_l2 factor (Figure 3.11b) penalizes high angular velocity of the trunk in the horizontal axes. The joint_torques_l2 factor (Figure 3.11c) penalizes high joint torques, and the joint_acc_l2 factor (Figure 3.11d) penalizes high joint accelerations. The control_error factor (Figure 3.11e) penalizes errors between the control input and actual robot motion. The inscribed_circle_radius factor (Figure 3.11f) measures the distance from the center of mass to the nearest edge of the support polygon, encouraging the robot to maintain its center of mass in a stable position. The foot_hip_distance factor (Figure 3.11g) measures the distance between the hip and foot in the XY plane, encouraging the robot to keep its feet moving in coordination with the body.

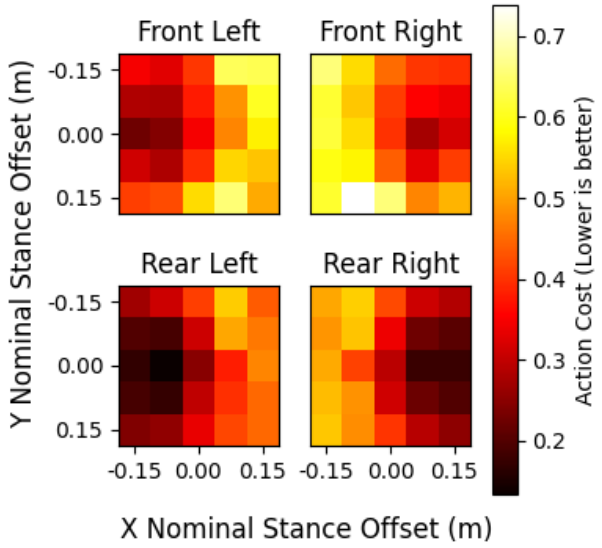
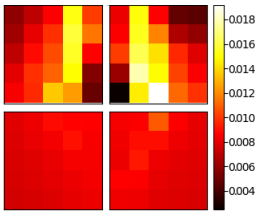
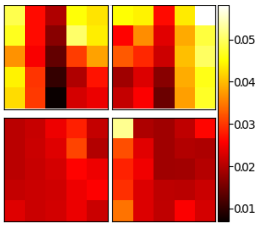


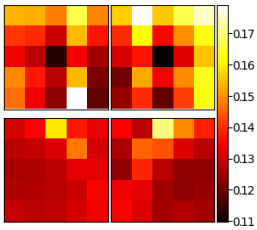
Figure 3.10: Combined cost map from factors in Figure 3.11 (not normalized).



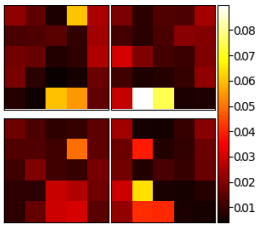
(a) Z linear velocity L2 normalized; balanced, with 0.017 spread.



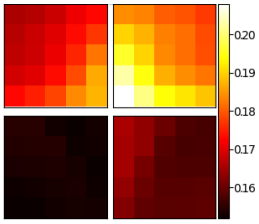
(b) XY angular velocity L2 normalized; balanced, with 0.051 spread.



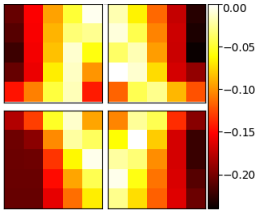
(c) Joint torques L2 normalized; balanced, with 0.069 spread.



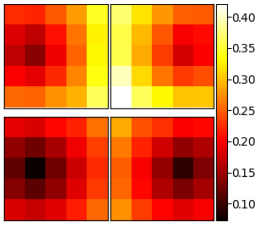
(d) Joint accelerations L2 normalized; balanced, with 0.086 spread.



(e) Control error; balanced, with 0.056 spread.



(f) Inscribed circle radius; with 0.240 spread.



(g) Foot-hip distance; with 0.348 spread.

Figure 3.11: Factors influencing footstep candidate maps. *balanced* indicates that the values for each leg were balanced to have a lower spread, mitigating factors that consistently prefer one leg over another. The last number in parentheses indicates the total range of the data, the most important factor for the combined cost map.

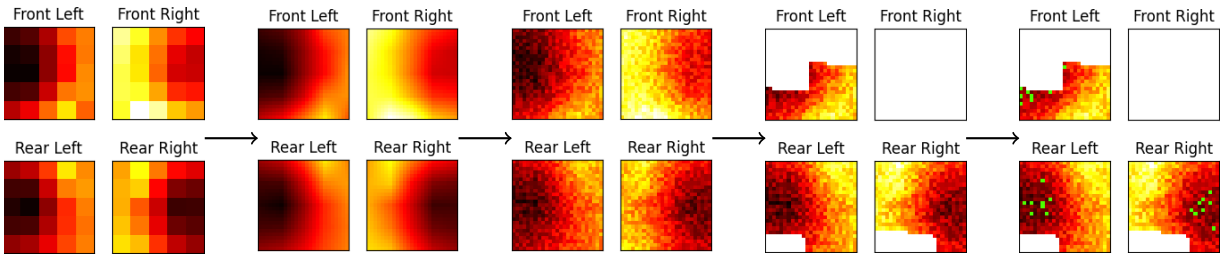
As in [1], the cost maps are normalized to enhance training performance. The approach here differs in that the maps are normalized directly to the range $[0, 1]$, rather than preserving only the relative ordering of costs as in [1]. This direct normalization is crucial for providing the upstream GaitNet model with maximal information.

3.3.3 Post-Processing

The primary purpose of the footstep evaluation network is to provide high-quality footstep candidates to the GaitNet model. We found the 5×5 grid overly restrictive masking invalid terrain. Additionally, the 5×5 grid could only provide a very limited set of footstep candidates. To address both of these issues, we propose a post-processing pipeline to augment the output of the footstep evaluation network to provide more diverse footstep candidates for this application.

The raw cost map output from the footstep evaluation network (Figure 3.12a) is first upsampled (Figure 3.12b) to increase the resolution of possible footstep positions and to match the resolution of the terrain data. Next, noise is added (Figure 3.12c) to encourage exploration of a more diverse set of footstep positions; without this noise, all candidates would cluster in close proximity.

The cost map is then filtered based on the robot state and terrain data (Figure 3.12d) to mask out invalid actions. This includes positions that are too close to terrain edges and movements that would attempt to reposition a leg already in the swing phase (as illustrated by the front right leg in the figure). Additional masking occurs based on the total number of legs in the swing phase; when two legs are already in swing, all remaining options are masked out to prevent unstable motions. Finally, the top eight candidates from each leg are selected (Figure 3.12e) for processing by GaitNet. If fewer than eight valid candidates exist for a given leg, no-action candidates are added to maintain consistent tensor dimensions.



(a) Raw (b) Upsample (c) Noise (d) C-space (e) TopK (green)

Figure 3.12: Cost map processing pipeline. Shows how the raw cost map is processed to produce the final footstep candidates.

3.4 GaitNet

GaitNet is designed to select the optimal footstep action from a discrete set of candidates. By restricting selection to a predefined set of continuous actions, the algorithm enforces constraints on the robot's motion, strictly preventing invalid foot placements and limiting the range of possible gaits. The inclusion of a no-action candidate, combined with continuous re-evaluation, distinguishes this approach: it enables the greedy generation of acyclic gaits in which multiple feet can be in the swing phase simultaneously.

3.4.1 Architecture

GaitNet is responsible for ranking footstep candidates based on the one-hot encoded leg index $\mathbf{f_c}'$ and the GaitNet input $\mathbf{x_g}$:

$$\mathbf{x_g} = \begin{bmatrix} \mathbf{p}_{b,xy} \\ \mathbf{r}_{w,z} \\ \mathbf{v}_b \\ \omega_b \\ \mathbf{u} \\ \mathbf{g} \\ \mathbf{c} \end{bmatrix} \tag{3.2}$$

Descriptions of the first five terms can be found in subsection 3.3.1. The additional terms, compared to the footstep evaluation network input (3.1), are $\mathbf{g}$ and $\mathbf{c}$, being the gravity vector and contact state respectively. Using $\mathbf{x_g}$ and $\mathbf{f_c}'$, GaitNet outputs a logit associated with each footstep and the desired swing duration if that step is selected.

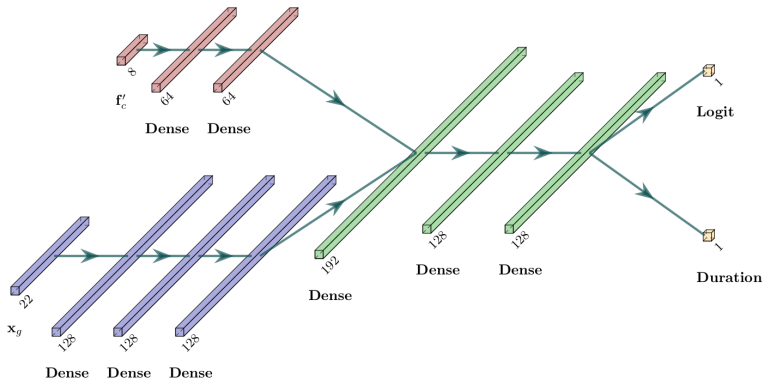


Figure 3.13: GaitNet architecture. Sections of the diagram include the footstep candidate encoder in red, robot state encoder in blue, and shared trunk in green. The final output is a logit encoding the value of this option and the desired swing duration if this action is taken.

The model is designed to jointly evaluate robot state and footstep candidates with a middle fusion architecture [45]. The architecture (shown in Figure 3.13) consists of two encoders, a shared trunk, and two task-specific output heads:

- Robot state encoder: The robot state vector is processed by a three-layer feedforward network with intermediate dimensionality of 128 and ReLU activations. This produces a fixed-dimensional latent representation of the current robot state.
- Footstep encoder: Each footstep candidate is encoded by a two-layer feedforward network with hidden size 64 and ReLU activations. No-action candidates are represented through a fixed embedding.
- Shared trunk: The concatenated robot state and footstep embeddings are processed by a three-layer feedforward trunk, reducing dimensionality while applying layer normalization and ReLU nonlinearity.
- Output heads: Two parallel prediction heads are applied to the shared trunk:
  - Logit head: A single-layer network outputs the predicted reward value as a logit.
  - Duration head: A single-layer network with sigmoid activation outputs a normalized swing duration, which is then scaled to the range $(0.1, 0.3)$ s.

This design allows the network to sequentially evaluate multiple footstep candidates, assigning each an expected value and a feasible swing duration, while explicitly supporting a no-action option via a dedicated embedding.

3.4.2 Training

Due to the inclusion of the $\mathbf{c}$ term in the input, GaitNet cannot be effectively trained using supervised learning because of the high dimensionality of the problem. Instead, it is trained using PPO [46]. In this setup, the actor network is GaitNet (Figure 3.13), while the critic follows a similar architecture but omits the duration head. All logits from the critic are passed through a final MLP to produce a single value estimate. This MLP consists of two hidden layers of size 64, with ReLU activations applied to all layers except the output.

A custom actor-critic implementation was necessary to handle GaitNet's dual outputs. The logits define a categorical distribution over footstep candidates, while the duration output is treated as a separate normal distribution with a fixed standard deviation of 0.01 s for each action. To prevent multiple no-action candidates from skewing the selection, all but one no-action candidate are removed (set to $-\infty$).

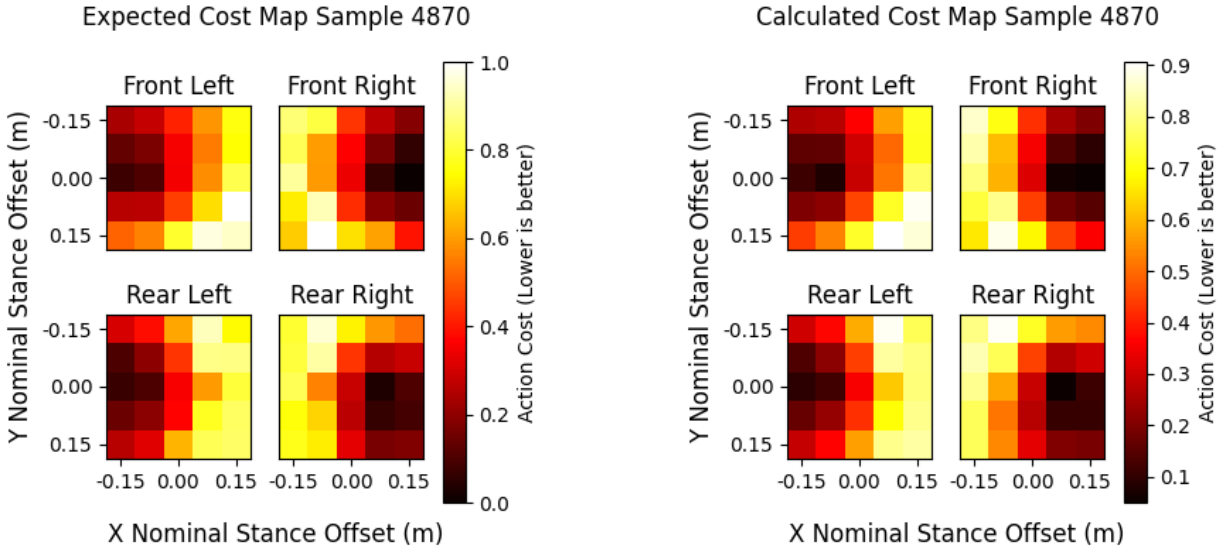
The total log probability of an action is computed as the sum of the categorical log probability of the selected footstep candidate and the log probability of the duration under the normal distribution. During action sampling, a footstep candidate is first sampled from the categorical distribution, followed by sampling the duration from its associated normal distribution. The footstep candidate and duration are then combined to form a complete footstep action.

Following the reward structure, termination conditions, commands, and hyperparameters outlined in Appendix A, GaitNet is trained for 700 environment steps using 100 agents in parallel. Each environment step simulates 20 s. The 700 step end point was chosen because improvement drastically slowed by then.

4 Results and Discussion

4.1 Footstep Evaluation Network

The results of the footstep evaluation network are highly promising, with the model able to predict footstep candidate maps with strong accuracy. As a brief recap, the cost maps visualize a set of footstep candidates $\mathbf{f}_c$, which show the heuristic cost of moving legs to different positions from an initial state. Figure 4.1 shows the model output alongside the ground truth for a typical data sample. There are some very minor differences between the ground truth and model generated cost maps, but the model is capable of correctly identifying the general low-cost regions (darker). Notice how the ground truth generally has the lowest cost regions at $y = 0$ and $x = \pm 0.07$ depending on the side of the leg, and the model generated cost map identifies the lowest cost areas as being in generally the same places.

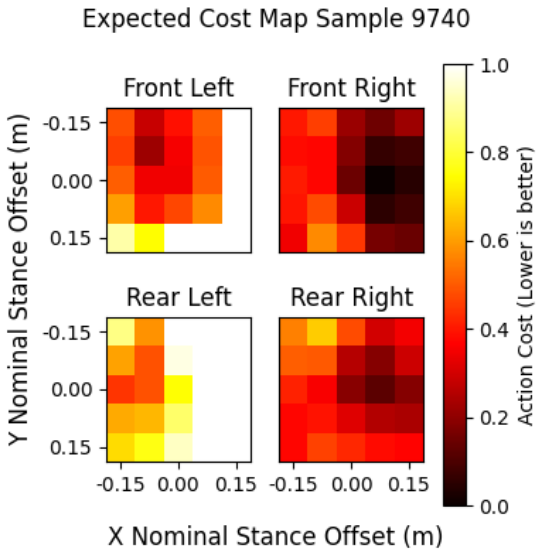


(a) Heuristically calculated cost map.

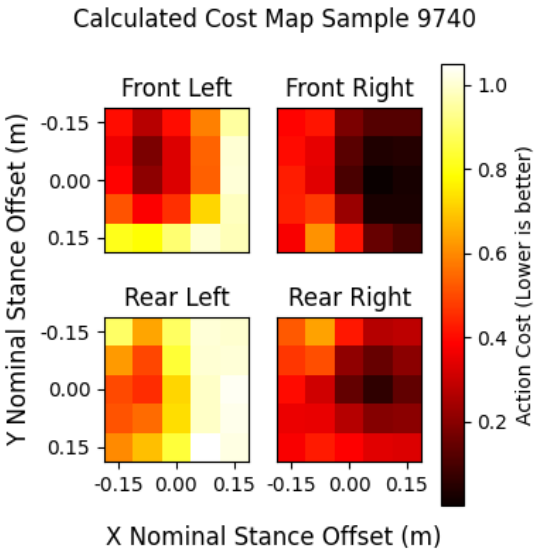
(b) Cost map generated by the footstep evaluation network.

Figure 4.1: Comparison of a typical heuristically calculated cost map (a ground truth used in network training) to the reconstructed cost map generated by the footstep evaluation network.

Figure 4.2 illustrates a particularly challenging sample, in which the model still successfully identifies the most suitable positions for each leg. Notably, the rear left leg must be positioned far from its nominal location to maintain stability in this scenario, and the model correctly predicts this adjustment.



(a) Heuristically calculated cost map.



(b) Cost map generated by the footstep evaluation network.

Figure 4.2: Comparison of a particularly challenging heuristically calculated cost map (a ground truth used in network training) to the reconstructed cost map generated by the footstep evaluation network.

4.1.1 Discussion

The footstep evaluation network effectively samples the continuous action space, providing diverse and feasible footstep candidates to GaitNet. In the context of this work, the precise accuracy of the model is not critical. Its primary role is to sample the continuous space of foot positions to generate candidate actions for the GaitNet policy, so the most important aspect is correctly identifying the general lowest cost regions to move the foot to.

While not perfectly accurate in all scenarios, the network's ability to identify suitable foothold regions—even in challenging configurations as seen in Figure 4.2—validates its role as a candidate generator rather than a precise predictor. When comparing the generated figures to the ground truth, the most important observation is that the darkest region of each sub-plot is in roughly the correct area. The results indicate that the footstep evaluation network is suitable for generating footstep candidates.

4.2 GaitNet

The primary objective of GaitNet is to generate fully dynamic and acyclic gaits for quadruped robots. An example swing sequence is shown in Figure 4.3, illustrating a non-repetitive, dynamic gait in which the robot performs motions with up to two feet off the ground simultaneously. Swing durations are non-uniform, with some legs remaining in the swing phase longer than others. A video of GaitNet navigating challenging terrain can be seen here.

Examining Figure 4.3, a short step is observed for the front-left leg (green) at 3 s, while the rear-right leg (red) executes a longer step at 5 s. Additionally, between 0–7.5 s, the robot is commanded to follow a slow input, moving one foot at a time. After 7.5 s, the input speed increases, prompting a more dynamic gait. During this faster phase, the robot occasionally executes two steps simultaneously and does not follow a strict alternating pattern.

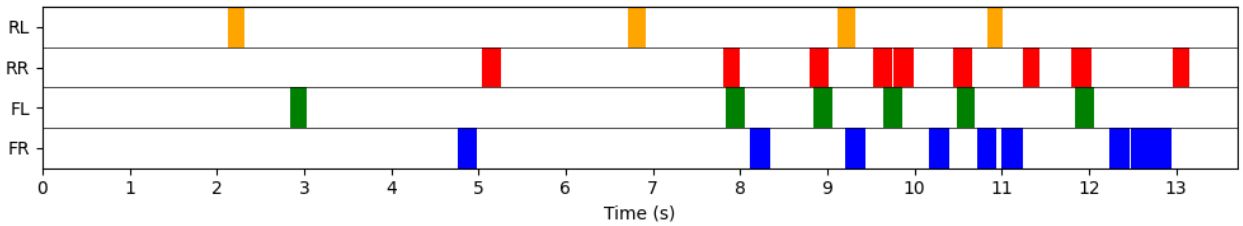


Figure 4.3: Example swing schedule for a single gait cycle. Each row represents a leg, with color indicating the leg is in swing phase.

To ensure the consistency of the training process, GaitNet was trained three times. Due to computational limitations, the 700-step training curriculum needed to be split into multiple runs. The runs were typically 350 steps long. This has the side effect of resetting the distribution of terrain levels, which can cause discontinuities in certain training metrics. Resetting the simulations did not appear to have an adverse affect on the learning process.

The mean episode reward during training is shown in Figure 4.4. Each training instance shows a similar trend, with the reward increasing as the model learns to navigate the terrain.

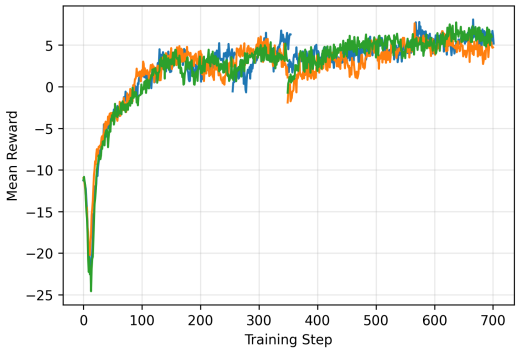


Figure 4.4: Mean episode reward during GaitNet training. Each color represents a different training instance.

Another important metric is the mean steps per second, shown in Figure 4.5. Initially, with limited training, the model moves its legs very frequently, leading to unstable motion. In initial experiments, it was found that this issue does not resolve well without guidance from the reward function. section A.1 provides more detail on this issue. The mean steps per second starts high and decreases as the model learns to take more stable steps.

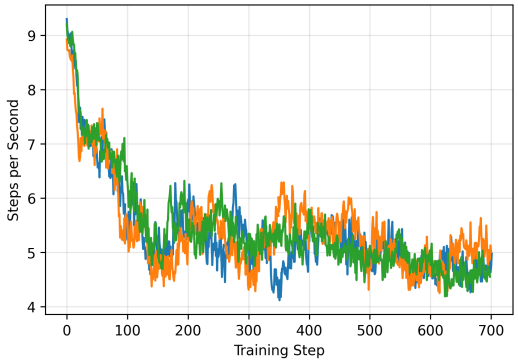


Figure 4.5: Mean steps per second during GaitNet training. Each color represents a different training instance.

During training, the GaitNet model learns to adjust swing durations based on its input, as illustrated in Figure 4.6 and Figure 4.7. The mean swing duration settles to 0.24 s with a standard deviation of 0.034 s. Figure 4.8 shows exactly what has been learned more clearly. For this histogram all swing durations from a GaitNet test described in section 4.3 were recorded. We can see a large peak between 0.225 s and 0.25 s, with a significant amount of swings down to 0.125 s.

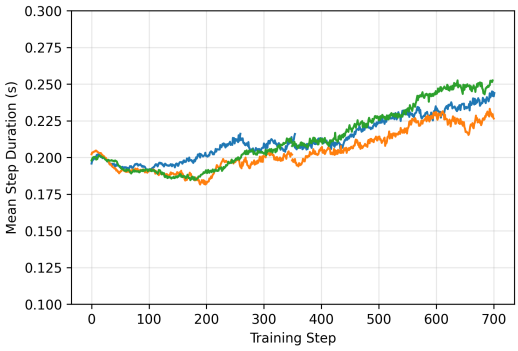


Figure 4.6: Mean swing duration across all environments during GaitNet training. Each color represents a different training instance.

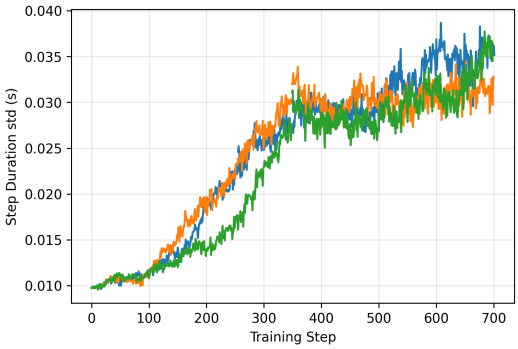


Figure 4.7: Swing duration standard deviation across all environments during GaitNet training. Each color represents a different training instance.

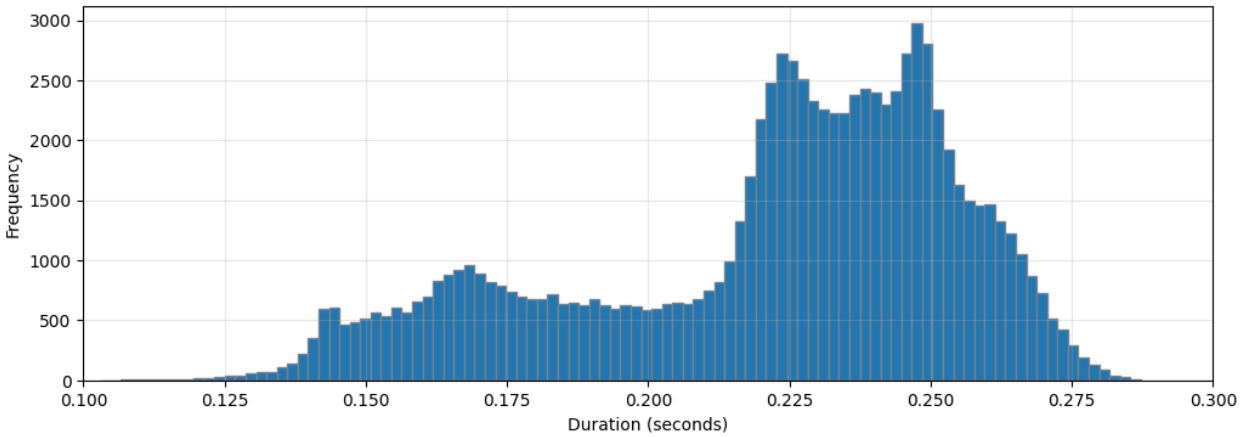


Figure 4.8: Histogram of swing durations selected by GaitNet.

4.2.1 Discussion

The results presented demonstrate that GaitNet successfully generates dynamic, acyclic gaits for quadruped locomotion in challenging environments. The system can synthesize a wide variety of actions, and the results from the swing schedule analysis indicate that the system is capable of dynamically varying step timing. The system's response to changing control inputs—taking a cautious approach at low speeds and executing simultaneous steps at high speeds—highlights the system's flexibility in generating acyclic gaits without pre-defined patterns.

Regarding the training metrics, there was no specific target for an ideal steps per second, but the change during training shows that the model is correctly learning when and when not to step. Similarly, there was no specific target value for the swing duration metrics. They are presented to verify that GaitNet is capable of learning to utilize its swing duration output. The convergence of the mean swing duration and standard deviation indicates that the model uses a wide range of its allowable swing durations. The histogram further clarifies this, showing that while the system favors durations between 0.225 s and 0.25 s for normal motion, it utilizes durations down to 0.125 s for situational motions, such as moving across tricky terrain or recovering from falls. These figures confirm that the model learns to effectively utilize dynamic swing durations to adapt to varying conditions.

4.3 Baseline Comparison

For the purpose of evaluating GaitNet's performance, a baseline method is established using a single leg motion planner. This planner operates as is described in [1], where in place of ContactNet, we directly use our footstep evaluation network described in section 3.3, excluding the noise post-processing step (Figure 3.12c), and only picking one candidate in the selection step (Figure 3.12e). The lowest cost candidate is then used as the footstep target for that leg, with a 200ms swing duration.

In order to benchmark the two methods, a custom test environment is created (Figure 4.9). This environment has the robots navigate straight forward across narrow strips of terrain with characteristics matching the sub-terrains used in training (section 3.2). The space of terrain difficulties and commanded forward velocities is systematically explored with a grid search, evaluating the success rate of each method over 50 trials of 20 s each. The terrain difficulty is a value between 0% and 100% represents the percent of terrain which can be stepped on. A checkerboard pattern (with alternating holes and valid terrain) would have a terrain difficulty of 50%, though in these tests the terrain is randomly distributed, so a checkerboard pattern is unlikely. The chosen range of terrain difficulties $\{0.0\%, 0.05\%, 0.1\%, \ldots, 0.4\%\}$ and command velocities $\{0.05\,\text{m/s}, 0.1\,\text{m/s}, 0.15\,\text{m/s}, 0.2\,\text{m/s}\}$ were chosen to represent a wide variety of situations while not straying too far from what any method is capable of.

An episode is considered successful if the robot is able to complete the full 20 s without terminating according to the conditions described in section A.2. Briefly, these conditions amount to terminating if the

robot falls over or has its foot slip into the holes in the ground.

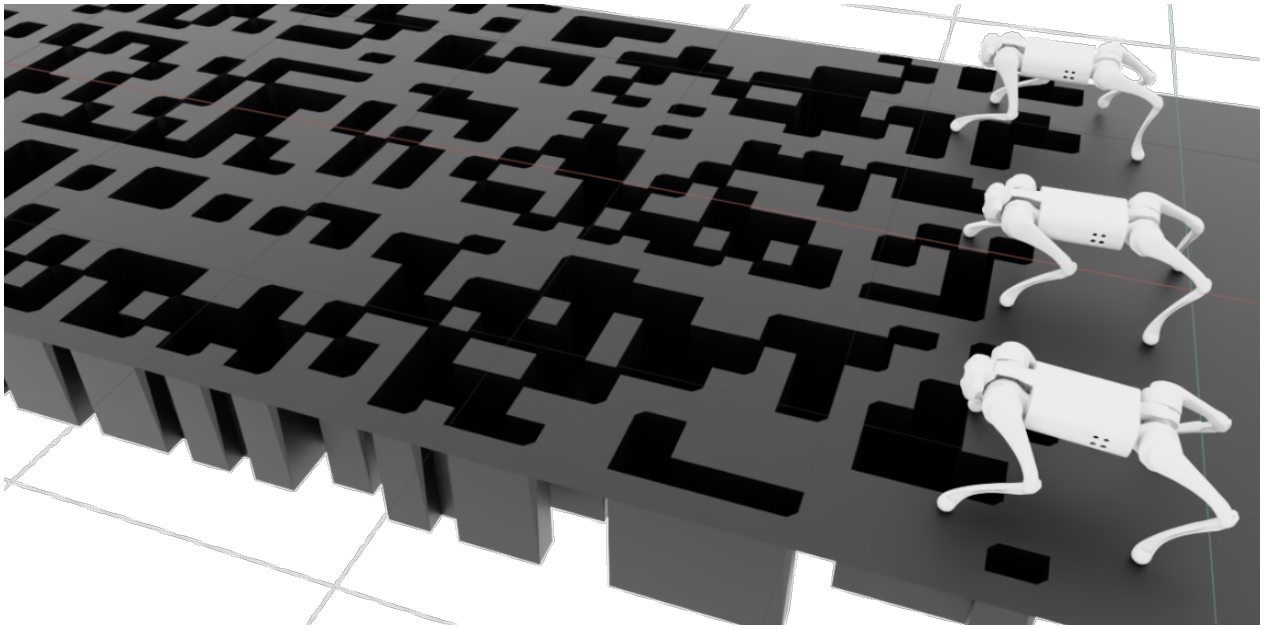


Figure 4.9: Three robots simultaneously navigating a test environment (40% terrain difficulty) used for baseline comparison. The terrain consists of narrow strips of a grid pattern with missing sections. Density of missing sections (difficulty) and commanded forward velocity are varied to evaluate performance.

Figure 4.10 and Figure 4.11 illustrate the performance of GaitNet and the baseline method, respectively, across a range of terrain difficulties and commanded velocities.

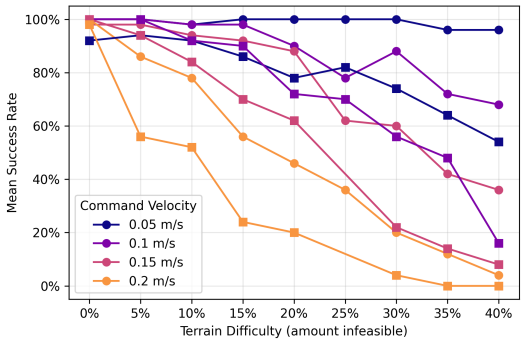


Figure 4.10: GaitNet Evaluation. Overall survival rate of 69.4%. Mean success rate measured as the percentage of 50 trials which completed 20 s without terminating, under the termination conditions described in section A.2. Data point shapes denote different training instances.

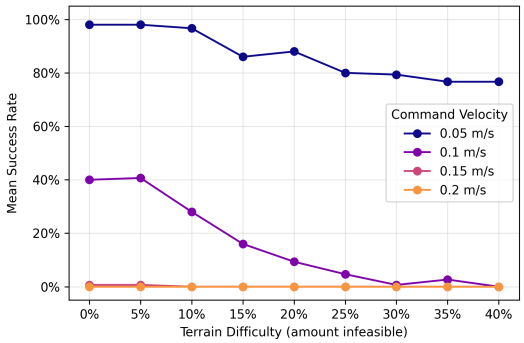


Figure 4.11: Single Leg Motion Planner Evaluation. Overall survival rate of 25.6%. Mean success rate measured as the percentage of 50 trials which completed 20 s without terminating, under the termination conditions described in section A.2.

4.3.1 Discussion

GaitNet's performance relative to the single leg motion planner baseline reveals the clear advantage of dynamic gait generation. The 69.4% survival rate achieved by GaitNet, compared to 25.6% for the baseline, demonstrates that the system's ability to coordinate multi-leg motions and adapt timing significantly improves robustness. This improvement is particularly pronounced at higher commanded velocities and terrain difficulties, where the baseline's single leg motion strategy fails.

A closer examination of the graphs reveals that GaitNet maintains strong performance when the commanded velocity is below 0.15 m/s or terrain difficulty is under 5%. In contrast, ContactNet's performance begins to degrade for commanded velocities above 0.05 m/s and deteriorates further as terrain difficulty increases. The performance gap narrows only in the easiest conditions (at 0.05 m/s velocity), suggesting that dynamic gait planning becomes increasingly valuable as task complexity increases. GaitNet's superior performance in these scenarios underscores the benefits of its dynamic, acyclic gait generation capabilities.

4.4 Swing Duration Ablation Study

In this section we present an ablation study to assess the impact of dynamic swing duration on GaitNet's performance. We compare two models: the standard GaitNet formulation as described in section 3.4, and a Duration-Ablated-GaitNet trained with a fixed swing duration. In the GaitNet formulation described previously, the model explicitly outputs a swing duration to use for every action. The corresponding swing duration is then used for the chosen action. For this ablation, we ignore the swing duration output of GaitNet and directly use a fixed value, 0.24 s. This value was chosen because it is roughly the mean step duration used by GaitNet. This will provide information about how important the variable swing duration is for this use case.

Now, we train the Duration-Ablated-GaitNet model, setting the swing duration to a constant 0.24 s. The training process is identical to that of the standard GaitNet model, ensuring a fair comparison. The performance of both models is then evaluated using the same metrics outlined in section 4.3.

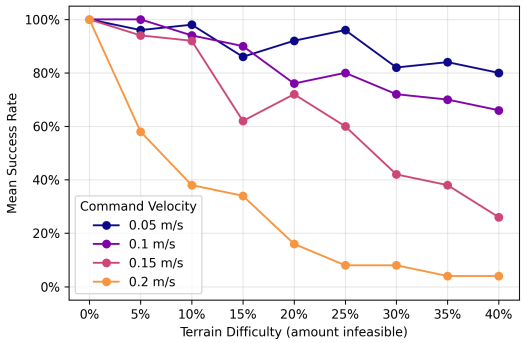


Figure 4.12: Evaluation of Duration-Ablated-GaitNet across various terrain difficulties and commanded velocities. Overall survival rate of 67.2%. Mean success rate measured as the percentage of 50 episodes which completed 20 s without terminating, under the termination conditions described in section A.2.

Figure 4.12 presents the performance of the Duration-Ablated-GaitNet. The results indicates negligible performance changes over standard GaitNet formulation. GaitNet was able to achieve an overall survival rate of 69.4% (Figure 4.10), while the Cost-Ablated-GaitNet achieved a survival rate of 67.2% (Figure 4.12). The mean episode reward during training, as shown in Figure 4.13, shows a similar trend.

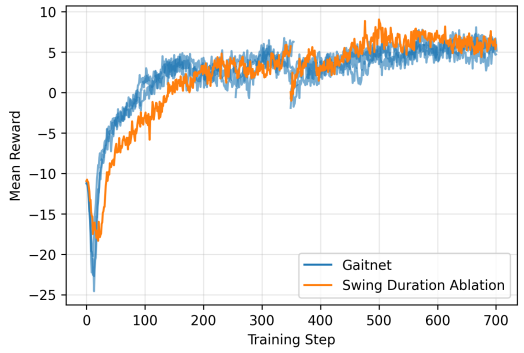


Figure 4.13: Mean episode reward during training for GaitNet and Duration-Ablated-GaitNet.

4.4.1 Discussion

The swing duration ablation study yields a surprising result: removing dynamic swing duration control has minimal impact on performance. The steady state behavior of the reward curves for both models indicates that the Duration-Ablated-GaitNet is able to learn a policy that performs comparably to the standard GaitNet, despite the lack of dynamic swing duration. This finding suggests two possibilities. First, the current training environment may not sufficiently challenge the system to exploit variable swing timing—the static terrain and relatively low speed requirements may not create scenarios where dynamic timing provides substantial benefits. Second, the reward function may not adequately incentivize optimal swing duration selection, causing the network to default to near-constant timing regardless of capability.

The observation that trained GaitNet naturally converges to a mean swing duration of 0.24 s with only modest variation (standard deviation of 0.034 s) supports this interpretation. These findings suggest that dynamic swing duration may not be a critical factor for GaitNet's performance in the tested environments. However, it is important to note that this conclusion may not hold in more complex or dynamic environments, where the ability to adjust swing duration could provide a significant advantage. Future work in more dynamic environments or with revised reward structures may reveal greater utility for this feature.

4.5 Action Cost Ablation Study

In this section we present an ablation study to assess the impact of the footstep candidate cost on GaitNet's performance. We compare two models: the standard GaitNet formulation as described in section 3.4, and a Cost-Ablated-GaitNet trained with the the footstep candidate ($\mathbf{f}_c$) zeroed out in the input. In the GaitNet formulation described previously, we pass the estimated heuristic cost of a footstep action into the model ($\mathbf{f}'_c$ in Figure 3.13), that is, the expected cost of taking the action if it were the only motion the robot was performing. For this ablation study we replace that value with a zero during the training and evaluation process to see how it affects the network without changing the shape of the network at all.

During training, the GaitNet model quickly learns to correlate the lower cost candidates with higher value logits. After this initial learning phase, the model learns to refine its predictions based on other features in the input. This is illustrated in Figure 4.14.

The inspiration for this ablation study comes from observing that the footstep candidate cost and GaitNet output logits become less correlated as training progresses. This suggests that while the footstep candidate cost is useful for initial learning, the model may be locked into a local minimum where it relies too heavily on this feature.

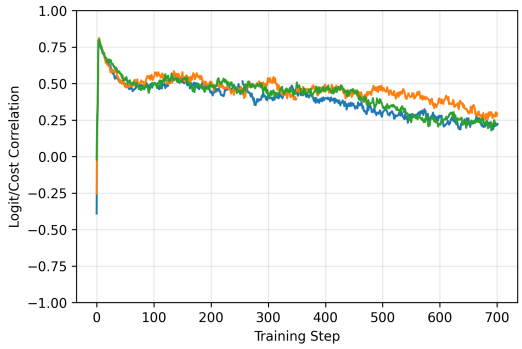


Figure 4.14: Correlation between negative footstep candidate cost and GaitNet output logits during training. Negative costs are used so that a costmap which perfectly captures optimal footstep locations has a correlation of +1.

Now, we train the Cost-Ablated-GaitNet model, over-writing all footstep candidate cost values with zero. The training process is identical to that of the standard GaitNet model, ensuring a fair comparison. The performance of both models is then evaluated using the same metrics outlined in section 4.3.

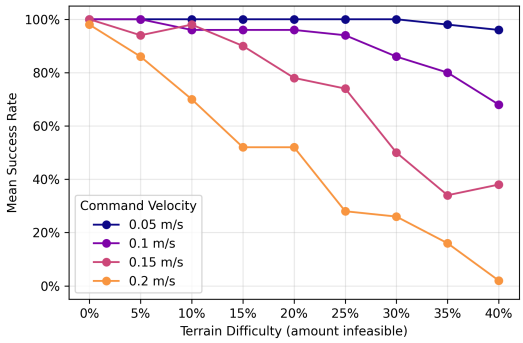


Figure 4.15: Evaluation of Cost-Ablated-GaitNet across various terrain difficulties and commanded velocities. Overall survival rate of 77.7%. Mean success rate measured as the percentage of 50 episodes which completed 20 s without terminating, under the termination conditions described in section A.2.

Figure 4.15 presents the performance of the Cost-Ablated-GaitNet. The results indicates a measurable

performance over the standard GaitNet formulation. GaitNet was able to achieve an overall survival rate of 69.4% (Figure 4.10), while the Cost-Ablated-GaitNet achieved a survival rate of 77.7% (Figure 4.15).

Looking at the mean episode reward in Figure 4.16, we see a similar trend. The Cost-Ablated-GaitNet achieves a higher mean episode reward during training compared to the standard GaitNet. This does come at the cost of initially slower learning, where the Cost-Ablated-GaitNet trails the performance of the standard GaitNet for the first $\sim$75 iterations.

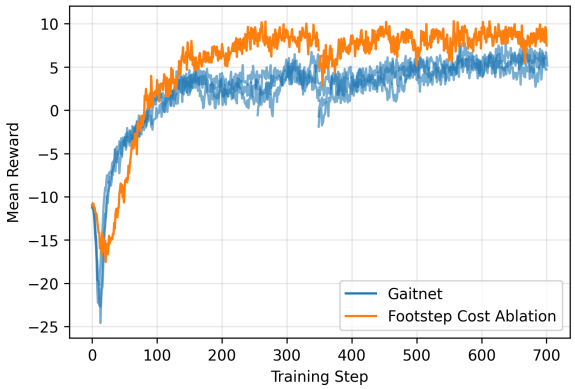


Figure 4.16: Mean episode reward during training for GaitNet and Cost-Ablated-GaitNet.

Interestingly, the Cost-Ablated-GaitNet also demonstrates a slower cadence during training, as shown in Figure 4.17.

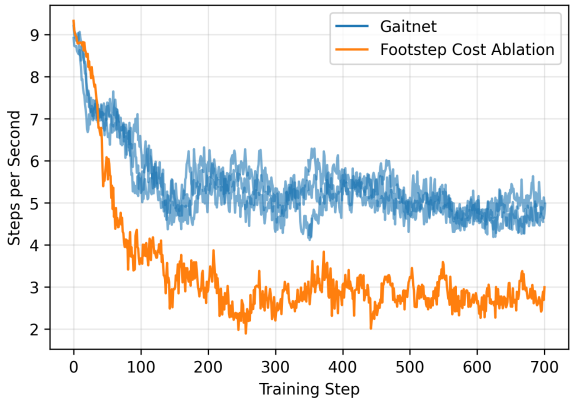


Figure 4.17: Comparison of steps per second during training for GaitNet and Cost-Ablated-GaitNet.

Looking into how this reduced cadence affects the swing schedule, we see in Figure 4.18 that under the same test parameters (10% terrain difficulty and 0.15 m/s commanded velocity), the Cost-Ablated-GaitNet takes less redundant steps—when the same foot is moved multiple times in succession.

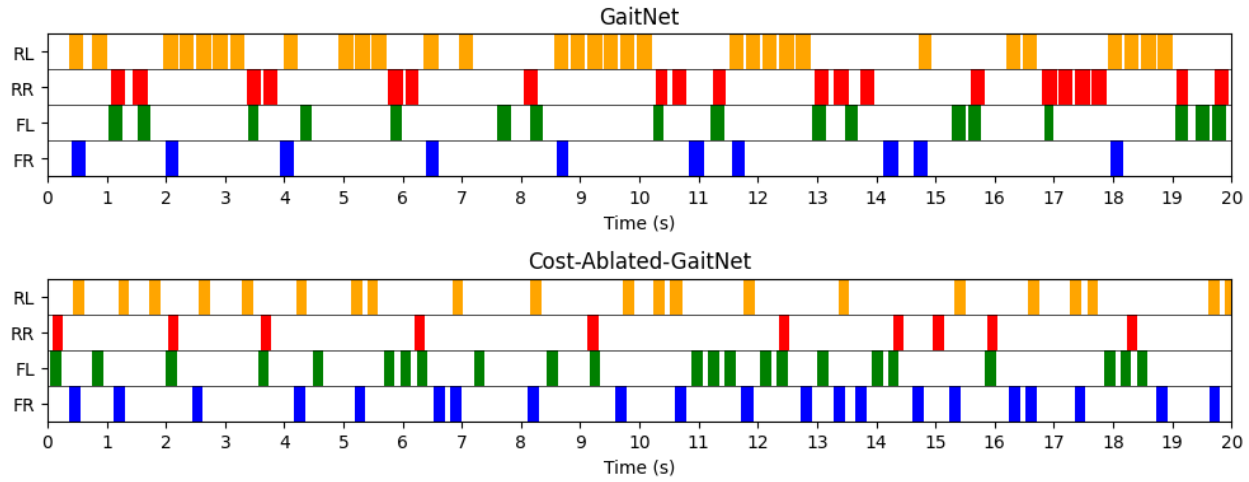


Figure 4.18: Comparison of swing schedules for GaitNet (top) and Cost-Ablated-GaitNet (bottom) at 10% terrain difficulty with a commanded velocity of 0.15 m/s.

4.5.1 Discussion

The action cost ablation study provides perhaps the most significant insight. Counterintuitively, removing the footstep candidate cost from GaitNet's input improves performance from 69.4% to 77.7% survival rate. This result challenges the initial assumption that providing the learned heuristic costs would accelerate training and improve final performance. Several factors likely contribute to this outcome. The correlation analysis (Figure 4.14) shows that while GaitNet initially relies heavily on the cost term (correlation near 0.8), this dependence decreases during training, suggesting the network learns to prioritize other state features. The inclusion of the cost term may constrain exploration, biasing the network toward locally optimal solutions that align with the footstep evaluation network's heuristics but prevent discovery of globally better strategies.

Intuitively this makes sense, as the footstep cost term included in the standard GaitNet model provides information about the robot dynamics, learned from the Footstep Evaluation Network. Without this information, the Cost-Ablated-GaitNet must learn the robot dynamics from scratch, leading to slower initial learning. However, as noted, this allows for more freedom to explore. Moreover, Cost-Ablated-GaitNet exhibits a naturally slower cadence (Figure 4.17) and fewer redundant footsteps (Figure 4.18), indicating more deliberate and efficient gait patterns. This behavior emerges without different reward shaping, suggesting that the model better captures the underlying dynamics when not constrained by pre-computed costs. The slower initial learning observed in Cost-Ablated-GaitNet represents a worthwhile trade-off for superior final performance.

4.6 General Synthesis of Results

These findings collectively suggest that GaitNet's strength lies in its ability to learn coordinated, multi-leg motion strategies through reinforcement learning. The hierarchical architecture successfully constrains the action space to valid movements while allowing sufficient flexibility for dynamic gait generation. However, the system appears to benefit more from direct exploration of state-action relationships than from intermediate learned heuristics. This insight has important implications for hybrid control architectures: while learned perception modules can effectively filter candidate actions based on geometric and kinematic constraints, value estimation may be better left to end-to-end learning that captures the full dynamics of the system.

The results also highlight opportunities for improvement. The relatively modest absolute performance (77.7% best-case survival rate) indicates room for enhancement through reward function refinement, network architecture modifications, or improved low level control. Future work incorporating prediction horizons or more complex terrain may better exploit these capabilities. Overall, this work demonstrates that greedy, neural network-based planners can generate effective dynamic gaits for quadruped robots.

5 Conclusions and Future Work

5.1 Summary of Findings

This thesis presents GaitNet, a novel neural network-based greedy planner for generating dynamic, acyclic gaits in quadruped robots operating in challenging environments. Through a hierarchical hybrid control architecture that combines learned footstep evaluation with reinforcement learning-based gait selection, this work demonstrates that efficient, terrain-adaptive locomotion can be achieved without relying on predefined gait patterns.

GaitNet successfully generates non-periodic, dynamically stable gaits that adapt to varying terrain conditions and commanded velocities. Unlike traditional approaches that rely on fixed gait patterns (trot, walk, etc.), the system learns to coordinate multi-leg motions with variable timing, moving up to two legs simultaneously when conditions permit. The generated swing schedules (Figure 4.3) demonstrate true acyclic behavior, with the robot adjusting its gait pattern in response to both terrain difficulty and commanded velocity without following repetitive cycles.

Comparative evaluation against a single leg motion planner baseline reveals substantial performance improvements. GaitNet achieves a 69.4% survival rate across diverse terrain difficulties and commanded velocities, compared to 25.6% for the baseline method (Figure 4.10 and Figure 4.11). This improvement is particularly pronounced at higher speeds and on more challenging terrain, demonstrating the value of dynamic gait coordination. The baseline method's performance degrades rapidly above 0.05 m/s commanded velocity, while GaitNet maintains robust performance up to 0.15 m/s.

The footstep evaluation network, adapted from ContactNet [1], successfully generates high-quality footstep candidates for the gait selection process. The network demonstrates strong generalization across diverse robot states, accurately identifying low-cost foothold regions even in challenging configurations (Figure 4.1 and Figure 4.2). While not perfect in all scenarios, the network fulfills its role as a candidate generator, providing sufficient diversity for downstream selection.

GaitNet learns to modulate swing durations based on environmental conditions, with mean swing duration of 0.24 s and standard deviation of 0.034 s during evaluation. The system generally selects swing durations between 0.225 s and 0.25 s, but occasionally selects shorter swing durations when needed. However, the swing duration ablation study reveals that this feature provides only marginal performance benefits (2.2% difference in survival rate) in the tested environments, suggesting either that the current scenarios do not fully exploit this capability or that the reward structure does not adequately incentivize optimal duration selection.

Perhaps the most surprising finding emerges from the action cost ablation study, which demonstrates that removing the footstep candidate cost from GaitNet's input actually improves performance. Cost-Ablated-GaitNet achieves a 77.7% survival rate, representing an 8.3 percentage point improvement over the standard formulation. The correlation analysis (Figure 4.14) shows that while GaitNet initially relies heavily on the provided costs, this dependence decreases during training as the network learns to prioritize other state features.

Cost-Ablated-GaitNet exhibits naturally more efficient behavior, taking fewer redundant steps and maintaining a lower overall cadence (Figure 4.17) without explicit reward shaping for these characteristics. This emergent efficiency suggests that when freed from pre-computed heuristic constraints, the reinforcement learning process discovers more globally optimal locomotion strategies. The comparison of swing schedules (Figure 4.18) illustrates this efficiency, with fewer instances of the same foot being repositioned multiple times in succession.

The greedy planning approach enables real-time operation suitable for online control. Unlike MCTS-based or full trajectory optimization methods that require substantial computation for each decision, GaitNet performs a single forward pass through a relatively compact neural network to evaluate and rank discrete action candidates. This computational efficiency is achieved while maintaining robust performance.

The hierarchical design successfully bridges learning-based and model-based control paradigms. By decomposing the problem into footstep candidate generation and gait selection, the architecture provides interpretability and safety through explicit action filtering while maintaining the adaptability of learned policies. The strict constraints on possible actions prevent invalid or unsafe motions while allowing sufficient flexibility for dynamic, terrain-adaptive behavior.

In summary, this work validates the hypothesis that a greedy, neural network-based planner integrated within a hybrid control framework can generate dynamic, acyclic gaits for robust quadruped locomotion. The findings provide valuable insights into the design of hybrid locomotion controllers, particularly regarding the role of learned heuristics and the balance between constraint and flexibility in action selection. While absolute performance leaves room for improvement, the demonstrated capabilities and unexpected insights from ablation studies establish a foundation for future development of adaptive legged locomotion systems.

5.2 Limitations

While the results presented in this work are promising, several limitations should be acknowledged. First, the proposed framework has only been validated in simulation. Real-world deployment may introduce additional challenges, such as unmodeled dynamics, sensor noise, and hardware constraints, which could affect performance. Second, GaitNet operates in a greedy, no-lookahead fashion, limiting its effectiveness at higher speeds or in highly dynamic environments where predictive planning could improve stability and efficiency. Additionally, the system's performance is constrained by the accuracy of the low-level controller in executing footstep placements. Errors in tracking or positioning can degrade the overall effectiveness of the generated gaits. Finally, GaitNet's performance is inherently tied to the quality of footstep candidates produced by the footstep evaluation network; suboptimal candidate sampling may restrict both the diversity and effectiveness of the generated actions.

These limitations highlight clear directions for future research, including real-world testing on physical hardware, the incorporation of predictive planning strategies, and improvements to both candidate generation and low-level control precision.

5.3 Future Work

Several avenues exist to extend and improve the current framework. First, real-world deployment remains a crucial next step. Testing GaitNet on physical hardware would expose the system to real-world dynamics, sensor noise, and hardware constraints, providing valuable insights for further refinement.

Swing re-planning is another potential improvement. Currently, GaitNet does not re-plan during the swing phase. Incorporating real-time re-planning would allow the robot to adjust to unexpected disturbances or terrain changes, though this would require a more robust low-level controller capable of accurately tracking the modified footstep trajectories.

Long-horizon planning also presents an opportunity for enhancement. GaitNet is presently trained to generate actions for a single point in time, which limits the MPC's ability to predict future contact states accurately; the MPC expects the robot to revert to a nominal stance after each swing. While this is acceptable at low speeds, it could become problematic at higher velocities, where predictive planning would improve stability and performance.

Currently, the MPC runs on the CPU, limiting the speed of reinforcement learning. Leveraging GPU-accelerated MPCs, as explored in recent works [47, 48], could significantly accelerate training.

GaitNet also selects only one action at a time, leading to slightly staggered swing start times. Initial attempts to select multiple simultaneous actions encountered difficulties with gradient flow and learning stability. Developing a reliable method for multi-action selection could enable more fluid and dynamic gait patterns, though it would require careful network and training design.

Finally, improvements to the action candidate sampler could enhance GaitNet's performance. At present, action selection relies on sampling-based methods, which may not always identify the highest-quality actions. This is seen in Figure 4.14 by the fact that GaitNet relies less and less on the footstep candidate cost as training progresses. Directly searching the action space using techniques such as projected gradient descent or other optimization strategies could improve solution quality at the expense of additional computation. Another possibility is to train a candidate selection network jointly with GaitNet to improve the diversity and relevance of sampled actions.

5.4 Final Remarks

This thesis set out to explore whether a hybrid control pipeline, combining a greedy neural network-based planner with a model-based controller, could generate dynamic and acyclic gaits for quadruped locomotion. Through the design, training, and evaluation of *GaitNet*, this work demonstrated that such an approach is not only feasible but can yield robust and adaptive locomotion behaviors in challenging simulated environments. The results validate the central hypothesis that hybrid architectures can bridge the gap between the adaptability of learning-based methods and the reliability of model-based control.

Beyond performance metrics, the findings carry broader implications for the design of locomotion systems. The success of the greedy planning strategy highlights the potential of lightweight, inference-efficient neural networks for real-time decision-making in high-dimensional control problems.

Equally important are the insights into learning dynamics and representation. The discovery that removing pre-computed cost terms improves performance suggests that hybrid controllers should not merely layer learned modules on top of existing heuristics, but rather allow networks to develop their own internal representations of value and risk through exploration. As such, the future of legged locomotion control may depend less on handcrafted structure and more on architectures that enable learned coordination within well-defined safety boundaries.

In closing, this work contributes to the growing evidence that data-driven methods can coexist effectively with analytical control frameworks, each compensating for the other's weaknesses. *GaitNet* provides a concrete example of how this integration can produce dynamic, adaptive, and interpretable motion strategies—paving the way for future quadruped systems capable of operating reliably in unstructured, real-world environments. Continued efforts toward real-hardware validation, richer environments, and predictive planning will further advance the vision of truly agile, autonomous legged locomotion.

References

[1] A. Bratta, A. Meduri, M. Focchi, L. Righetti, and C. Semini, "ContactNet: Online Multi-Contact Planning for Acyclic Legged Robot Locomotion," arXiv, Sept. 2022.

[2] J. J. . Di Carlo, Software and control design for the MIT Cheetah quadruped robots. PhD thesis, Massachusetts Institute of Technology, 2020.

[3] H. Chai, Y. Li, R. Song, G. Zhang, Q. Zhang, S. Liu, J. Hou, Y. Xin, M. Yuan, G. Zhang, and Z. Yang, "A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach," Biomimetic Intelligence and Robotics, vol. 2, p. 100029, Mar. 2022.

[4] Y. Fan, Z. Pei, C. Wang, M. Li, Z. Tang, and Q. Liu, "A Review of Quadruped Robots: Structure, Control, and Autonomous Motion," Adv. Intell. Syst., vol. 6, p. 2300783, June 2024.

[5] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control," arXiv, Sept. 2019.

[6] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-Based Control for Dynamic Legged Robots," arXiv, Nov. 2022.

[7] M. Geisert, T. Yates, A. Orgen, P. Fernbach, and I. Havoutis, "Contact Planning for the ANYmal Quadruped Robot using an Acyclic Reachability-Based Planner," arXiv, Apr. 2019.

[8] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization," IEEE Rob. Autom. Lett., vol. 3, pp. 1560–1567, Feb. 2018.

[9] M. Gurram, P. K. Uttam, and S. S. Ohol, "Reinforcement Learning For Quadrupedal Locomotion: Current Advancements And Future Perspectives," arXiv, Oct. 2024.

[10] L. Bao, J. Humphreys, T. Peng, and C. Zhou, "Deep Reinforcement Learning for Bipedal Locomotion: A Brief Survey," arXiv, Apr. 2024.

[11] Z. Wang, W. Wei, A. Xie, Y. Zhang, J. Wu, and Q. Zhu, "Hybrid Bipedal Locomotion Based on Reinforcement Learning and Heuristics," Micromachines, vol. 13, p. 1688, Oct. 2022.

[12] Z. Xie, X. Da, B. Babich, A. Garg, and M. van de Panne, "GLiDE: Generalizable Quadrupedal Locomotion in Diverse Environments with a Centroidal Model," in Algorithmic Foundations of Robotics XV, pp. 523–539, Cham, Switzerland: Springer, Dec. 2022.

[13] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control."

[14] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," arXiv, Oct. 2020.

[15] O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and Continuous Foothold Adaptation for Dynamic Locomotion through CNNs," arXiv, Sept. 2018.

[16] L. Amatucci, J.-H. Kim, J. Hwangbo, and H.-W. Park, "Monte carlo tree search gait planner for non-gaited legged system control."

[17] I. Taouil, L. Amatucci, M. Khadiv, A. Dai, V. Barasuol, and G. Turrisi, "Non-Gaited Legged Locomotion With Monte-Carlo Tree Search and Supervised Learning," IEEE Rob. Autom. Lett., vol. 10, pp. 1265–1272, Dec. 2024.

[18] Y. Meng and C. Fan, "Hybrid Systems Neural Control with Region-of-Attraction Planner," arXiv, Mar. 2023.

[19] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in 2014 IEEE-RAS International Conference on Humanoid Robots, pp. 18–20, IEEE, 2014.

[20] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernandez-Lopez, and C. Semini, "Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization," arXiv, Apr. 2019.

[21] R. Akizhanov, V. Dhédin, M. Khadiv, and I. Laptev, "Learning feasible transitions for efficient contact planning," arXiv, July 2024.

[22] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter, "Learning Agile Locomotion on Risky Terrains," arXiv, Nov. 2023.

[23] D. Zhang, X. Chen, Z. Zhong, M. Xu, Z. Zheng, and H. Lu, "A novel multi-gait strategy for stable and efficient quadruped robot locomotion."

[24] H. Shi, Q. Zhu, L. Han, W. Chi, T. Li, and M. Q.-H. Meng, "Terrain-aware quadrupedal locomotion via reinforcement learning."

[25] H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, and J. Siekmann, "Sim-to-Real Learning of Footstep-Constrained Bipedal Dynamic Walking," in 2022 International Conference on Robotics and Automation (ICRA), pp. 23–27, IEEE, 2022.

[26] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning."

[27] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion," arXiv, Sept. 2020.

[28] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and Efficient Locomotion via Learned Gait Transitions," arXiv, Apr. 2021.

[29] H. Sun, J. Yang, Y. Jia, and C. Wang, "Online Hierarchical Planning for Multicontact Locomotion Control of Quadruped Robots," IEEE/ASME Trans. Mechatron., vol. 30, pp. 1718–1728, July 2024.

[30] C. Gaspard, G. Passault, M. Daniel, and O. Ly, "FootstepNet: an Efficient Actor-Critic Method for Fast On-line Bipedal Footstep Planning and Forecasting," in 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 14–18, IEEE, 2024.

[31] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning," ACM Trans. Graph., vol. 36, pp. 1–13, July 2017.

[32] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning."

[33] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," vol. 38, no. 5, pp. 2908–2927.

[34] S. Omar, L. Amatucci, G. Turrisi, V. Barasuol, and C. Semini, "Fast Convex Visual Foothold Adaptation for Quadrupedal Locomotion," arXiv, July 2023.

[35] L. Chen, P. Du, P. Zhan, and B. Xie, "Gait Learning for Hexapod Robot Facing Rough Terrain Based on Dueling-DQN Algorithm," International Journal of Computer Science and Information Technology, vol. 2, pp. 408–424, Mar. 2024.

[36] Q. Yao, J. Wang, D. Wang, S. Yang, H. Zhang, and Y. Wang, "Hierarchical Terrain-Aware Control for Quadrupedal Locomotion by Combining Deep Reinforcement Learning and Optimal Control," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2021–01, IEEE.

[37] A. Meduri, M. Khadiv, and L. Righetti, "DeepQ Stepper: A framework for reactive dynamic walking on uneven terrain," in 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 2021–05, IEEE.

[38] Z. Gao, X. Chen, Z. Yu, C. Li, L. Han, and R. Zhang, "Global footstep planning with greedy and heuristic optimization guided by velocity for biped robot," Expert Syst. Appl., vol. 238, p. 121798, Mar. 2024.

[39] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," Int. J. Rob. Res., vol. 30, pp. 175–191, Jan. 2011.

[40] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," Int. J. Rob. Res., vol. 30, no. 2, pp. 236–258, 2011. Publisher: SAGE Publications.

[41] M. Asselmeier, Y. Zhao, and P. A. Vela, "Steppability-informed quadrupedal contact planning through deep visual search heuristics."

[42] S. Omar, L. Amatucci, V. Barasuol, G. Turrisi, and C. Semini, "SafeSteps: Learning Safer Footstep Planning Policies for Legged Robots via Model-Based Priors," in 2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids), pp. 12–14, IEEE, 2023.

[43] Y. Zhuang, "rl-mpc-locomotion," Nov. 2025. [Online; accessed 17. Nov. 2025].

[44] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments," arXiv, Jan. 2023.

[45] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 3, pp. 1341–1360, 2021.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv, July 2017.

[47] A. L. Bishop, J. Z. Zhang, S. Gurumurthy, K. Tracy, and Z. Manchester, "ReLU-QP: A GPU-Accelerated Quadratic Programming Solver for Model-Predictive Control," arXiv, Nov. 2023.

[48] B. Plancher, GPU Acceleration for Real-time, Whole-body, Nonlinear Model Predictive Control. PhD thesis, Harvard University, Cambridge, MA, USA, April 2022.

A GaitNet Training Configuration

A.1 Reward Function Analysis

Designing the reward functions for training GaitNet (see Table A.1) proved to be a challenging task. Various combinations of rewards often resulted in undesirable behaviors, such as the robot frequently lifting its feet while idle or dragging feet during locomotion. Achieving a natural gait required careful balancing, particularly between *op_reward* and *mdp.foot_slip_penalty*.

Function[1]	**Weight**	**Description**
mdp.is_alive	0.4	Reward for being alive
mdp.track_lin_vel_xy_exp	0.5	Reward for tracking linear velocity in the XY plane
mdp.track_ang_vel_z_exp	0.5	Reward for tracking angular velocity around the Z axis
op_reward	-0.1	Penalty for performing actions
mdp.is_terminated	-200	Penalty for termination
mdp.lin_vel_z_l2	-2.5	Penalty for linear velocity in the Z direction
mdp.ang_vel_xy_l2	-0.1	Penalty for angular velocity in the XY plane
mdp.flat_orientation_l2	-8	Penalty for non-flat orientation
mdp.foot_slip_penalty	-6	Penalty for foot slip

Table A.1: Reward functions used to train GaitNet.

In addition to the rewards listed in Table A.1, several other reward functions were explored but ultimately not used.

The *a_foot_in_swing* reward was intended to encourage the agent to frequently lift its feet during early learning. However, it proved unnecessary, as the initial network weights already favored foot movement. Furthermore, including this reward caused the agent to move its feet excessively when idle.

The *no_op_reward* was designed to encourage the agent to remain idle when no useful actions were available. In practice, this reward required an excessively high weighting to have any effect, at which point it would overshadow other rewards. The *op_reward* was found to be a more effective alternative for achieving the desired behavior.

A.2 Termination Functions

The termination functions used in training GaitNet are summarized in Table A.2. These functions help define the conditions under which an episode ends, either applying the termination penalty or simply signaling an invalid state.

[1]Functions named "mdp.*" are built-in functions provided by the NVIDIA Isaac Lab framework.
[2]Functions named "mdp.*" are built-in functions provided by the NVIDIA Isaac Lab framework.
[3]Time Out indicates whether the termination applies the mdp.is_terminated penalty.

Function[2]	Time Out[3]	Description
mdp.time_out	True	Terminate at the end of the episode
mdp.bad_orientation	False	Terminate if the robot's orientation is too far from upright
mdp.root_height_below_minimum	False	Terminate if the robot's base height is too low
mdp.terrain_out_of_bounds	True	Terminate if the robot leaves the terrain bounds
foot_in_void	False	Terminate if any foot steps into the void

Table A.2: Termination functions used to train GaitNet.

A.3 Commands

The command used in training GaitNet is summarized below. This command defines the highest level input to the environment, $\mathbf{u}$.

- **UniformVelocityCommand**: This command samples a desired velocity vector $\mathbf{v} = [v_x, v_y, \omega_z]$ from a uniform distribution.
  - resampling time range: (2.5, 10) s
  - v_x range: (-0.2, 0.2) m/s
  - v_y range: (-0.2, 0.2) m/s
  - ω_z range: (-0.4, 0.4) rad/s
  - probability of zero command: 0.05

A.4 PPO Hyperparameters

The PPO hyperparameters used for training GaitNet are summarized in Table A.3. While not all parameters were rigorously tuned, they were found to perform well in practice. The discount factor *gamma* was specifically selected in relation to the agent observation frequency of 25,Hz to provide a reasonable effective horizon. The *learning_rate* was chosen relatively high to compensate for the slower data collection speed.

Hyperparameter	Value
clip_param	0.3
num_learning_epochs	8
num_mini_batches	4
value_loss_coef	0.5
entropy_coef	0.02
learning_rate	3e-4
max_grad_norm	1.0
use_clipped_value_loss	True
gamma	0.99
lam	0.95

Table A.3: Hyperparameters used for PPO training.