On-robot Deep Reinforcement Learning for Quadruped Locomotion

Tiefes verstärkendes Lernen zur Fortbewegung von Vierbeinrobotern in der realen Welt
Master thesis by Jonathan Kinzel
Date of submission: December 13, 2024

1. Review: Nico Bohlinger
2. Review: Daniel Palenicek
3. Review: Jan Peters
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

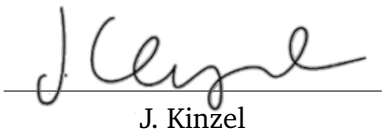
Hiermit erkläre ich, Jonathan Kinzel, dass ich die vorliegende Arbeit gemäß § 22 Abs. 7 APB der TU Darmstadt selbstständig, ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe mit Ausnahme der zitierten Literatur und anderer in der Arbeit genannter Quellen keine fremden Hilfsmittel benutzt. Die von mir bei der Anfertigung dieser wissenschaftlichen Arbeit wörtlich oder inhaltlich benutzte Literatur und alle anderen Quellen habe ich im Text deutlich gekennzeichnet und gesondert aufgeführt. Dies gilt auch für Quellen oder Hilfsmittel aus dem Internet.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 13. Dezember 2024



J. Kinzel

Acknowledgements

I would like to express my deepest gratitude to the team at MAB Robotics for their invaluable support, knowledge, feedback, resources, and time, as well as for providing the robot used in this research. Special thanks for hosting me over the summer of 2024 in Poznań, Poland, for four memorable months, which turned out to be one of the best experiences of my life. I learned immensely, had great fun, and am incredibly grateful to have met each one of you.

A heartfelt thank you to Łukasz Antczak, for making this opportunity possible, for being an exceptional host and supervisor, and for the enjoyable lunches. Many thanks to Jakub Matyszczak for his guidance as a supervisor, to Jakub Bartoszek for consistently providing feedback and invaluable industry insights, and to Krzysztof Kwapisz for his assistance with experiments, setup, and his friendship throughout my stay. I am also grateful to Ania Boltruczuk, who helped immensely with the organization and logistics of my stay, Michał Raszewski, for his skill in repairing the robot and 3D printing the AprilTag Cube, and to Paweł Korsak for his help with the NVIDIA Jetson Nano Orin.

I extend additional thanks to my academic supervisor, Nico Bohlinger, for his exceptional support, insightful feedback, and the opportunity to work with MAB Robotics. I am also grateful to Daniel Palenicek for his valuable perspectives, and to Prof. Dr. Jan Peters, for giving me the chance to work within the Intelligent Autonomous Systems Group and with MAB Robotics.

Lastly, I wish to thank my family and Hannah for their unwavering support, understanding, and encouragement, especially during my four months abroad. Thank you all for standing by me during this journey and for your patience despite our prolonged time apart.

Abstract

Deep reinforcement learning presents a promising framework for enabling autonomous agents to learn effective policies in uncontrolled environments without requiring extensive domain knowledge. However, sample inefficiency has traditionally confined its applications to simulated environments. This thesis addresses these limitations by advancing on-robot reinforcement learning techniques that optimize sample efficiency, enabling quadruped robots to learn locomotion behaviors directly in real-world settings. Leveraging recent advancements in machine learning algorithms and carefully tuned robot controllers, the proposed approaches enable rapid learning of walking gaits across diverse terrains, including indoor and outdoor environments known to challenge classical model-based controllers. The work evaluates these methods through experiments that highlight the interplay between algorithmic design, sensory feedback, and environmental constraints. Results demonstrate the ability to consistently achieve stable and efficient locomotion within limited training time while addressing challenges such as noisy state estimation and terrain variability. This research contributes to the field by providing insights into achieving practical and adaptable quadruped locomotion, bridging the gap between simulation and real-world deployment.

Zusammenfassung

Tiefes verstärkendes Lernen bietet einen vielversprechenden Ansatz, um autonome Agenten in unkontrollierten Umgebungen effektive Verhaltensstrategien erlernen zu lassen, ohne umfangreiches Domänenwissen zu benötigen. Allerdings war die Anwendung bisher aufgrund ineffizienten Nutzung von Trainingsdaten hauptsächlich auf simulierte Umgebungen beschränkt. Diese Arbeit adressiert diese Einschränkungen durch die Weiterentwicklung von Techniken für verstärkendes Lernen, die die Dateneffizienz optimieren und es vierbeinigen Robotern ermöglichen, direkt in realen Umgebungen Fortbewegungsverhalten zu erlernen. Unter Verwendung aktueller Fortschritte in maschinellen Lernalgorithmen sowie präzise abgestimmter Robotercontroller ermöglichen die vorgeschlagenen Ansätze ein schnelles Erlernen von Gangarten auf verschiedenen Untergründen, darunter Innen- und Außenbereiche, die als besonders herausfordernd für klassische modellbasierte Steuerungen gelten. Die Methoden werden in Experimenten evaluiert, die die Wechselwirkungen zwischen algorithmischem Design, Sensorfeedback und Umgebungsbedingungen aufzeigen. Die Ergebnisse zeigen, dass es möglich ist, innerhalb kurzer Trainingszeit stabile und effiziente Fortbewegung zu erreichen, wobei Herausforderungen wie ungenaue Zustandsschätzungen und Geländeschwankungen adressiert werden. Diese Forschung liefert einen Beitrag zum Verständnis und zur Realisierung praktischer und anpassungsfähiger Fortbewegung quadrupeder Roboter durch tiefes verstärkendes Lernen und schließt die Lücke zwischen Simulation und realer Anwendung.

Contents

Figures and Tables

List of Figures

List of Tables

List of Abbreviations

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
AS	Action Space
OS	Observation Space
SAC	Soft Actor-Critic
UTD	Update-To-Data Ratio
PDEC	Progressive Directional Expansion Curriculum
NN	Neural Network
IMU	Inertial Measurement Unit
MPC	Model Predictive Control
CPG	Central Pattern Generator
CPG-D	Central Pattern Generator with Constant Frequency
CPG-RT	Central Pattern Generator with Residual Target
CPG-VF	Central Pattern Generator with Variable Frequency
JT	Joint Target
JT-NT	Joint Target with No Filter
JT-OEF	Joint Target with One Euro Filter
JT-LPF	Joint Target with Low-Pass Filter

1 Introduction

Quadruped robots hold significant potential for navigating and performing tasks in rough and uncertain terrain, with applications spanning search and rescue missions, remote area explorations, and industrial inspections [1, 2]. Achieving reliable locomotion in these challenging environments requires robots to adapt to varied surfaces and obstacles. As a result, numerous approaches have been explored to enable robust and adaptable locomotion. These strategies generally fall into two categories: white-box approaches, such as Model Predictive Control (MPC) [3], which rely on predefined, interpretable models of the robot's dynamics, and black-box, data-driven methods, such as deep reinforcement learning (DRL), which leverage large-scale data to learn complex behaviors [4, 5]. While MPC methods offer interpretability and some theoretical guarantees, they often require precise models of the robot and the environment, which can be difficult to obtain for complex tasks. In contrast, data-driven approaches like RL offer the promise of flexibility and adaptability, making them suitable for complex locomotion challenges such as traversing rough terrain, climbing obstacles, and executing jumps [6, 7, 8].

RL algorithms can be trained to handle diverse locomotion tasks, adapting to unpredictable scenarios without explicit programming [9]. However, many RL approaches are trained in simulation due to the high sample efficiency required for real-world training. Simulation allows extensive data collection and model tuning without risking hardware damage [10, 11, 12], but it also introduces challenges, as simulated environments can only approximate real-world conditions. This gap between simulation and reality, known as the *reality gap*, leads to inaccuracies in the model, which researchers attempt to mitigate through domain randomization. Moreover, inference in RL models can be computationally demanding compared to control-based approaches like MPC, which typically benefit from more efficient real-time performance and come with performance guarantees.

A compelling solution to this challenge is *on-robot learning*, which focuses on training and refining RL models directly on the physical robot in real-world environments. On-robot learning addresses the limitations of simulation by exposing the RL models to the robot's actual operating conditions, ensuring that the learned policies are adapted to real-world dynamics. However, on-robot learning is constrained by hardware limitations, such as processing power, and requires careful time management due to the physical wear on the robot and the expense of each real-world interaction. Given these constraints, sample efficiency becomes critical: algorithms must achieve robust performance with minimal training data.

Furthermore, real-time on-robot data presents additional challenges, as state estimations from noisy and potentially unreliable sensors introduce uncertainty into the training process. Accurate state estimation is crucial for effective decision-making, and the reliance on sensor data requires RL algorithms to contend with inherent inaccuracies in real-world measurements. Consequently, advancing sample-efficient algorithms that can cope with noisy data and operate in real-time is essential to achieving effective on-robot learning for quadruped locomotion. This thesis investigates the development of such algorithms to enable quadruped robots to learn adaptable and robust locomotion behaviors directly on the robot, bringing the field closer to practical and resilient deployment in real-world applications.

2 Foundations

2.1 Universal Function Approximators

Deep learning, a subset of machine learning, has emerged as a powerful approach for modeling complex patterns and representations in data. At its core are *deep neural networks*, which are artificial neural networks composed of multiple layers of interconnected nodes or neurons. These networks are capable of approximating complex functions by learning from data and are considered universal function approximators due to their ability to approximate any measurable function to an arbitrary degree of accuracy, given sufficient data and computational resources [13].

Mathematically, a neural network defines a function $f(x;\theta)$, where x represents the input data and θ denotes the set of parameters (weights and biases) that define the network. The learning process involves finding the optimal parameters θ^* that minimize a loss function $\mathcal{L}(f(x;\theta), y)$, where y is the target output. This optimization is typically performed using algorithms like stochastic gradient descent and its variants.

Deep learning has proven effective across various tasks, including regression and classification. Beyond these foundational applications, specialized architectures have expanded deep learning's capabilities in handling specific data types and problems. Autoencoders, as unsupervised networks, learn compressed representations by encoding data into a latent space and reconstructing it, which aids in dimensionality reduction, anomaly detection, and generative modeling. Convolutional Neural Networks (CNNs) are optimized for grid-like data, such as images, where they learn spatial feature hierarchies, excelling in computer vision tasks like image classification and segmentation [14, 15]. Recurrent Neural

Networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM) networks, model sequential data by capturing information from previous inputs, making them ideal for tasks like language modeling and time series prediction [16]. Transformers have revolutionized natural language processing through self-attention mechanisms, enabling models to assess the relevance of different input parts [17]. This architecture has led to breakthroughs in language understanding and generation, with models like BERT [18] and GPT [19] setting new performance standards.

2.2 Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning focused on training agents to make sequential decisions in an environment in order to maximize a cumulative reward signal [20]. The primary goal of reinforcement learning is to derive an optimal policy that dictates the best action an agent should take in any given state to maximize the expected sum of future rewards. This approach enables the agent to learn through interactions with the environment, receiving feedback in the form of rewards, which guide the learning process [20, 21].

RL problems are often modeled as a Markov Decision Process (MDP), where the environment is defined by a set of states S, a set of actions A, a transition function $P(s'|s,a)$ that describes the probability of reaching a new state s' given the current state s and action a, a reward function $R(s,a)$ that specifies the reward obtained after taking action a in state s, and a discount factor γ which determines the importance of future rewards . The agent's objective in this MDP framework is to maximize the expected return, often defined as the cumulative discounted reward over time.

One of the foundational algorithms in reinforcement learning is Q-learning, introduced by *Watkins et al.* [22], which is an off-policy algorithm that seeks to learn the optimal action-value function, $Q^*(s,a)$. The action-value function, or Q-function, represents the expected cumulative reward of taking a specific action a in a given state s and subsequently following the optimal policy. In Q-learning, the Q-value of a state-action pair is iteratively updated using the Bellman

equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right],$$

where α is the learning rate, $R(s,a)$ is the immediate reward, γ is the discount factor, and $\max_{a'} Q(s',a')$ represents the maximum Q-value for the next state s'. By iteratively updating the Q-values in this way, Q-learning converges towards an approximation of $Q^*(s,a)$, enabling the agent to select actions that maximize long-term rewards.

As environments and state-action spaces become more complex, traditional Q-learning struggles due to the vast number of state-action pairs that need to be evaluated and stored. To address this limitation, deep Q-learning (DQN) combines Q-learning with deep neural networks, allowing the Q-function to be approximated by a neural network rather than a table of values. In their seminal work, *Mnih et al*. [23] introduced the DQN algorithm, which demonstrated unprecedented performance by enabling an agent to learn directly from raw pixels to achieve human-level control in Atari games. The network is trained using a replay buffer, where past experiences (state, action, reward, next state) are stored and randomly sampled to break correlations in the data, and a target network that stabilizes training by periodically updating to the current Q-network's weights. Deep Q-learning, as popularized by DQN, has proven effective in complex environments, setting a foundation for advanced applications of reinforcement learning in areas as diverse as gaming [24] and real-world control tasks.

Further advancements in reinforcement learning have sought to address scalability and generalization, particularly for robotics applications. *Kalashnikov et al* [25] introduced QT-Opt, a scalable vision-based reinforcement learning framework, which leverages off-policy training on large datasets to learn complex manipulation tasks such as grasping. By utilizing over 580k grasp attempts across multiple robots, QT-Opt achieved generalization to previously unseen objects. This work underscores the potential of combining Q-learning methodologies with scalable neural network architectures to tackle high-dimensional control problems.

2.3 Soft Actor-Critic

The Soft Actor-Critic (SAC) algorithm [26, 27] 1 is a model-free, off-policy actor-critic reinforcement learning algorithm that achieves high sample efficiency and stable learning in continuous action spaces. Actor-critic methods, in general, use two primary components: the actor, which represents the policy by selecting actions, and the critic, which evaluates the chosen actions by estimating their expected returns. The actor and critic operate concurrently, with the critic providing feedback that helps the actor improve its policy. In SAC, both the policy (actor) and the value functions (critic) are approximated by neural networks, which allow the algorithm to generalize across high-dimensional or continuous state and action spaces.

SAC extends the reinforcement learning objective by employing a maximum entropy framework to promote exploration and robustness. In traditional reinforcement learning, the objective is to find a policy π that maximizes the expected cumulative reward. However, SAC augments this with an entropy term to maximize both the expected reward and the entropy of the policy. This entropy-enhanced objective is represented as:

$$J_{\text{MaxEnt}}(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right)\right],$$

where $\alpha > 0$ is a temperature parameter that balances the reward and the entropy term $\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}_{a_t \sim \pi}\left[\log \pi(a_t|s_t)\right]$. This maximum entropy reinforcement learning approach encourages the agent to prefer more stochastic policies, which increases exploration, particularly beneficial in environments with sparse or noisy rewards.

SAC employs a method known as soft policy iteration, which alternates between policy evaluation and policy improvement steps. In policy evaluation, the Q-function is updated to estimate the expected return for each action while accounting for the entropy. This is achieved by minimizing the soft Bellman residual with respect to the Q-function, where the Q-function update equation incorporates both reward and entropy:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[R(s, a) + \gamma \mathbb{E}_{a' \sim \pi}\left(Q(s', a') - \alpha \log \pi(a'|s')\right) - Q(s, a)\right].$$

In policy improvement, the policy π is updated by minimizing the expected KL divergence between the policy and an exponential of the soft Q-function:

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\theta} \left[\alpha \log \pi_\theta(a_t|s_t) - Q(s_t, a_t) \right] \right],$$

where θ denotes the parameters of the policy network.

An essential component of SAC is the temperature parameter α, which adjusts the trade-off between exploration and exploitation by controlling the weight of the entropy term in the objective. Rather than keeping α fixed, SAC often includes an automatic temperature tuning mechanism to adapt α dynamically, aligning the agent's entropy with a target entropy $\mathcal{H}_{\text{target}}$. The temperature update is performed by minimizing the objective:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_\theta} \left[-\alpha \left(\log \pi_\theta(a_t|s_t) + \mathcal{H}_{\text{target}} \right) \right],$$

where a higher α encourages more random actions, thus increasing exploration, and a lower α focuses the agent on exploitation.

The update-to-data (UTD) ratio [27] in SAC, denoted as κ, is another key parameter that influences the learning dynamics. The UTD ratio represents the number of gradient updates applied to the policy and value networks for each environment interaction step. Adjusting κ allows for improved sample efficiency by controlling how frequently the agent learns from collected data relative to its interaction with the environment. A higher UTD ratio improves sample efficiency by enabling more learning per collected sample, but it can lead to instability if the data in the replay buffer is not sufficiently diverse. Conversely, a lower UTD ratio places more emphasis on data collection and exploration but may reduce learning efficiency.

Beyond the original SAC, several variants have been developed to further improve sample efficiency and stability. Dropout Q-Functions (DroQ) [28] leverages dropout regularization within Q-function updates, reducing overestimation bias and enhancing training stability, particularly in environments with high UTD ratios. Another variant, Randomized Ensemble Double Q-Learning (REDQ) [29], uses an ensemble of Q-functions to provide more accurate value estimates, significantly boosting sample efficiency.

Algorithm 1: Soft Actor-Critic Algorithm

Initialization : policy parameters θ, Q-function parameters ϕ, temperature α, target parameters $\bar{\phi} \leftarrow \phi$

for *each iteration* **do**

  for *each environment step* **do**

    Observe state s_t

    Select action $a_t \sim \pi_\theta(a_t|s_t)$

    Execute action a_t in the environment

    Observe next state s_{t+1} and reward r_t

    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$

  end

  for *each gradient step* **do**

    Sample mini-batch of transitions from $\mathcal{D}$

    Update critic parameters ϕ by minimizing $J_Q(\phi)$

    Update policy parameters θ by minimizing $J_\pi(\theta)$

    Adjust temperature α by minimizing $J(\alpha)$

    Update target network parameters: $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$

  end

end

2.4 CrossQ

CrossQ [30] is a lightweight, sample-efficient algorithm designed for continuous control tasks in reinforcement learning. It builds upon the SAC framework but introduces several key modifications to improve sample efficiency while reducing computational overhead. The core innovations of CrossQ are threefold: (1) the removal of target networks, (2) the careful application of Batch Normalization (BatchNorm) [31] in critic networks, and (3) the use of wider critic networks for faster learning.

In traditional off-policy reinforcement learning algorithms, such as SAC, target networks are used to stabilize learning by providing a delayed update of the critic's Q-value estimates. The target network is updated through Polyak averaging, which introduces computational overhead and slows down the learning process. CrossQ eliminates the need for target networks by leveraging alternative mechanisms to stabilize learning. Instead of using a separate network to estimate the target Q-values, CrossQ computes the Q-value updates directly from the critic network itself. This simplification accelerates training while maintaining stability, which is further enhanced by the use of BatchNorm in the critic network.

Batch Normalization, commonly used in supervised learning, has been shown to improve the stability and convergence of neural networks [32]. However, its application in reinforcement learning has been limited due to issues with distribution shifts between the target and behavior policy. CrossQ overcomes these challenges by concatenating state-action pairs from both the current and next steps, allowing BatchNorm to normalize the combined distributions in a single forward pass. This approach ensures that the normalization statistics are consistent, preventing the performance degradation that can arise from distribution mismatch.

To further improve learning efficiency, CrossQ uses wider critic networks, inspired by recent research demonstrating that wider networks are easier to optimize and lead to better performance in deep learning [33]. The use of wider networks allows CrossQ to extract more robust features, which accelerates the learning process and leads to higher sample efficiency. Unlike prior methods that rely on large ensembles or dropout for bias reduction, CrossQ achieves similar or better performance using these simpler architectural modifications.

2.5 Filtering

2.5.1 Low-Pass Filter

A low-pass filter (LPF) [34], also called Butterworth filter, is a signal processing technique used to attenuate high-frequency components while allowing low-frequency components to pass, thereby smoothing out rapid fluctuations in a signal. The LPF is widely utilized in applications requiring noise reduction, as it effectively suppresses undesired high-frequency noise while preserving the essential characteristics of the signal.

In the context of digital signal processing, an LPF operates by incrementally adjusting the current output $y(t)$ toward the input $x(t)$ over time, based on a cutoff frequency f_c. This cutoff frequency determines the speed of the response and the degree of smoothing applied to the signal. The output $y(t)$ is recursively updated as a weighted average of the current input $x(t)$ and the previous output $y(t-1)$, controlled by a smoothing factor α. Given a time interval Δt, the smoothing factor α can be calculated as follows:

$$\alpha = \frac{1}{1 + \frac{2\pi f_c \Delta t}{1+2\pi f_c \Delta t}}$$

This formula links the smoothing factor directly to the cutoff frequency, allowing for effective control of the filter's responsiveness. The LPF equation then updates the output $y(t)$ as:

$$y(t) = \alpha \cdot x(t) + (1 - \alpha) \cdot y(t-1)$$

By applying the LPF recursively, noise is gradually filtered from the input signal, producing a smoother output. The simplicity and efficiency of the LPF make it particularly suitable for real-time applications in which computational resources are limited, and reliable noise reduction is essential.

A low-pass filter is valuable for smoothing joint angle predictions generated by a neural network in locomotion tasks, particularly because neural network outputs initially resemble Gaussian noise due to random weight initialization. This noise

can lead to erratic, high-frequency fluctuations in the predicted angles, which would destabilize the movement if directly applied. By attenuating these high-frequency components, the low-pass filter reduces noise and stabilizes the predictions, allowing the network to gradually learn coherent, smooth joint trajectories essential for effective locomotion.

2.5.2 One Euro Filter

The one Euro filter (OEF), proposed by *Casiez et al.* [35], is a low-pass filter specifically designed to address real-time signal smoothing challenges. The core principle of the one Euro filter is to dynamically adjust the cutoff frequency based on the rate of change of the input signal. By employing an adaptive cutoff frequency, the filter effectively smooths out noise while preserving relevant high-frequency signal changes.

A description can be seen in algorithm 2. The one Euro filter's cutoff frequency for a signal $x(t)$ is controlled by two parameters: the minimum cutoff frequency f_{min}, which controls the basic level of smoothing, and a frequency β that determines the influence of the rate of change on the cutoff frequency. The filter equation can be described as follows:

The one Euro filter adjusts its cutoff frequency based on the rate of change of the input signal, allowing for effective noise filtering while preserving sharp transitions. Given an input signal $x(t)$, the filter uses the following steps:

The adaptive cutoff frequency f_c allows the one Euro filter to dynamically adjust between noise reduction and responsiveness to rapid changes, achieving a balance that suits real-time applications.

The one Euro filter is ideal for refining joint angle predictions from a neural network because it adapts its smoothing intensity based on the speed of changes in the signal. Early in training, network outputs often resemble random noise, which the one Euro filter can effectively smooth without excessively dampening rapid movements needed for responsive joint control. This adaptive smoothing allows it to better preserve the motion compared to a basic low-pass filter, which may overly smooth the signal and miss critical transitions essential for stable locomotion.

Algorithm 2: One Euro Filter Pseudo Algorithm

Input: Input signal $x(t)$, minimum cutoff frequency f_{min}, tuning parameter β, time step Δt

Output: Filtered signal $y(t)$

Initialization: Initialize $y(0) = x(0)$, $dx_{\text{smooth}}(0) = 0$

for *each time step* t **do**

Compute the rate of change of the signal:

$$dx(t) = \frac{x(t) - x(t-1)}{\Delta t}$$

Smooth the derivative $dx(t)$ using a low-pass filter to obtain a stable estimate:

$$dx_{\text{smooth}}(t) = \text{LowPassFilter}(dx(t))$$

Calculate the adaptive cutoff frequency $f_c(t)$:

$$f_c(t) = f_{\text{min}} + \beta \cdot |dx_{\text{smooth}}(t)|$$

Filter the input signal $x(t)$ using the adaptive cutoff frequency $f_c(t)$:

$$y(t) = \text{LowPassFilter}(x(t), f_c(t))$$

end

return *Filtered signal* $y(t)$

2.6 State Estimation

In this chapter, the problem of state estimation in the context of legged robots is explored. The discussion begins with an introduction to the problem, highlighting the challenges it poses for reinforcement learning. Various approaches to state estimation are then examined, including forward kinematics, IMU fusion through Kalman filtering, learning-based methods, and marker-based tracking systems.

In this thesis, state estimation defines the combination of the sensor data to reflect the current state of the agent, also reflecting the environment. There are two kinds of sensor information: exteroceptive and proprioceptive. Exteroceptive is the information that is gathered from the environment, like camera images or LIDAR scans. Proprioceptive information is the information that is gathered from the agent itself, like joint angles or IMU data. The state estimation is particularly important in the context of reinforcement learning with legged robots because this information is used to judge the current state of agent to determine the next action. An accurate state estimation makes learning easier because the agent does not need to deal with any uncertainties. The current state is also used to assign a credit (reward) to the action that was taken. Therefore, if the state estimation is wrong, the assigned reward is also wrong which can lead to a conflicting signal and which slows down the learning process.

There are several problems, that come with the state estimation in the real world. The two most contributing problems are delay and noise. The delay is caused by the time it takes to process the sensor data and the time it takes to execute the action. The noise is caused by the sensor itself, like the camera or the IMU. The noise can be reduced by filtering the sensor data, but this comes with the cost of delay. Simulation data has the advantage of having the true state of the environment with no delay and noise but does not reflect the real world. Therefore, the state estimation is crucial for the real-world experiments. The available information of the used MAB Robotics HoneyBadger 4.0 relies on purely proprioceptive sensor readings of joint encoder and IMU that give precise information about joint angle information, trunk linear acceleration and orientation relative to gravity. The linear velocity cannot be measured directly and has to be estimated. This is especially important because the reward is given for the achieved velocity. To estimate the trunk linear velocity, we investigated three different approaches, that we are going to evaluate.

2.6.1 Forward Kinematics and Acceleration Fusion via Kalman Filter

One of the most commonly used approaches for velocity estimation is the fusion of forward kinematics with acceleration data from the IMU [36] . In this approach, forward kinematics is used to calculate the velocity of each hip relative to its foot. The overall velocity of the trunk is then obtained by averaging the velocities of the four hips. This serves as the correction step in the Kalman filter. Acceleration data from the IMU, which serves as the measurement, is then used to update and correct the velocity estimates. The velocity of each leg is only considered when the leg is in contact with the ground and producing traction. The velocity and acceleration data are fused using a linear Kalman filter [37, 38]. The primary advantage of this method is its computational efficiency, and the high-frequency sensor data ensures minimal delay.

Foot contact information is essential and is most reliably acquired using torque sensors in the feet. Other methods for estimating foot contact include setting thresholds for end-effector velocity and applied force [39, 40]. In our case, we rely on an onboard method that utilizes planned foot contact information from the gait planner. However, since our model learns joint angles through an end-to-end process, the gait planner's data is unavailable, so we assume that all legs are in contact with the ground. This assumption leads to inaccurate linear velocity estimation, which can indicate the direction of travel but lacks precision.

Typically, during locomotion, two legs are in contact with the ground while the other two are in the air, moving in opposite directions. When averaging the velocities of all four legs, the opposing velocities tend to cancel each other out, resulting in an underestimated velocity that has a slight bias toward the true direction of travel. Consequently, this approach significantly underestimates the actual velocity.

In figure 2.1 the blue line represents the velocity estimation of the Kalman filter fusion and is an example of the reliability. It is apparent that the velocity estimation experiences the same displacement as the ground truth but is bound around the zero velocity. This is due to the fact that in the correction step, the average foot velocity is used, which is always approximately zero.

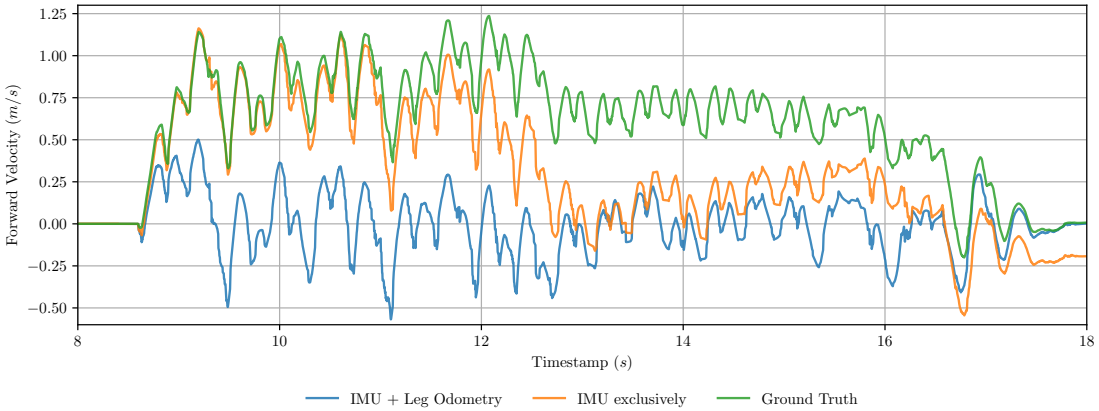


Figure 2.1: Velocity estimation of Kalman filter fusion of forward kinematics and acceleration data from the IMU, pure acceleration integration Kalman filter, compared to ground truth.

2.6.2 Pure Acceleration Integration Kalman Filter

An alternative to the fusion of forward kinematics and acceleration data is to use only the acceleration data from the IMU. This approach tries to not rely on the forward kinematics, which can be inaccurate due to the lack of foot contact information. The acceleration data is integrated using the composite trapezoidal rule to obtain the velocity of the trunk. To be able to accurately calculate the local velocity, the local acceleration needs to be transformed into the global frame with the measured orientation given by the Magnetometer. Additionally, the gravity needs to be compensated. The data is fused using a linear Kalman filter.

The measurement of the acceleration and orientation are noisy and therefore are a source of drift which can only corrected by halting the robot and recalibrating the filter. In figure 2.1 the orange line represents the velocity estimation of the pure acceleration integration Kalman filter. After the initial calibration the velocity estimation is reliable and follows the ground truth. However, the velocity estimation drifts over time and the velocity estimation is not reliable anymore. This is due to the fact that the acceleration data is integrated and the drift of the acceleration data is accumulated over time.

2.6.3 Deep-Learning-Based Velocity Estimation

In this section, we evaluate a deep-learning approach for estimating the trunk velocity of a quadruped robot. This method leverages a neural network trained to predict the velocity of the robot's trunk based on input features such as joint angles and acceleration data from an Inertial Measurement Unit (IMU). The training process employs supervised learning using a dataset of joint angles, acceleration data, and the corresponding ground truth trunk velocities. The primary advantage of this approach lies in its ability to capture complex relationships between the input features and the desired output. This capability can potentially lead to more accurate velocity estimates compared to traditional methods. However, the approach comes with significant limitations, including the need for a large dataset and substantial computational resources for training. For this evaluation, a dataset comprising 2M steps was collected in a simulated environment where an agent was trained to walk in all directions. No noise was introduced into the dataset to ensure clean input-output mappings. The dataset was divided into training and validation subsets with an 80:20 ratio. The neural network architecture consists of three fully connected layers with 256, 256, and 1 neuron, respectively, and ReLU activation functions. The model was trained for 100 epochs using the Adam optimizer with a learning rate of 0.001, and the mean absolute error (MAE) was used as the loss function. The model's accuracy on the validation set is shown in figure 2.2. To be considered adequate, the MAE should be below 0.05. However, this threshold was not achieved in the present experiment. One key limitation of the current approach is the exclusion of critical features such as torque and foot contact information. Without this data, it becomes unreliable to estimate the velocity accurately. Velocity estimation depends heavily on determining whether the robot's feet are in contact with the ground, as this significantly affects the dynamics of locomotion. Such information can be inferred if torque and joint velocity data are included or if torque sensors are integrated into the robot's feet. As *Agarwal etãl.* [41] notes, these features play a crucial role in capturing the dynamics of locomotion, and their absence in the dataset likely contributed to the model's inability to meet the expected accuracy threshold.

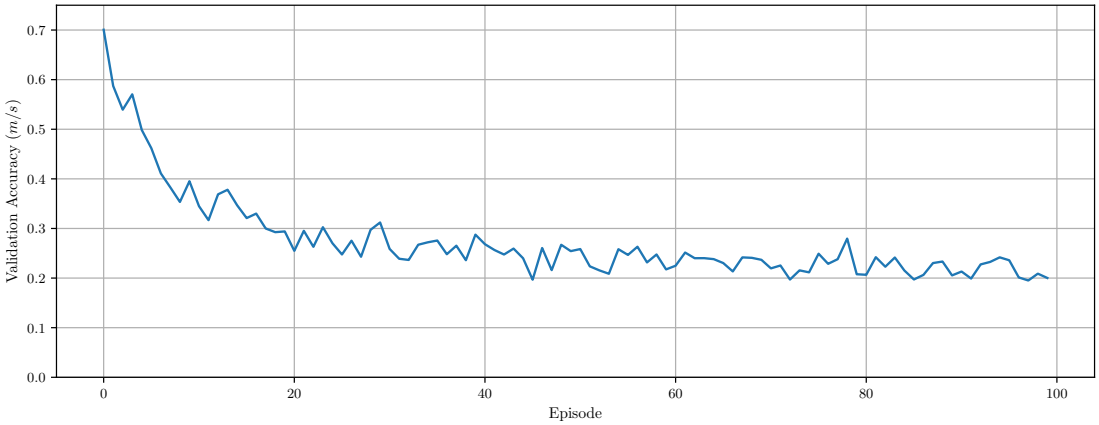


Figure 2.2: Fully connected neural network loss for velocity estimation. Training conducted for 100 epochs and shows saturation at a loss of 0.2.

2.6.4 Marker Based Tracking System

AprilTags [42] are a family of 2D fiducial markers that are widely used in robotics and computer vision applications for precise localization and tracking. They consist of a unique, binary-encoded ID, which aids in reliable detection under various lighting conditions and backgrounds. AprilTags are particularly suited for tasks requiring accurate pose estimation, as they provide a robust means of identifying the 6-DOF (degrees of freedom) position and orientation of the marker relative to a camera. The detection process is efficient and optimized for real-time applications. It involves identifying the tag's square shape in the image, decoding the binary information, and computing the camera's relative pose. This makes AprilTags valuable in environments requiring fast, reliable fiducial detection, such as robotic navigation, SLAM (Simultaneous Localization and Mapping), and augmented reality systems.

This approach tries to compensate the fact that the forward kinematics does not have foot contact information. The idea is to use a marker-based tracking system to track the position of the robot. The webcam of the Laptop is used and pointed towards the robot. The camera is calibrated using the camera calibration toolbox from OpenCV [43]. The camera calibration is used to undistort the images. The AprilTag library with the Python binding provided by Duckietown [44] is used to

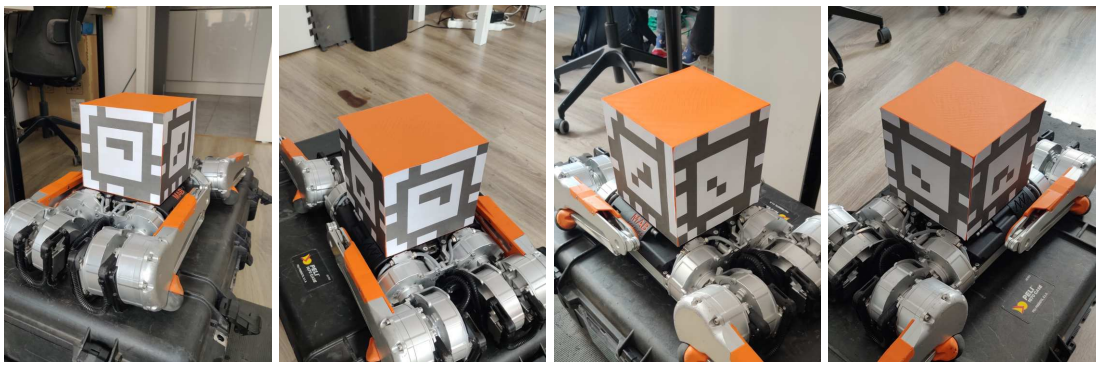


Figure 2.3: Custom made 3D printed cube with AprilTag on each side mounted on the MAB HoneyBadger 4.0.

detect the marker. The marker is a 3D printed cube with an individual AprilTag tagStandard41h12 on each side. The cube is mounted on the robot. The position of the marker relative to the camera is used to calculate the position of the trunk. To obtain the local linear velocity, at first, the position and orientation of the marker is recognized relative to the camera. The best results are achieved by using a Kalman filter and combing the position data with the acceleration data from the IMU. This also means that the velocity estimation increases the frequency from 30 Hz, which is the camera frequency to 416.67 Hz yielded by the IMU. It is necessary that the acceleration data and position data are in the same frame. This can be achieved by transforming the position data from the camera frame to the local frame by using the calculated orientation from the AprilTag. Then, the local position and local linear acceleration can be transformed into the IMU-global frame and fused. This approach was not reliable due to the fact that the estimated orientation from the AprilTag in certain situations had multiple solutions leading to a false estimate. Furthermore, this wrong orientation lead to fluctuations in the local position estimate relative to the camera which was used as the correction step, throwing off the estimated velocity.

Therefore, a different approach was elaborated, that uses the position data in the camera frame that was assumed to be the world frame. The acceleration data cannot be included in this approach because this also needed to be transformed by the estimated orientation which was not reliable. Therefore, this approach used

a linear Kalman filter to estimate velocity by only using the position data. This has the disadvantage that only one type of data was used and that the velocity estimation frequency is the same as the frequency of the camera (30 Hz).

Despite its advantages, tracking a robot using AprilTags comes with several limitations. One of the primary issues is the potential for the marker to be occluded, either by the robot itself or by elements in the environment. This occlusion can result in intermittent tracking or complete loss of the marker, leading to unreliable position estimation. Additionally, false detections may occur due to reflections or the presence of objects with similar visual patterns, which can confuse the detection system. Another significant challenge is the delay introduced by the camera and the computational overhead of processing the tracking data. This delay can negatively impact real-time performance, particularly in applications requiring precise and rapid velocity estimation.

The necessity of equipping the robot with a physical marker can also impose constraints. Depending on the robot's design, the marker may affect its dynamics or restrict freedom of movement. Moreover, using a camera for tracking introduces additional complexity to the system setup. When the camera is used as the reference frame (i.e., assumed to be the world frame), it must be properly aligned with the gravity vector to ensure accurate position and orientation estimation. The distance between the camera and the robot is another constraint. As the robot moves further away from the camera, the accuracy of the position estimation decreases, leading to larger errors. This limitation restricts the operational range of the tracking system, especially in larger environments.

As it can be seen in figure 2.4, it is possible to conduct experiments with this setup. However, it turned out, that this kind of system had too much overhead and did not meet the timing requirements. Therefore, the update frequency was reduced to 10 Hz to account for the delay and reduced tracking rate. Additionally, the mounted cube was damaged multiple times, rendering this approach not feasible for long-term experiments.

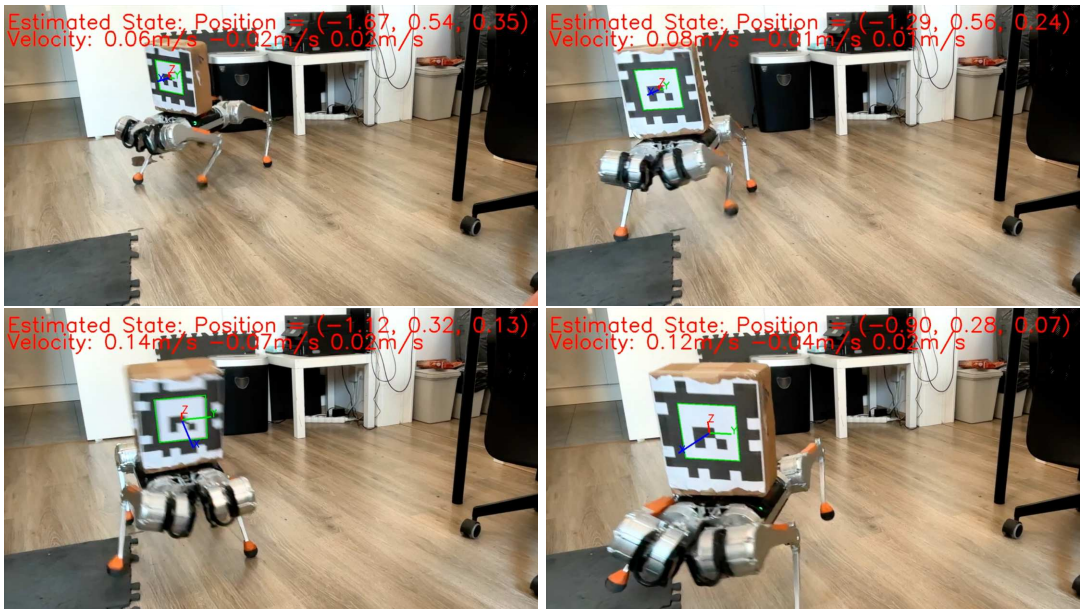


Figure 2.4: CPG-based experiment with the AprilTag tracking system. Update frequency was reduced to 10 Hz to account for delay and reduced tracking rate. The coordinate system visualizes the local linear velocity.

3 Related Work: Deep Reinforcement Learning for Quadruped Locomotion

This chapter discusses the current state of the art in quadruped locomotion, exploring various methodologies and their advancements. The discussion begins with an overview of different approaches to locomotion, including joint angle prediction and central pattern generators. Furthermore, it examines methods that are either trained in simulation and subsequently transferred to real-world applications or learned directly on the robot. Traditional approaches to quadruped locomotion, such as model predictive control, dynamic movement primitives (DMPs) [45], and Whole-Body Control (WBC) [46, 47], have demonstrated significant success in controlling quadruped robots. However, these methods often face challenges related to adaptability and robustness in dynamic or unstructured environments. In contrast, deep reinforcement learning has emerged as a promising alternative, offering the potential for more flexible and adaptive locomotion control strategies.

3.1 Zero-Shot Sim-to-Real Transfer

Recent advancements in deep reinforcement learning have established it as a powerful and robust approach for developing autonomous control policies for complex robotic tasks, especially in quadrupedal locomotion. Traditionally, DRL policies are trained in simulated environments to mitigate real-world risks and expedite development.

Initial studies by *Hwangbo et al.* [48] validated the feasibility of training dynamic locomotion skills for legged robots in simulated settings, achieving policies that

demonstrated robustness and agility. *Lee et al.* [7] furthered this work by developing DRL-based policies capable of navigating rugged terrain, highlighting DRL's utility in tasks that require adaptability and perceptive responses.

A notable shift towards zero-shot sim-to-real transfer has emerged in recent research. *Miki et al.* [49] introduced a robust perceptive locomotion controller that integrates proprioceptive and exteroceptive data via an attention-based recurrent encoder, enabling the robot to autonomously adapt to challenging terrains. Field tests in natural and variable environments, including snow, vegetation, and wet surfaces, demonstrated the controller's ability to maintain stability and high-speed locomotion, even completing an hour-long hike in alpine conditions. This study effectively bridged the gap between simulation-based training and real-world deployment, illustrating the potential of DRL-based controllers to achieve seamless transitions across different environments.

Building on sim-to-real transfer, *Rudin et al.* [50] proposed an end-to-end learning framework that integrates locomotion and local navigation, removing the conventional segmentation of navigation tasks. By focusing on the end-state rather than intermediate waypoints, their unified policy could simultaneously learn multiple behaviors, such as walking, turning, and climbing, leading to improved adaptability and efficiency in real-world environments. The system trained entirely in simulation, successfully transferring zero-shot to natural settings, where the robot navigated varied obstacles, including stairs and inclines.

In a related approach for agile navigation, *Hoeller et al.* [51] developed a hierarchical DRL framework for the ANYmal quadruped, enabling high-speed parkour-style maneuvers, such as jumping and climbing, at speeds up to 2 m/s. Their system comprises perception, locomotion, and navigation modules that estimate terrain features and dynamically select the appropriate skills for each challenge.

Torque-based control has also emerged as a compelling alternative to traditional position-based methods in DRL for legged locomotion. *Chen et al.* [52] introduced a torque-control framework where the RL policy directly predicts joint torques at a high frequency, avoiding the need for PD controllers. This approach enhances the robot's response time and stability on varied terrains, enabling it to resist external disturbances.

Addressing the challenges of agility and adaptability, *Li et al.* [53] proposed Curricular Hindsight Reinforcement Learning (CHRL), a framework that combines an automatic curriculum strategy with an adapted Hindsight Experience Replay (HER) [54]. This combination allowed the controller to perform high-speed maneuvers and quickly recover from perturbations, achieving forward speeds of up to 3.45 m/s and spinning velocities of 3.2 rad/s. The CHRL framework enhances sample efficiency and resilience to disturbances, making it well-suited for varied outdoor terrain.

In the domain of extreme agility, *Margolis et al.* [55] presented an RL-based controller for the MIT Mini Cheetah that sustained high-speed locomotion at 3.9 m/s on grass and 3.4 m/s on gravel and ice. The system uses an adaptive curriculum that gradually increases velocity commands, equipping the quadruped to navigate the dynamic complexities of high-speed running, relying on minimal sensing inputs such as an IMU and joint encoders.

Further extending perception-driven reinforcement learning, *Chen etal.* [56] presented a cross-modal learning framework for estimating terrain physical parameters like friction and stiffness based on visual data. This approach allows quadrupeds to anticipate and adapt to varying terrain without direct contact, reducing energy consumption and enhancing stability on deformable or slippery surfaces, thus advancing the potential for zero-shot transfer in challenging environments.

Research focused on navigating extreme terrains, such as ladder climbing and parkour, has also yielded notable advances. *Vogel et al.* [57] achieved zero-shot ladder climbing using reinforcement learning and a custom end-effector, reaching a 90% success rate across various ladder configurations. Similarly, *Cheng et al.* [58] employed image-based inputs from a depth camera to train a single policy that enabled a quadruped to perform parkour maneuvers, such as high jumps and inclined running, in unstructured environments.

Together, these studies showcase innovative approaches to overcoming the challenges of real-world deployment for DRL-trained controllers. By integrating methods like hierarchical learning, torque-based control, curriculum learning, and adaptive feedback, researchers have developed highly agile, stable, and adaptable behaviors that are increasingly reliable in real-world settings. These advancements show DRL's potential as a robust solution for deploying legged robots in dynamic, unpredictable environments.

3.2 Central Pattern Generators

Nevertheless, DRL alone often suffers from slow learning rates and high sample complexity, particularly when applied to tasks like legged locomotion, which involve rich dynamics and require precise control over numerous degrees of freedom.

To address these limitations, various architectures have been proposed that incorporate domain knowledge into the learning process. One notable example is the use of Central Pattern Generators (CPGs), which have been extensively studied for their ability to generate periodic, rhythmic movements in both animals and robots. *Gay et al.* [59] explored the modulation of CPGs with learned neural network controllers to achieve adaptive locomotion. Similarly, *Sharma and Kitani* [60] exploited the cyclic nature of locomotion by designing phase-parametric policies for reinforcement learning tasks.

Incorporating predefined controllers, such as CPGs, into a learning-based framework has been shown to improve the efficiency and robustness of learned policies. *Tan et al.* [61], for example, demonstrated that combining a feedback balance controller with a user-specified motion generator leads to more stable locomotion behaviors. A limitation of these approaches is that the predefined controllers are often fixed, allowing little flexibility for the learning algorithm to adapt them.

Building on these foundations, *Kasaei et al.* [62] proposed a hybrid locomotion framework that integrates a closed-loop CPG-ZMP-based walk engine with reinforcement learning. Their system adapts walk engine parameters in real-time through a Proximal Policy Optimization (PPO) algorithm enhanced with a Proximal Symmetry Loss function. This innovative framework demonstrated robust omnidirectional locomotion across diverse terrains, showcasing human-like gait adaptability. By leveraging symmetry in the optimization process, the approach improves sample efficiency and generalization, enabling the robot to handle noisy environments and external perturbations effectively.

Similarly, *Bellegarda and Ijspeert* [63] introduced CPG-RL, a method that incorporates CPGs into a deep reinforcement learning framework for quadruped locomotion. This approach enables the agent to directly modulate intrinsic oscillator parameters, such as amplitude and frequency, facilitating real-time

adaptation to environmental changes. The framework was successfully transferred from simulation to real-world scenarios without extensive domain randomization or artificial noise. Demonstrating exceptional robustness, the Unitree A1 quadruped maintained stability while carrying an additional load of 13.75 kg, equivalent to 115% of its nominal mass. Furthermore, CPG-RL allowed for stable omnidirectional locomotion with minimal sensory input, such as foot contact booleans, showcasing its versatility and resilience in practical applications.

3.3 On-Robot Deep Reinforcement Learning

A common limitation of simulation-based approaches is the "reality gap," where policies trained in simulation do not directly transfer to real-world tasks due to differences between the simulated and physical environments. This has led to significant interest in learning directly in real-world settings, bypassing the need for simulation-to-reality transfer.

Addressing the challenge of minimal human intervention in real-world training, *Ha et al.* [64] proposed a system focused on automation and safety, aiming to reduce the need for human supervision. By integrating a safety-constrained reinforcement learning framework within a multi-task setup, their approach enabled a quadrupedal robot to autonomously learn to walk on three distinct terrains: flat ground, a soft mattress, and a doormat with crevices. This system involved a constrained Markov Decision Process (cMDP) that prioritized safety, limiting roll and pitch to prevent the robot from falling. Additionally, their system automated resets, allowing the robot to retry tasks independently and stay within a designated training area by switching between forward and backward walking tasks.

In the domain of real-world robotic learning, several key studies have advanced the field by enabling robots to bypass traditional simulation training. *Haarnoja et al.* [8] introduced a sample-efficient DRL algorithm based on maximum entropy principles, applied specifically to quadrupedal locomotion on the Minitaur robot. Their approach demonstrated that learning policies entirely in real-world settings is feasible, highlighting the robustness of DRL in adapting to real-world conditions without relying on simulation.

Building on this trajectory, *Wu et al.* [65] presented the Dreamer algorithm, which leverages a world model to enable robots to learn from imagined rollouts, thereby reducing the need for extensive trial-and-error interactions in the real world. Dreamer utilizes a learned latent model to simulate future action sequences, allowing for a more efficient policy update process. The researchers applied this model to various robotic platforms, including a quadruped that learned to walk within one hour and adapt to physical perturbations within ten minutes. Their findings demonstrated that world models, when coupled with real-world reinforcement learning, could make on-hardware learning practical and data-efficient, thereby bypassing the limitations of traditional simulators.

Smith et al. [66] further explored real-world deep reinforcement learning by focusing on model-free reinforcement learning to develop a highly efficient training process for quadrupedal locomotion. By implementing carefully optimized task setups and algorithm parameters, they enabled a Unitree A1 quadrupedal robot to learn effective locomotion skills within 20 minutes across diverse terrains, including both indoor and outdoor settings such as flat ground, mulch, grass, and a hiking trail. The system was based on a variant of the Soft Actor-Critic algorithm, specifically the DroQ variant, which provides regularization and layer normalization to improve sample efficiency. After testing various SAC methods, they chose DroQ due to its superior stability and performance on real hardware. The robot achieved a maximal forward speed of 0.44 m/s after approximately 20k update steps, leveraging joint target predictions as actions. To enhance learning efficiency and minimize hardware strain, they restricted the action space to a limited range around a nominal joint position. Action outputs were further smoothed using a low-pass filter, and training was accelerated using JAX, which allowed for just-in-time compilation and optimized execution. Velocity estimation was achieved with a Kalman filter that fused data from an IMU and forward kinematics. The control updates were performed at a frequency of 20 Hz, with high proportional and derivative gains (60 Kp and 5 Kd) applied to ensure stability. An automatic resetting mechanism was used to reposition the robot when it fell, reducing the need for human intervention.

In a subsequent study, *Smith et al.* [67] developed Adaptive Policy Regularization for Learning (APRL), a framework that dynamically modulates the robot's action space to enhance both training stability and sample efficiency. APRL's regularization strategy begins with restricted joint angles that gradually expand

as the policy improves, balancing exploration and safety by constraining early training actions to prevent instability. Using the DroQ variant of SAC, APRL enables the Unitree A1 robot to refine its locomotion skills across challenging terrains, such as inclines, foam, and thick grass, with over 80k training steps, which takes approximately 80 minutes to collect the data. The system achieved an increased maximum speed of 0.62 m/s and utilized an adaptive penalty mechanism, where the dynamics model's prediction errors informed adjustments to the exploration bounds. APRL's training process incorporated periodic neural network resets to maintain the model's plasticity and prevent overfitting to earlier data. Real-time state estimation was supported by an Intel RealSense T265 camera mounted on the robot, which provided drift-free velocity estimates that enhanced stability in complex terrains. The APRL framework demonstrates significant improvements in real-world robotic DRL, enabling the robot to continuously improve its performance in unstructured and variable environments.

4 Method

This chapter introduces the core modifications implemented to enhance performance in quadruped locomotion using on-robot reinforcement learning. Two distinct approaches to locomotion are then explored: Joint Angle Prediction (JT), as similar to the approach of *Smith et al.* [66, 67], and an alternative approach based on Central Pattern Generators (CPG). These methods are critically analyzed, with a focus on their respective advantages, limitations, and the fundamental differences between them. The chapter also delves into the application of on-robot reinforcement learning in the context of quadruped locomotion, addressing the unique challenges and opportunities this approach presents. The discussion begins with an overview of the setup for the deep reinforcement learning algorithm, providing a foundation for understanding its application. Finally, the reward function and curriculum utilized in the experiments are described in detail, highlighting the individual components and their contributions to the agent's learning process.

In this thesis, we aim to investigate the following research questions:

- How can a robust and efficient locomotion strategy be developed for quadruped robots?
- What methods can enable versatility in both direction and speed for quadruped locomotion?
- How effective are Joint Angle Prediction and Central Pattern Generators in enhancing quadruped locomotion?
- What is the performance of on-robot reinforcement learning in the context of quadruped locomotion?

We are going to evaluate different reward functions and curricular for two primary tasks: (1) walking to the forward direction with a constant speed and no yaw velocity (no turning), and (2) walking in any desired direction with a specific velocity. Moreover, the agent must learn a robust gait capable of adapting to various terrains and environmental conditions. To validate the effectiveness of the proposed method, we will evaluate the agent's performance using the MAB Robotics HoneyBadger 4.0 platform (shown in 4.1) in an office and outdoor environment.

Figure 4.1: The MAB Robotics HoneyBadger 4.0 quadruped robot.

4.1 Setup

In this work, we explored a variety of reinforcement learning algorithms based on Soft Actor-Critic, yet our primary focus remains on the CrossQ algorithm, as introduced by *Palenicek et al* [30]. CrossQ offers notable improvements in sample efficiency and computational performance, making it a favorable alternative to other SAC-based approaches. These characteristics are essential for achieving high-performance learning within the constrained time and computational limits typical of on-robot, real-time applications.

The observation space, see table 4.1, for the agent includes approach specific information. If the Joint Target approach is chosen, the agent is solely depended on joint angle positions (corrected by nominal positions). We found that joint angles of $[0.1, 0.6, -1.2]^T$ were suitable nominal positions. Additional, we restrict the action space to $\pm[0.2, 0.4, 0.4]$ of the nominal position. This restriction ensures that the possible configurations represent poses that lead to stable behavior. On the other hand, if a CPG-based approaches is used, a CPG-progress variable is included to give agent information about the state of the walking cycle.

The action space, see table 4.2, again varies depending on the selected approach to enable adaptability in movement control strategies specific to each approach. The neural network architecture deployed in all cases is composed of two layers, each containing 256 neurons, which balances computational efficiency with representational capacity for processing the observation and action spaces effectively.

In RL, the exploration-exploitation trade-off is a well-known issue. The agent must explore the environment sufficiently to discover an optimal policy, while simultaneously exploiting its learned policy to maximize reward. To facilitate efficient exploration, we incorporate techniques such as reward shaping, curriculum learning, action space restriction, and entropy regularization. These methods ensure that the agent explores the environment effectively while still focusing on high-reward actions. The chosen hyperparameter for every used algorithm can be seen in table 4.3 and the respective filter parameter for the JT approaches in table 4.4.

Table 4.1: Observations Space Specifications for JT and CPG Models

Joint Target Prediction (JT)	**Dim**	**Central Pattern Generator (CPG)**	**Dim**
Joint Angle	12	Joint Angle	12
Joint Velocity	12	Joint Velocity	12
Predicted Joint Angle	12	Predicted Feet Position	12
Trunk Linear Acceleration	3	Trunk Linear Acceleration	3
Trunk Angular Acceleration	3	Trunk Angular Acceleration	3
Trunk Linear Velocity	3	Trunk Linear Velocity	3
Trunk Angular Velocity	3	Trunk Angular Velocity	3
Target Velocities	3	Target Velocities	3
Projected Gravity	3	Projected Gravity	3
		CPG Progress	2
Total	54		56

Table 4.2: Action Space Specifications for JT and CPG Models

JT	**CPG-D**	**CPG-VF**	**CPG-RT**
Joint Angle	Feet Position	Feet Position	Feet Position
–	–	Frequency	Joint Angle
12	12	13	24

Table 4.3: Hyperparameter Configurations and Computational Complexity for Different Models

Hyperparameter	SAC	SAC UTD 20	DroQ	CrossQ
Learning Rate	0.003	0.003	0.001	0.005
Batch Size	256	256	256	128
Target Entropy	$-\dim(A)/2$	$-\dim(A)/2$	$-\dim(A)/2$	$-\dim(A)/2$
Inference Frequency	20 Hz	20 Hz	20 Hz	40 Hz
Network Size	256, 256	256, 256	256, 256	256, 256
Ensemble Size	2	2	2	2
Gamma	0.99	0.99	0.99	0.99
Optimizer	Adam	Adam	Adam	Adam
Replay Ratio	1	20	20	1
Computed Updates	2	40	40	2

Table 4.4: Filter Parameters

Filter	None	Lowpass Filter	One Euro Filter
mincutoff	–	0.4	2.5
beta	–	–	0.1
dcutoff	–	–	100

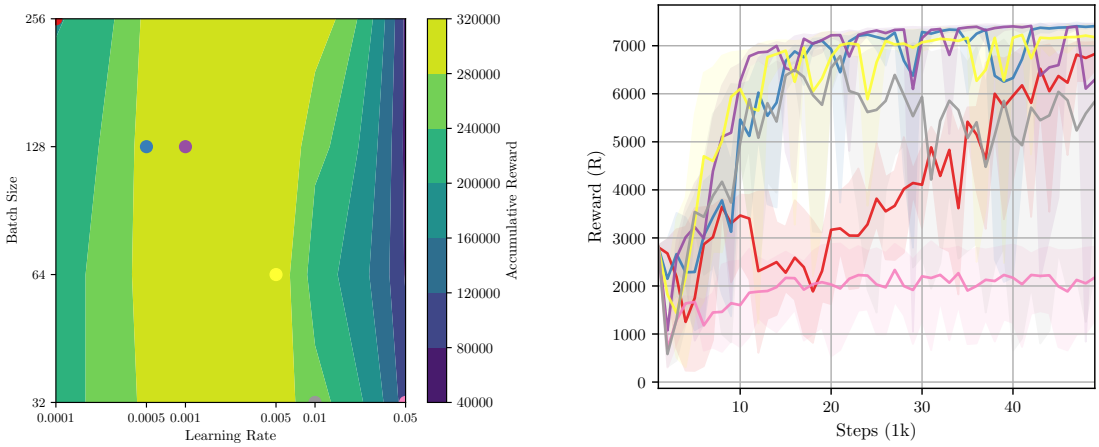
4.2 Joint Angle Prediction

One widely used technique in deep reinforcement learning-based quadruped locomotion is Joint Angle Prediction, also known as Joint Target Prediction (JT). This method involves an agent predicting target joint angles, which are then implemented through a proportional-derivative (PD) controller to produce a stable gait that can adapt to environmental and velocity conditions. The chosen PD gains are critical in determining the agent's ability to learn and perform effective movements, with higher gains enabling more assertive actions to realize the target gait. Lower gains, on the other hand, impose the physical limitations of the robot more strictly, requiring the agent to explore configurations that avoid falls. This exploration of stable configurations is essential, as the agent must encounter and learn to manage potentially destabilizing states within the search space to effectively avoid them.

While force-based control is another option, it is usually applied with control architectures, such as model predictive or whole-body controllers, that consider comprehensive dynamic models of the robot. In the context of RL, however, direct torque-based control is challenging due to the initial exploration phase, where the agent's policy often begins as a random Gaussian distribution. This inherent randomness leads to large, uncoordinated actions that are not only likely to cause falls but also make it difficult for the agent to achieve stable locomotion early in training. For this reason, we incorporate a nominal or default joint position as a reference, which stabilizes the robot in a standing configuration from the start, mitigating erratic movements and reducing the chance of exploration failures in the policy space.

To further improve the agent's control stability, we investigate different types of signal filtering for the output actions, including no filtering, low-pass filtering, and the one Euro filter. Signal filtering is essential to counteract the Gaussian nature of the agent's initial policy, which, without intervention, can lead to abrupt and potentially harmful actuator movements. Such unsmoothed outputs risk producing a high-frequency, vibrational gait that compromises both energy efficiency and the mechanical integrity of the robot's hardware.

Additionally, the adoption of CrossQ within our framework allows us to increase the control loop frequency from 20 Hz to 40 Hz compared to prior implementations,



(a) Accumulative reward of whole run.

(b) Training progress of marked parameters.

Figure 4.2: Hyperparameter search using a grid search approach with learning rates $\in [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05]$ and batch size $\in [32, 64, 128, 256]$. This is a 50k steps experiment using a Joint Target-One Euro Filter approach. Marked runs (a) are shown in (b).

such as those presented by *Smith et al.* This increase in control frequency provides a more responsive and stable gait, which enhances overall performance by reducing the latency between action decisions and their physical implementation. Notably, while the control frequency has been increased, the sample efficiency of the model remains approximately constant, reducing the time required to gather sufficient training samples.

Finally, hyperparameter tuning plays a vital role in optimizing the agent's performance, as it can be seen in figure 4.2. Parameters such as PD gains, filter settings, and reward weights require careful adjustment to strike a balance between stability, energy efficiency, and effective locomotion. Additionally, other hyperparameters, including the learning rate, batch size, entropy coefficient, and action space bounds, significantly influence the agent's behavior and learning efficiency.

4.3 Central Pattern Generators

The Central Pattern Generator, or CPG, is designed to produce a stable trotting gait for quadruped robots by predefining a general gait pattern that minimizes the risk of falling. This predefined pattern is generated by creating a sinusoidal signal for each leg, which is subsequently transformed into motor commands using analytical inverse kinematics. The generated pattern only accounts for the height of the foot, the agent is predicting a positional offset, that leads to the advancement of the robot. The trajectory for the swing phase of each leg is defined by a spline function, specified as follows:

$$f(t_l) = \begin{cases} h \cdot (-2 \cdot t_l^3 + 3 \cdot t_l^2), & t_l \in [0, \pi/2) \\ h \cdot (2 \cdot t_l^3 - 3 \cdot t_l^2 + 1), & t_l \in [\pi/2, \pi) \end{cases}$$

where t_l represents the normalized phase of the gait cycle, and h is the apex height of the trajectory, set to $0.15\,\text{m}$. This apex height, while higher than necessary for many surfaces, has been empirically validated to provide robustness on rough terrain with inclines of up to approximately 3 degrees. The motor command generated by this function is sent to the robot's actuators and executed using the same PD controller employed in the joint angle prediction method. Figure 4.3a and 4.3b demonstrate the feet trajectories over time. To facilitate coordination and gait timing, the observation space of the agent is extended with a CPG phase variable $[l_1, l_2] \in [0, 1]$, tracking the current phase within the gait cycle. While the CPG provides a stable foundational gait pattern, the reinforcement learning agent is responsible for fine-tuning this pattern, enabling the quadruped to adapt dynamically to different terrains and environmental conditions. We explore three CPG operational modes within this study:

- A fixed, predefined gait frequency.
- A variable frequency, adjustable by the agent to match changing requirements.
- A fixed frequency combined with residual learning for fine-grained control.

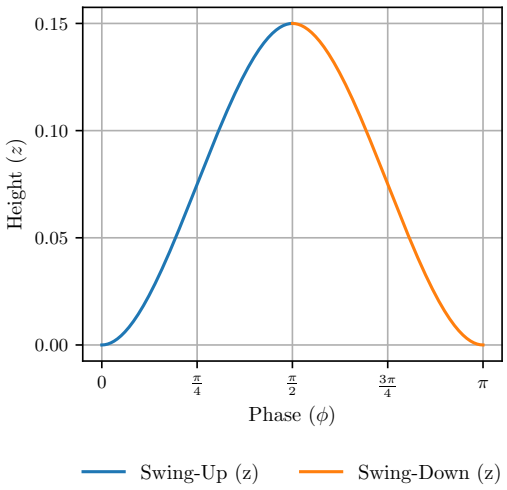
In each mode, the action space is constrained to ensure safe and realistic foot movements. Specifically, the foot offset predictions $[x_i, y_i, z_i]$ are limited to ranges

of $[-0.15, 0.15]$ for x_i and y_i, and $[-0.1, 0.1]$ for z_i. This constraint prevents excessive or destabilizing foot placements. The final position of the foot is calculated as the sum of the predicted offset and the current CPG trajectory, defined by $p = p_{\text{offset}} + p_{\text{cpg}}$. In the adjustable frequency mode, the policy additionally predicts the CPG frequency within a range of $[1.0, 4.0]$, allowing the agent to modify the gait speed as needed.

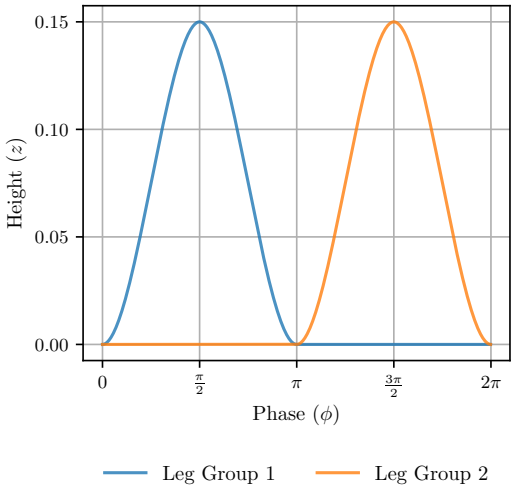
In the residual learning mode, the action space includes both the joint position offsets and the raw joint angles $[x_i, y_i, z_i, \phi_{h_i}, \phi_{t_i}, \phi_{c_i}]$, where joint angle bounds are set to $[\phi_{h_i}, \phi_{t_i}, \phi_{c_i}] \in \pm[0.1, 0.2, 0.2]$. This extension permits multiple integration methods for combining residuals with the CPG-based control strategy, including:

1. A combination of torque outputs from two PD controllers, where the resulting torque $\tau = (\tau_{\text{cpg}} + \tau_{\text{residual}})/2$ averages the CPG and residual components. Here, the residuals are adjusted by the nominal joint position, which provides damping and ensures that the action space remains within safe limits. This configuration is primarily used in simulations.
2. A similar approach applied to joint angles, where $\phi = (\phi_{\text{cpg}} + \phi_{\text{residual}})/2$, used in real-robot experiments. Since the robot's internal control loop operates at 10 kHz, this method leverages the high-frequency internal controller for torque outputs while using a 100 Hz control rate for sending joint angle commands.
3. A summation of the joint angle solution from the CPG and offset approach with the predicted residuals, defined as $\phi = \phi_{\text{cpg}} + \phi_{\text{residual}}$. This method requires careful boundary management to ensure that the resulting joint angles stay within the defined action space.

This CPG-based approach offers the significant advantage that the agent does not need to learn a stable gait pattern from scratch. Instead, the CPG provides a reliable and adaptable baseline for the agent to refine. The RL agent can focus its learning efforts on optimizing the gait for varying terrains and dynamic environments, rather than constructing a gait pattern from the ground up. By contrast, in joint angle prediction without CPG assistance, the agent must develop the entire gait pattern independently, a process that is often inefficient and lacks generalization across different velocities and headings. This lack of generalization requires the agent to learn specific gaits for each direction, increasing both the complexity and time required for effective locomotion.

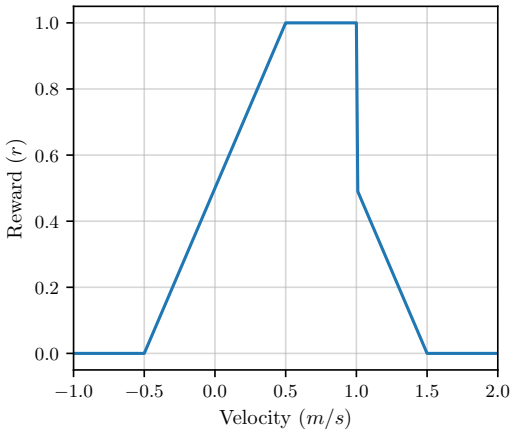


(a) Trajectory for a single foot. Swing-up and swing-down phase are shown.

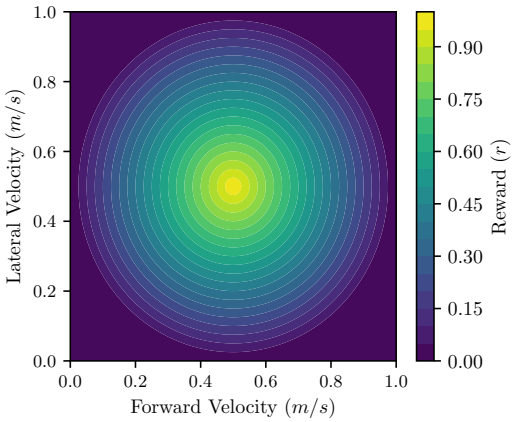


(b) Timings of crossing feet, showing the trot gait.

Figure 4.3: Central Pattern Generator foot-height trajectory and trot gait.



(a) Reward function for x-axis velocity tracking. The reward is linearly shaped from $[-v_t, v_t]$ with a plateau from $[v_t, 2v_t]$.



(b) Two-dimensional velocity tracking reward function. This reward function also emphasizes x- and y-velocities.

Figure 4.4: Target velocity tracking reward function with $v_t = 0.5$ m/s.

4.4 Reward Functions

4.4.1 Forward Locomotion Velocity Reward

The velocity tracking reward employed in this study is derived from the reward function introduced by *Smith et al.* [66], with modifications designed to enhance the agent's performance in stabilizing and optimizing forward locomotion. The primary objective of this reward structure is to encourage the agent to reach a specified forward target velocity, corresponding to the robot's local positive x-axis, while minimizing or eliminating movement in the lateral (local y-axis) direction. This objective simplifies the control demands by focusing on forward motion, allowing the agent to refine its gait along a single axis. Although the reward function does not explicitly penalize lateral velocity, it encourages an efficient, forward-focused gait as the most effective way for the agent to maximize reward. For experimental contexts prioritizing faster convergence, an additional penalty on lateral velocity could be incorporated to explicitly discourage any off-axis movement, effectively reducing the exploratory space in the lateral direction.

The reward function contains multiple terms and can be seen in table 4.5. At its core, a velocity tracking component rewards the agent for achieving a target velocity in the x-axis direction. As it can be seen in figure 4.4a, this term is linearly scaled within a bounded range centered around the target velocity, providing the agent with a signal that guides its actions toward the desired speed. Such a design aids in promoting smooth convergence, as it defines a clear gradient of reward values that guide the agent's gait refinement process. In addition to the forward velocity tracking term, the reward function incorporates penalties that deter actions potentially destabilizing or energy-inefficient. Specifically, a yaw velocity penalty discourages excessive angular rotation, incentivizing the agent to maintain a straight trajectory. A penalty on high torque usage encourages the agent to lift its legs rather than dragging them, promoting a gait pattern that reduces ground impact and minimizes mechanical wear on the robot's feet. This torque constraint also discourages the agent from exerting excessive force, which further promotes energy efficiency and prevents erratic, high-energy movements.

The reward function further penalizes excessive pitch and roll motions in the trunk to discourage exploratory actions that lead to significant trunk rotation, which

can destabilize the robot. By penalizing trunk rotations, the reward structure promotes a stable and balanced posture. Maintaining a stable trunk orientation also reduces the likelihood of sudden loss of balance, implicitly decreasing the risk of falls.

4.4.2 Omnidirectional Locomotion Velocity Reward

The Omnidirectional Locomotion Velocity Reward is an extension of the Forward Locomotion Velocity Reward. This reward incentivizes the agent to move in a specified direction with a desired velocity. While the one-dimensional reward focuses solely on forward velocity along the x-axis, it does not explicitly account for lateral velocities along the y-axis. However, the agent inherently learns to minimize lateral movement over time, as such deviations negatively impact the reward. The primary objective of this reward is to replicate the functionality of the one-dimensional reward while extending its applicability to multidirectional locomotion.

To achieve this, two approaches were developed:

1. A two-dimensional counterpart to the one-dimensional reward. This approach involves rotating the current velocity by the target direction to align along the same direction, followed by calculating the reward based on Forward Locomotion Velocity Reward. This method does not explicitly address velocities that are perpendicular to the target direction.
2. A joint reward formulation that simultaneously tracks and evaluates velocities in both the x- and y-directions, thereby providing a unified incentive for multidirectional movement.

In figure 4.5, Python code for the rotation of the current velocity, as mentioned in approach 1, is presented. Additionally, a Lateral Locomotion Penalty, given by $r_{\text{lat\_penalty}} = -|\mathbf{v}_{t,y}|$, can be applied to discourage any lateral movements relative to the target direction.

Alternatively, as described in approach 2, the reward can be defined as

$$r(v_x, v_y) = \min\left(\max\left(1 - \frac{\|\mathbf{v}_{xy} - \mathbf{v}_{d,xy}\|}{0.5}, 0\right), 1\right)$$

Table 4.5: Forward Locomotion Velocity Reward Terms

Reward		Weight	Explanation
Velocity Tracking $\begin{cases} 1, & \text{for } v_x \in [v_t, 2v_t] \\ 0, & \text{for } v_x \in (-\infty, -v_t] \cup [4v_t, \infty) \\ 1 - \frac{\lvert v_x - v_t \rvert}{2v_t}, & \text{otherwise} \end{cases}$		1	Track a target velocity v_t. See 4.4a.
Yaw Penalty	ω_z^2	-0.1	Penalize high yaw rate ω_z.
Roll and Pitch Penalty	$\theta_{\text{pitch}}^2 + \theta_{\text{roll}}^2$	-10	Prevent excessive rolling or pitching.
Energy Penalty	$\lVert\tau\rVert^2$	-0.0003	Penalize energy consumption based on control input τ.
Total Reward	$r_v(s,a) + r_y(s,a)$ $+r_p(s,a) + r_t(s,a)$	10	Combination of all sub-rewards.
Non-negative Reward	$\max(R(s,a), 0)$	1	Ensure that the reward remains non-negative.

Table 4.6: Forward Locomotion Maximal Velocity Reward Terms

Reward Term	Formula	Weight	Explanation
Target Velocity Reward	$v_x - \lvert v_y \rvert$	2	Incentivizes to increase x-velocity and decrease y-velocity.
Yaw Penalty	$\lvert\omega_z\rvert$	-0.1	Penalize high yaw rate ω_z.
Pitch and Roll Penalty	$\theta_{\text{pitch}}^2 + \theta_{\text{roll}}^2$	-10	Prevent excessive rolling or pitching.
Total Reward	$r_v(s,a) + r_y(s,a)$ $+r_p(s,a) + r_t(s,a)$	10	Combination of all sub-rewards.
Non-negative Reward	$\max(R(s,a), 0)$	1	Ensure that the reward remains non-negative.

The latter reward function is used in the experiments to evaluate the performance of the agent in a two-dimensional velocity tracking task. The reward shaping can be seen in 4.4b with an exemplary target velocity $\mathbf{v}_t = [0.5, 0.5]$.

```
forward = np.array([1, 0])
angle = np.sign(target_y) * np.arccos(local_direction @ target_xy) /
    (np.linalg.norm(forward) * np.linalg.norm(target_xy))
if np.isnan(angle):
    angle = 0
rotation = np.array([[np.cos(angle), -np.sin(angle)],
                     [np.sin(angle), np.cos(angle)]])

rotated_xy = rotation.T @ current_xy
```

Figure 4.5: Rotation of the current velocity to the target velocity. Used to reward movement in the target direction and neglect velocity in the perpendicular direction.

4.5 Curriculum

The ability of humans and animals to learn complex behaviors is greatly enhanced when training examples are presented in a structured and meaningful sequence, progressing from simple to more challenging concepts. This principle, formalized in machine learning as curriculum learning [68], provides a framework for improving the efficiency and effectiveness of training processes by gradually increasing the complexity of tasks. Curriculum learning has been shown to not only accelerate the convergence of training but also to influence the quality of solutions obtained, particularly in the context of non-convex optimization problems encountered in deep learning. By guiding the learning process through an organized sequence of tasks, curriculum learning enables models to achieve better generalization and more robust solutions.

In RL, curriculum learning has proven especially useful for tackling complex tasks that may be too challenging for an agent to learn directly. By decomposing these tasks into smaller, more manageable sub-tasks, the agent is able to progressively acquire the skills required for more intricate behaviors. In the context of quadruped locomotion, a curriculum can be employed to first train the agent to walk in a straight line on flat terrain before introducing more complex scenarios, such as navigating uneven surfaces, inclines, or obstacles. This gradual increase in task difficulty allows the agent to develop a stable and efficient gait that can adapt to diverse environmental conditions, improving its overall robustness and generalization capabilities.

When extending locomotion to include movement in arbitrary directions and achieving a desired yaw velocity, a curriculum becomes even more critical. The complexity of these tasks requires a structured approach to ensure that the agent can build on foundational locomotion skills while incorporating the additional demands of directional control and orientation. To address these challenges, we compare a curricular approach tailored to guide the agent through progressively more complex scenarios to a randomly sampled target direction approach.

4.5.1 Progressive Directional Expansion Curriculum

The curriculum employed in this study is termed the "Progressive Directional Expansion Curriculum" to emphasize its structured approach to incrementally increasing the range of locomotion directions explored by the agent. This curriculum is designed to systematically guide the quadruped robot from simple backward locomotion, exploiting its natural tendencies, to achieving stable movement in all directions on the planar x-y axis. The framework models the range of possible movement directions as a circle, divided into two half-circles representing leftward and rightward directions. Each half-circle is further partitioned into five bins, corresponding to smaller directional increments. The curriculum starts with the robot learning to move backward and then progressively incorporates additional bins adjacent to the already-learned regions, expanding the range of directions in which the robot can reliably move.

Performance tracking is central to this curriculum. The robot's ability to achieve target velocities is evaluated for each bin based on several criteria: achieving at least 95% of the target velocity, maintaining low lateral movement (< 0.05), and minimizing angular velocity (< 0.05). These metrics are averaged over 400 steps, equivalent to half an episode in the environment, to determine whether the robot has mastered the current bin. If the criteria are met, the bin is considered learned, and the curriculum progresses by introducing the next adjacent bin for training. This process continues until the full circle of movement directions is covered, signifying the robot's ability to navigate in any direction.

Sampling new movement directions follows a structured approach to balance exploration and consolidation of learned behaviors. A $2/5$ probability is assigned to selecting the next left bin, $2/5$ to the next right bin, and $1/5$ to randomly sampling from previously learned bins. Within the chosen bin, the direction is sampled uniformly, ensuring gradual and consistent expansion of the movement range.

During the initial stage of the curriculum, the robot's maximum velocity is limited to half of the overall target velocity. This restriction is designed to prioritize the learning of directional movement patterns while avoiding the additional complexity of high-speed locomotion. Once the robot has successfully navigated the full circle of directions, the velocity constraint is removed, allowing the agent to focus on improving speed and efficiency.

5 Experiments

In this chapter we are going to present the experiments that we conducted to evaluate the performance of our proposed approach. The experiments are twofold, simulation and experiments on the real system itself. We will use simulation data to evaluate the performance of the approach because we can observe the true state of the environment. For the real world experiments we use the onboard state estimation that is a Kalman filter fusion of forward kinematics and acceleration data from the IMU. As described in section 2.6 the measured state does not align with the real state and vastly underestimates the current velocity. Therefore, the performance of the robot is evaluated by analyzing the recorded video footage.

5.1 Simulation

The simulation environment that we used for training the robot is based on MuJoCo [69]. We used the MAB Robotics HoneyBadger 4.0 quadruped robot model (see figure 5.1) for training the robot. The experiments were conducted with the parameters that are displayed in table 5.1.

Sim. Freq.	**Act. Delay**	**Dom. Rand.**	**Obs. Noi.**	**Par. Env.**	**Terrain**
200 Hz	$\lvert t + 0.03 \rvert$, $t \sim \mathcal{N}(0, 0.05^2)$	None	None	False	Flat

Table 5.1: Simulation Parameters used in the experiments. Abbreviations: Sim. Freq. (Simulation Frequency), Act. Delay (Action Delay), Dom. Rand. (Domain Randomization), Obs. Noise (Observation Noise), Par. Env. (Parallel Environment).

The experiments conducted in this study are divided into two primary categories: Forward and Multidirectional Locomotion experiments. The experiments are initially conducted in a simulated environment to allow for controlled evaluations and comparisons, and subsequently on a physical quadruped robot to validate real-world performance. Simulation is particularly valuable as it provides a reliable source of velocity measurements, enabling precise performance assessment.

The Forward Locomotion experiments focus on evaluating the robot's ability to move efficiently in a straight line along the x-axis. These experiments are further subdivided into two objectives. In one set, the goal is to achieve and maintain a specified target velocity of 0.5 m/s, assessing the agent's ability to track and stabilize around a predefined speed. In the other, the objective is to determine the maximum achievable forward velocity, providing insights into the potential speed limits of the learned locomotion policies. These experiments allow for a detailed comparison of the presented control methods and their capability to generate stable and efficient forward motion.

The Multidirectional Locomotion experiments, on the other hand, evaluate the agent's ability to move in arbitrary directions on the x-y plane. The aim is to assess the versatility of the locomotion strategies by testing whether the robot can achieve specified velocities in both the x- and y-directions. These experiments also examine the ability to integrate and balance directional control and orientation, crucial for navigating complex environments.

Approach	**Filter/Extension**	**Abbreviation**
Joint Target Prediction	No Filter	JT-NF
Joint Target Prediction	Low-Pass Filter	JT-LPF
Joint Target Prediction	One Euro Filter	JT-OEF
Central Pattern Generator	Default	CPG-D
Central Pattern Generator	Variable Frequency	CPG-VF
Central Pattern Generator	Residual Target	CPG-RT

Table 5.2: Overview of locomotion control approaches, their associated filters or extensions, and corresponding abbreviations.

The approaches that are displayed in table 5.2, introduced in the methodology, are evaluated in these experiments.

All experiments begin with a warm-up phase of 1000 steps to prefill the replay buffer, ensuring that the initial learning process is not hindered by a lack of experience data. In the Forward Locomotion experiments, training is conducted over 50k steps with a target velocity of 0.5 m/s, with the target of conducting 20k steps in real-world experiments corresponding to approximately 9 minutes of wall time. For Multidirectional Locomotion, training runs for 50k steps, equating to about 21 minutes of real time. These time estimates exclude any potential overhead, such as manually resetting the robot or the time required for just-in-time compilation of the learning loop.

To ensure statistical reliability and minimize the impact of outliers, in simulation, each experiment is repeated 10 times. By conducting these experiments in both simulation and real-world settings, this study provides a comprehensive analysis of the locomotion strategies, ensuring their robustness and applicability in practical scenarios.

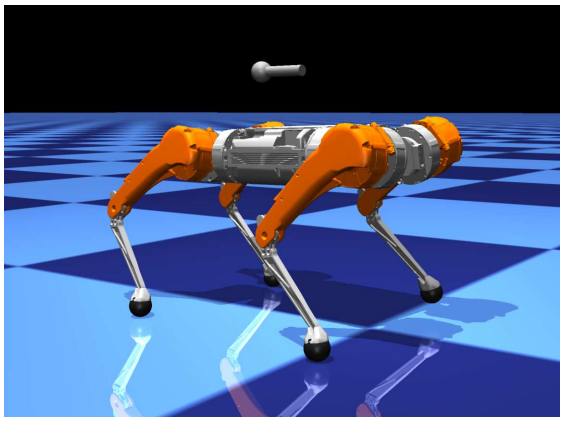


Figure 5.1: The MAB Robotics HoneyBadger 4.0 in the simulation environment.

5.1.1 SAC Algorithm Comparison in Forward Locomotion

This experiment compares different Soft-Actor Critic based algorithms. State-of-the-art algorithm include SAC with a higher update-to-date ratio of 20 update steps, DroQ which implements Dropout layers to support generalization by using regularization. CrossQ is based on batch normalization and crosses out much of the introduced complexity by removing target network and uses a UTD rate of 1. This experiment was conducted in the setting of Joint Angle prediction with no filter applied. The goal of this experiment is to evaluate the performance of the different algorithms and choose a class to continue with. This experiment utilizes the Forward Locomotion Velocity Reward Function and the Forward Locomotion with Target Velocity command function. The target velocity is set to 0.5 m/s. The results are shown in figure 5.2. One of the considerations is the computational complexity of the algorithms, see table 4.3. Due to the increased computation time needed, SAC UTD 20 and DroQ are used with a reduced frequency of 20 Hz. On the other hand, CrossQ runs in real-time with a frequency of 40 Hz, during the training process. This results in a faster data collection which helps to collect the necessary data faster. The plot 5.2 shows the training by wall time by using the frequency of each algorithm. Each episode is limited to 20 seconds. This means that CrossQ executes 800 steps per episode, while the other algorithms execute 400 steps per episode. Additionally, the maximal possible return approximately doubles when using CrossQ compared to the other algorithms. The simulation results do not account for the time needed to reset the robot and only track training time. CrossQ completes the training within approximately 21 minutes, whereas the other algorithms require about 42 minutes. Furthermore, CrossQ learns to walk with the required target velocity after approximately 2 minutes. The next best-performing algorithm, DroQ, requires about 7 minutes to reach the target velocity. On average, they achieve a velocity of around 0.6 m/s. This behavior is necessary to consistently obtain the highest reward, as the velocity exhibits slight variations during the gait cycle. All the algorithms naturally observe that lateral motions do not aid in reaching the goal. Additionally, CrossQ exhibits a lower average termination rate of about 14 falls. Similarly, DroQ performs second best but still has an average termination rate of about 32 falls. Notably, DroQ achieves better performance regarding the yaw and torque penalties, although these have only a slight impact on the overall return. SAC and SAC UTD 20 also reach the target velocity but fail to match the velocities achieved by the other algorithms.

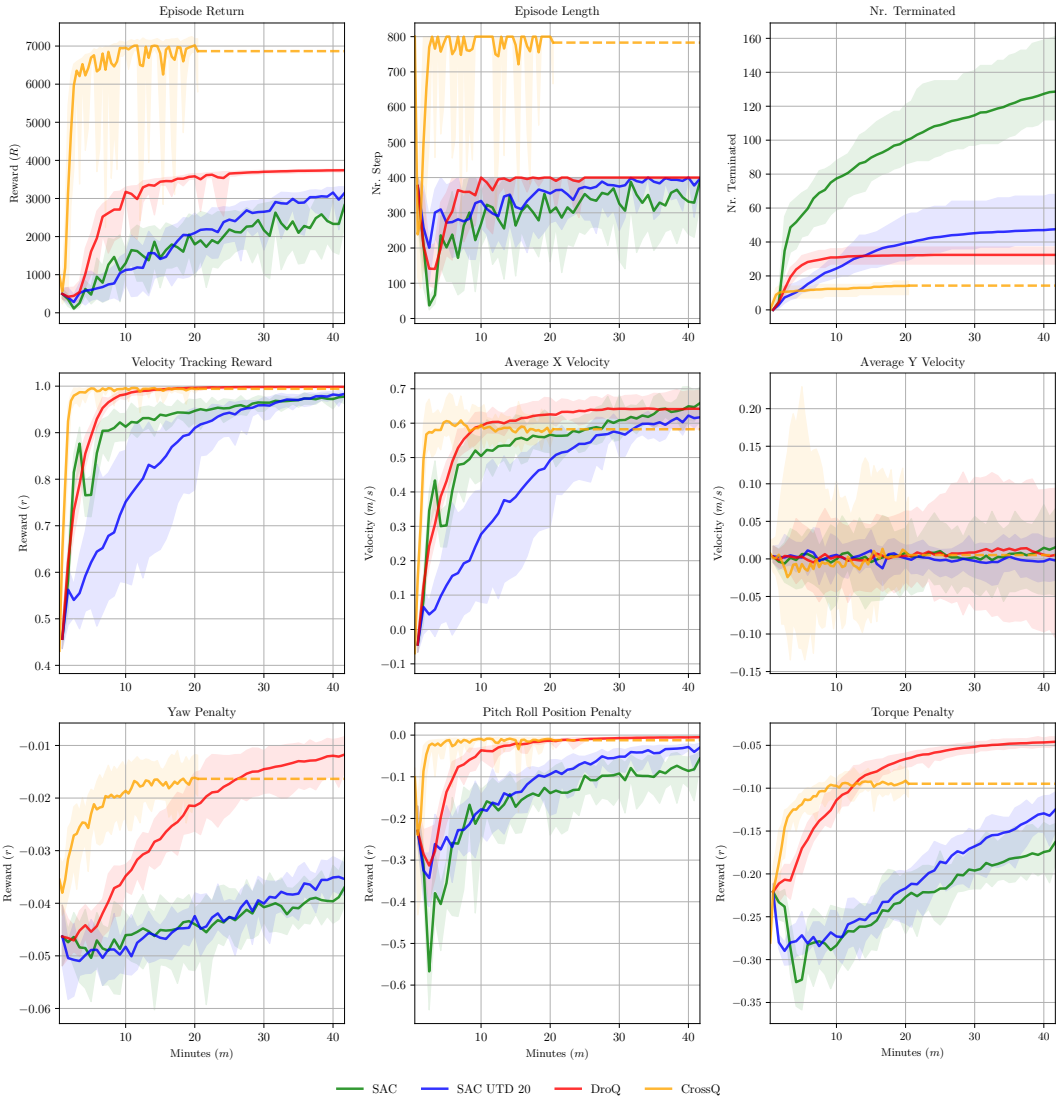


Figure 5.2: Comparison of SAC, SAC UTD 20, DroQ, and CrossQ in the Forward Locomotion experiment. Performance is evaluated over 10 runs, displaying the mean and min/max performance, with time plotted on the x-axis. Theoretical wall time of CrossQ within 21 minutes, compared to the 41 minutes required by the other algorithms.

5.1.2 Forward Locomotion with Target Velocity

This experiment evaluates the performance of various approaches in the Forward Locomotion with Target Velocity scenario, designed to assess their ability to achieve and maintain a desired forward velocity. The results can be seen in figuer 5.3. The target velocity is set at $0.5\,\text{m/s}$, a value chosen as it represents a moderate speed, not the maximum achievable velocity. This setup allows for a comparison with the approach of *Smith et al.*, where a maximum velocity of $0.62\,\text{m/s}$ was achieved under similar conditions. The total training limit for this experiment is set to 50k steps, with fewer steps being desirable to reduce training time. Achieving convergence within 20k steps is considered an optimal outcome.

The general performance of the Joint Target and Central Pattern Generator approaches shows distinct characteristics. JT-based methods exhibit higher walking speeds but are prone to more terminations due to falls, whereas CPG approaches demonstrate greater stability throughout the learning process. The JT-based methods require slightly more training steps to achieve comparable velocities. Overall, both categories of approaches achieve the target velocity within the desired limits.

The JT approaches, specifically JT-NF, JT-OEF, and JT-LPF, converge to the desired velocity at different rates. JT-NF and JT-OEF reach convergence within approximately 5k steps, while JT-LPF takes around 10k steps. However, JT-NF exhibits the highest number of terminations, with an average of 13 falls during training. JT-OEF and JT-LPF exhibit greater stability, averaging approximately 7 terminations each. These observations suggest that JT approaches are capable of rapid learning but require careful management of stability during training.

The CPG approaches demonstrate a more stable and consistent learning process. Among them, CPG-VF achieves the highest velocity and fastest learning speed, converging to the target within 5k steps. CPG-D, on the other hand, achieves slightly lower velocities and requires approximately 10k steps to converge. The CPG-RT approach takes the longest among the CPG methods, exceeding 10k steps to reach the target velocity. Despite these differences, all CPG approaches avoid terminations entirely during training, showing their robustness. Furthermore, the CPG approaches effectively balance velocity tracking and penalty minimization, resulting efficient learning process.

The gait evaluation (see figure 5.4) provides further insights into the behavior of the two categories. The JT-based approaches produce asymmetrical gaits with distinct frequency characteristics. JT-NF results in smaller, higher-frequency steps at approximately 6 Hz, emphasizing the use of the front legs. In contrast, JT-LPF generates larger, lower-frequency steps at around 2.5 Hz, achieving a trot gait where the rear and front legs are synchronized. JT-OEF displays a blend of these behaviors, producing a trot gait with a frequency of 4 Hz.

The CPG approaches exhibit the predefined trot gait encoded by the Central Pattern Generator. CPG-D operates at a frequency of 2 Hz, while CPG-VF adapts to increase the frequency slightly to 2.5 Hz, optimizing for higher velocities. The CPG-RT approach maintains the same frequency as CPG-D but introduces a slight overlap in timings, differentiating it from the default behavior of CPG-D.

In figure 5.5, the forward velocity over a one-second period is shown after training completion. JT-LPF and CPG-VF demonstrate a consistent safety margin, maintaining velocities near the target without significant drops. In contrast, other approaches exhibit occasional velocity drops below the target. These drops are short-lived and minor, except for CPG-D, which shows more pronounced velocity dips, and CPG-RT, which experiences longer durations below the target velocity.

The approach of *Smith et at.* takes about 10k update steps to achieve this velocity. Although a different robot platform was used, the results of the algorithm comparison show a similar performance. Nevertheless, our approach is able to exhibit similar performance within 5k update steps and a higher update frequency.

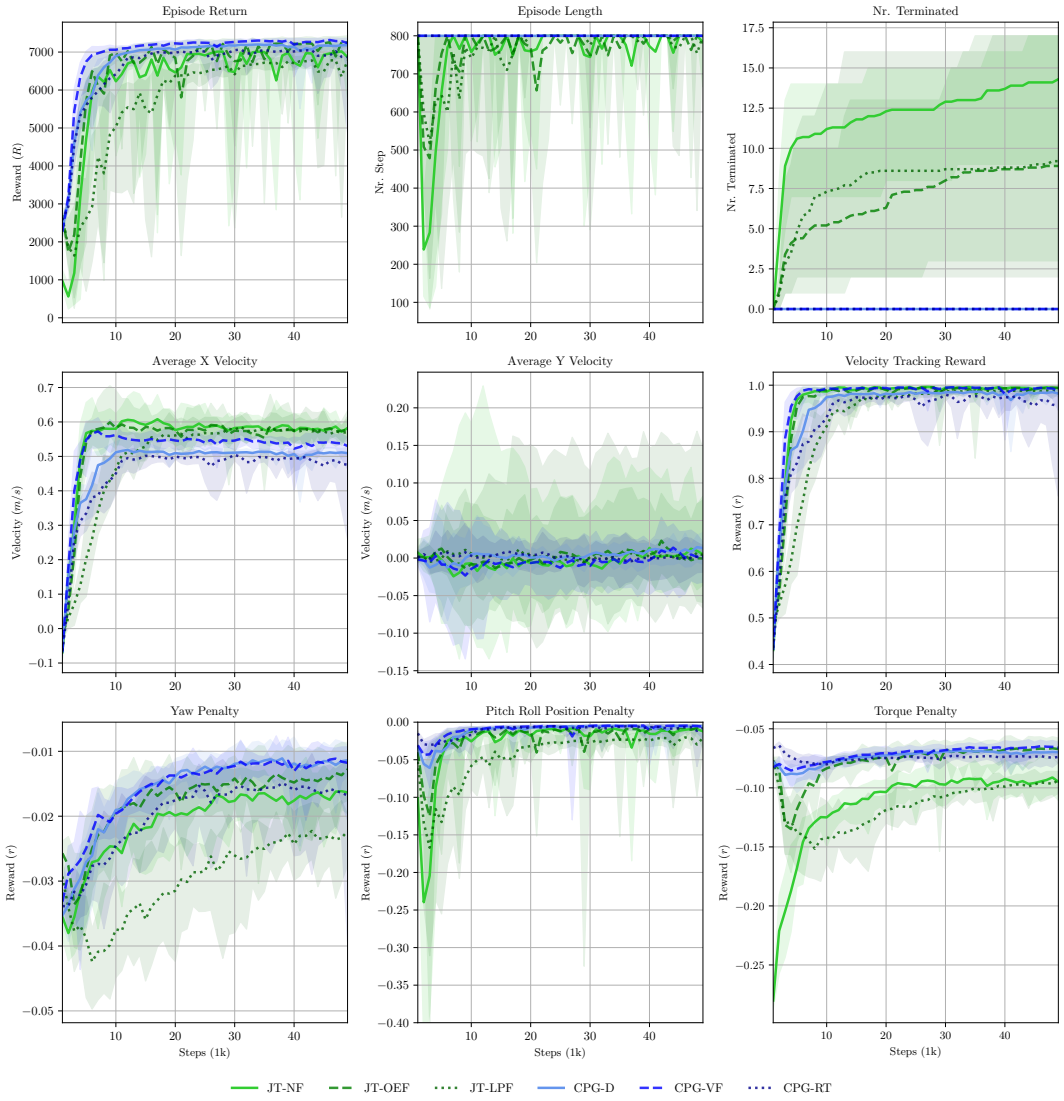


Figure 5.3: Comparison in the Forward Locomotion Experiment, with 10 runs per approach. The values represent the mean of the average episode values across runs, and the shaded area shows the range between the maximum and minimum values.

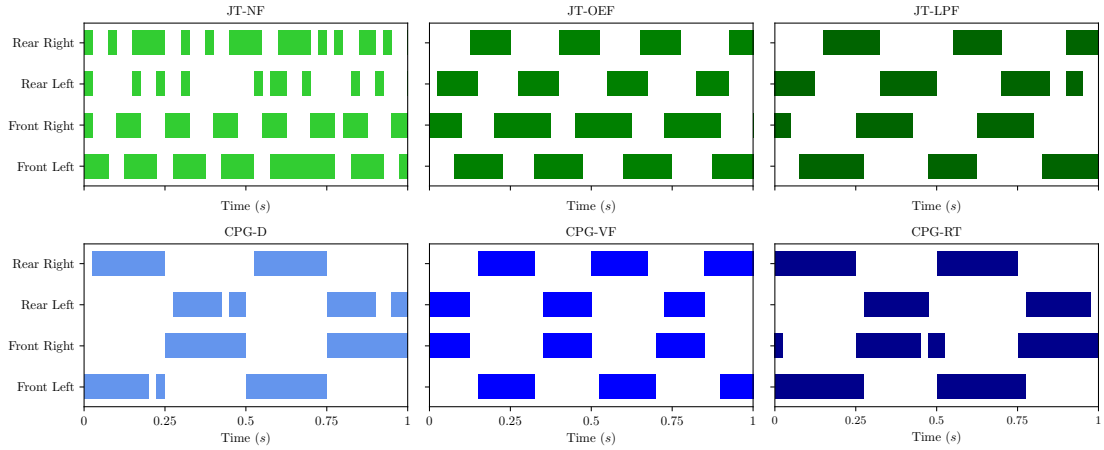


Figure 5.4: Gait evaluation in the Forward Locomotion Experiment. Data was collected after training completion using deterministic sampling from the policy. The figure illustrates the contact points of the feet with the ground over a one-second duration.

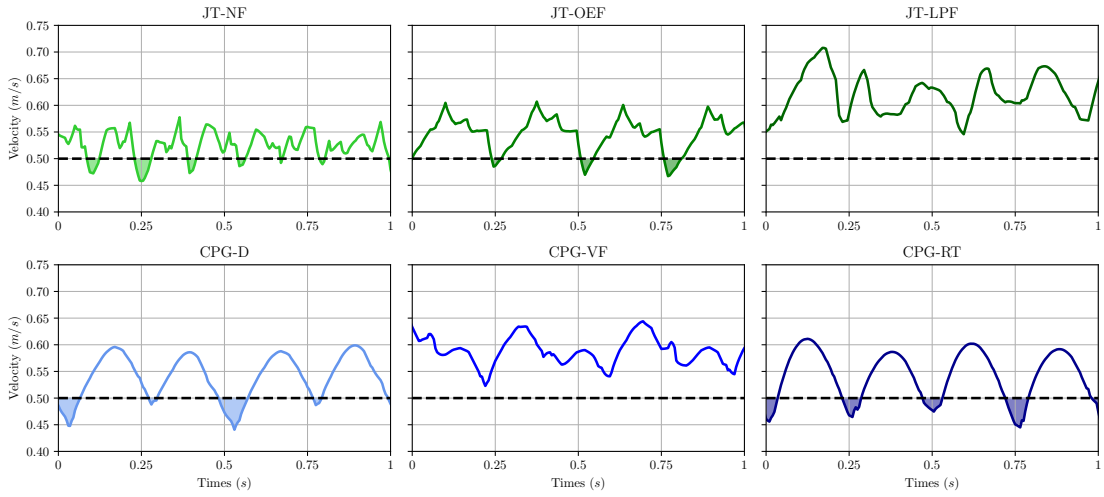


Figure 5.5: Forward velocity of the Joint Target and CPG-based approaches in the Forward Locomotion Experiment. Velocities below $v_x < 0.5$ m/s are highlighted.

5.1.3 Forward Locomotion with Maximal Velocity

This experiment aims to determine the maximum achievable velocity along the forward direction. By designing a custom reward function that incentivizes higher velocities, the agent is guided toward optimizing forward motion while minimizing lateral movement. The terms of the reward function are outlined in table 4.6, and the performance results are shown in figure 5.6. Penalizing lateral movement provides the agent with an explicit signal discouraging sideward motion, which aids exploration by focusing efforts on optimizing forward velocity. Without these constraints, the agent naturally avoids lateral movement, as it does not contribute to improving performance, but penalization actively enforces this behavior and improves training efficiency. This effect can be observed by comparing the lateral velocity in figure 5.3, where the restriction led to reduced exploration and fewer unintentional lateral movements.

The experiment involved training each approach for 50k steps, with all agents exhibiting saturating performance over time. The JT-NF approach achieved the highest maximum velocity at $2.20\,\text{m/s}$, with an average velocity of $1.90\,\text{m/s}$. Next, JT-OEF has a final velocity of $1.3\,\text{m/s}$ and JT-LPF the lowest velocity of $1.15\,\text{m/s}$. The other Joint Target-based approaches followed, exceeding an average velocity of $1.1\,\text{m/s}$. In contrast, the CPG-based approaches achieved both lower maximum and average velocities. Among these, the CPG-VF approach achieved an average velocity of $0.75\,\text{m/s}$, benefiting from higher frequency modulation. The CPG-D and CPG-RT approaches performed similarly, with average velocities of $0.55\,\text{m/s}$.

The differences in velocity performance among the Joint Target approaches can be attributed to the responsiveness of their respective filters. The JT-NF approach, without any filtering, shows the highest responsiveness, which allows it to achieve the highest velocity. However, this responsiveness typically comes at the cost of stability, as evidenced in the Forward Locomotion with Target Velocity experiment (see section 5.1.2). The JT-OEF approach strikes a balance between responsiveness and output smoothing, making it a middle ground between the extremes of no filtering (JT-NF) and low-pass filtering (JT-LPF).

The number of terminations is a critical factor in evaluating the stability of the approaches. The CPG-based methods demonstrate exceptional resilience, with no premature terminations observed throughout the training process. Among the Joint Target approaches, JT-LPF has the highest number of terminations, averaging

around 30, with one extreme case reaching 81 terminations. In contrast, JT-NF and JT-OEF show lower termination rates, averaging approximately 9 falls each. These results highlight the trade-offs between stability and responsiveness inherent in the filter choices for the Joint Target approaches.

The gait evaluation (see figures 5.7 and 5.8) provides further insights into the dynamics of the approaches. The JT-NF approach maintains a trot gait, increasing its step frequency to approximately 4 Hz, which contributes to its higher velocity. The JT-OEF and JT-LPF approaches refine their gaits further, successfully transitioning to a gallop. The increased step frequency and more dynamic motion allow these approaches to achieve higher velocities with greater efficiency.

The CPG-based approaches, in contrast, retain the predefined trot gait generated by the Central Pattern Generator. The CPG-D approach operates at a frequency of 2 Hz, while CPG-VF adapts to a slightly higher frequency of 2.5 Hz, optimizing for increased velocity. The CPG-RT approach maintains a forced frequency of 2 Hz, similar to CPG-D, but introduces a slight overlap in timings, which differentiates its gait pattern. These results underscore the constrained adaptability of the CPG-based approaches, which focus on stability at the expense of higher velocities.

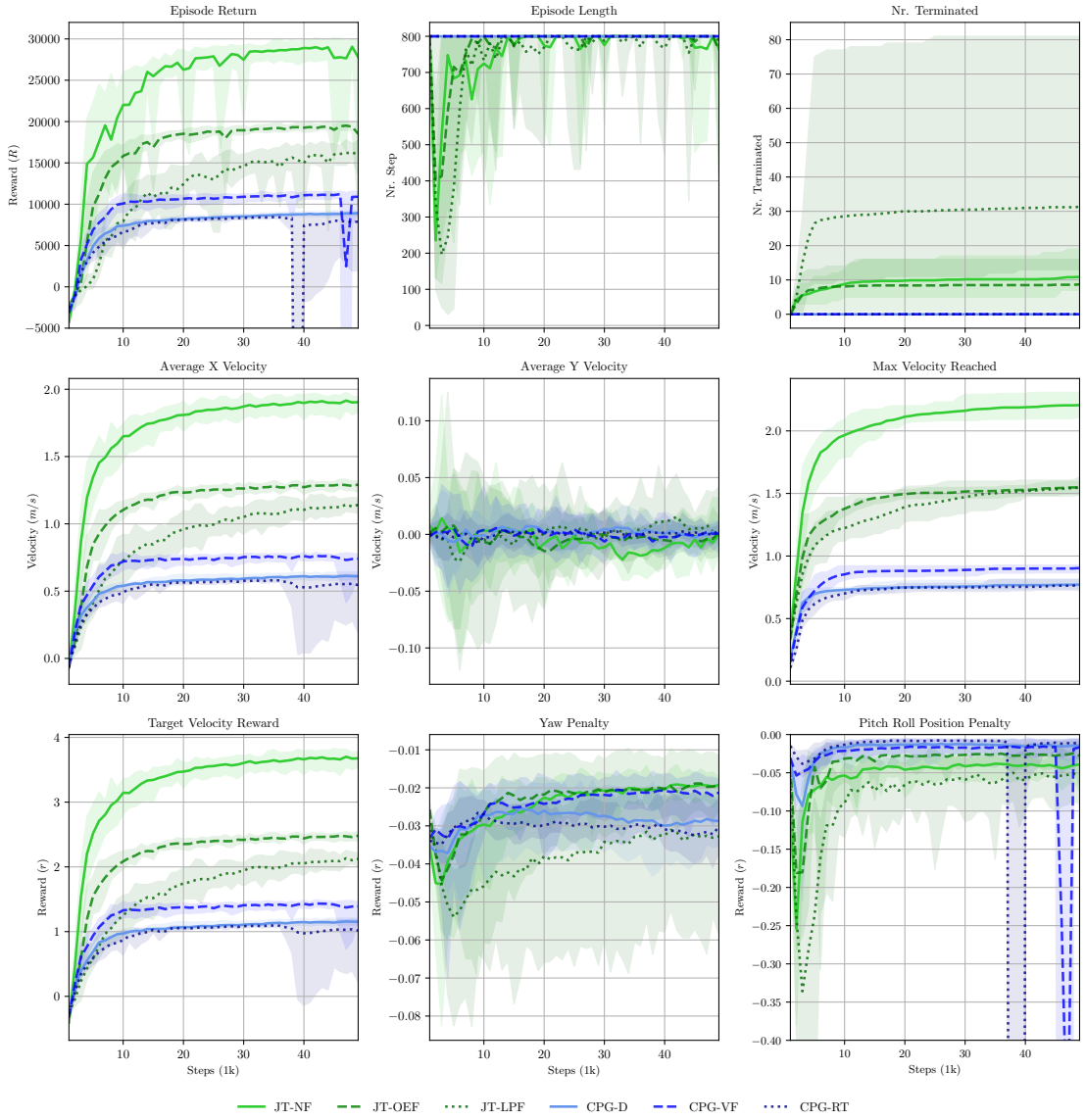


Figure 5.6: Comparison of JT-NF, JT-OEF, JT-LPF, CPG-D, CPG-VF, CPG-RT in the Maximal Velocity experiment. This Forward Locomotion experiment evaluates the ability of each approach to achieve the highest possible velocity.

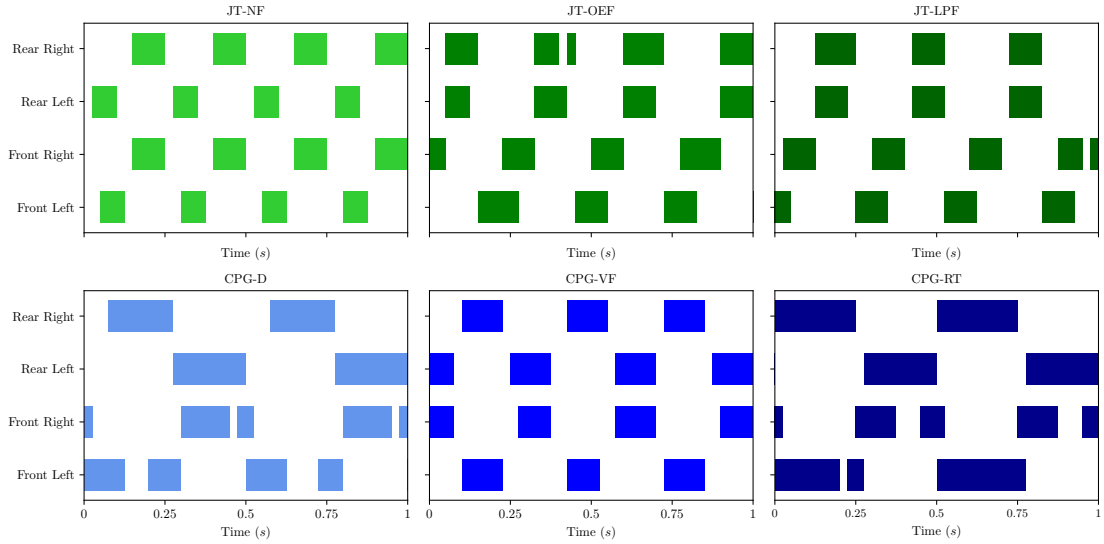


Figure 5.7: Gait evaluation in the Forward Locomotion with Maximal Velocity Experiment. The figure illustrates the contact points of the feet with the ground over a one-second duration.

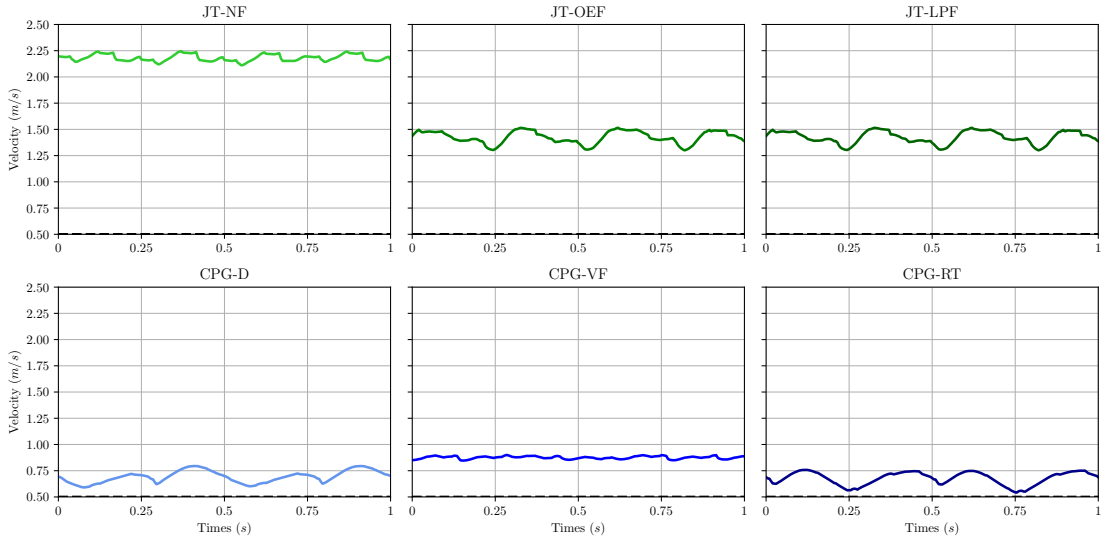


Figure 5.8: Local Trunk Forward Velocity in the Forward Locomotion with Maximal Velocity Experiment.

5.1.4 Omnidirectional Locomotion with Random Target Command

This experiment focuses on teaching the agent to move in all possible horizontal directions, making the task significantly more challenging than the Forward Locomotion experiment. Unlike the forward-only scenario, this task requires conditioning the agent on a task vector that specifies the target direction, greatly increasing complexity. The agent must learn to navigate effectively while minimizing yaw velocity. The command function, which uniformly samples directions from a unit circle, ensures random exploration across all directions, while the Omnidirectional Locomotion Velocity Reward incentivizes movement toward the target direction and discourages lateral and yaw deviations. The experiments are conducted for 50k steps with a target velocity of 0.5 m/s, and evaluation runs are performed every 1000 training steps using the deterministic policy. During evaluation, the agent is tested in four major directions—forward, backward, left, and right—at both the full target velocity (0.5 m/s) and half target velocity (0.25 m/s).

The evaluation results are presented in plots 5.9, 5.10, and 5.11 for each direction, divided into three categories: the velocity along the target direction, referred to as the desired velocity, the velocity perpendicular to the target direction, referred to as the lateral velocity, and the yaw velocity. For the lateral and yaw velocity plots, the target is 0 m/s and 0 rad/s, respectively. These plots provide a detailed performance comparison, allowing the evaluation of both directional accuracy and stability for each approach. The mean absolute error (MAE) between the target and actual velocities is used as a key performance metric: "Velocity Desired MAE" quantifies errors in the target direction, while "Velocity Lateral MAE" captures deviations in the perpendicular direction.

The evaluation encompasses both JT and CPG-approaches, with results highlighting significant differences in performance. The first metric examined is the number of terminations, where Joint-Target approaches perform poorly, averaging approximately 30 terminations. In contrast, all CPG approaches show no terminations, underscoring their superior stability during training. Among the Joint-Target approaches, none achieve meaningful velocity in any direction. Instead, they appear to converge on a standing behavior, failing to make progress toward learning directional locomotion. This suggests that the Joint-Target approaches require more time to learn how to move in multiple directions,

making a curriculum-based approach compelling for step-by-step learning.

In terms of velocity performance, the CPG approaches significantly outperform the Joint-Target methods. CPG-VF achieves the best overall results, reaching the target velocity in forward and backward directions and demonstrating effective modulation of step frequency for different directions. The CPG-D approach follows a similar trend, achieving consistent and stable movement across all directions, with slightly reduced performance compared to CPG-VF. The CPG-RT approach also demonstrates directional locomotion, with slightly reduced effectiveness compared to CPG-VF and CPG-D. However, it exhibits an uneven performance, with better results in forward and backward directions compared to lateral directions. Movements to the left are moderately controlled, but movements to the right show inconsistencies, indicating challenges in adapting to all directions uniformly.

The yaw velocity plots further illustrate these differences. CPG-VF effectively constrains yaw velocity within a range of approximately -0.2 to 0.2, demonstrating stable and controlled turning behavior. In contrast, CPG-D and CPG-RT exhibit wider yaw velocity ranges, reflecting less precise control.

The "Half" plots, which evaluate the agents at half the target velocity ($0.25\,\text{m/s}$), reveal the ability of each approach to interpolate between velocities. The CPG approaches successfully adjust their velocities across all directions, consistently achieving the desired target. Joint-Target approaches, however, struggle to reach the target velocity, failing to demonstrate reliable control. These results further emphasize the effectiveness of the CPG methods in achieving robust omnidirectional locomotion.

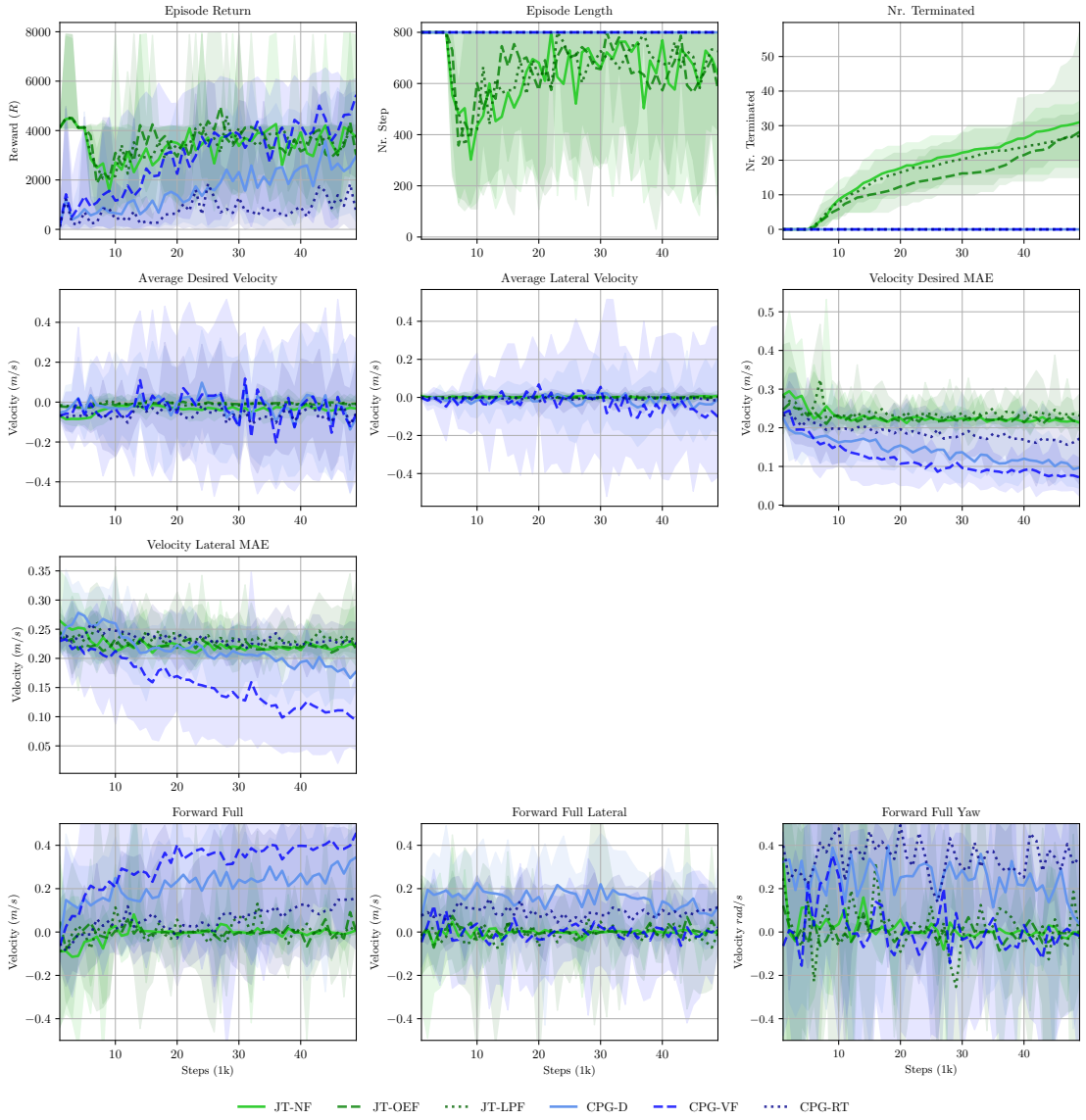


Figure 5.9: Comparison in the Omnidirectional Locomotion with Random Target Command experiment.

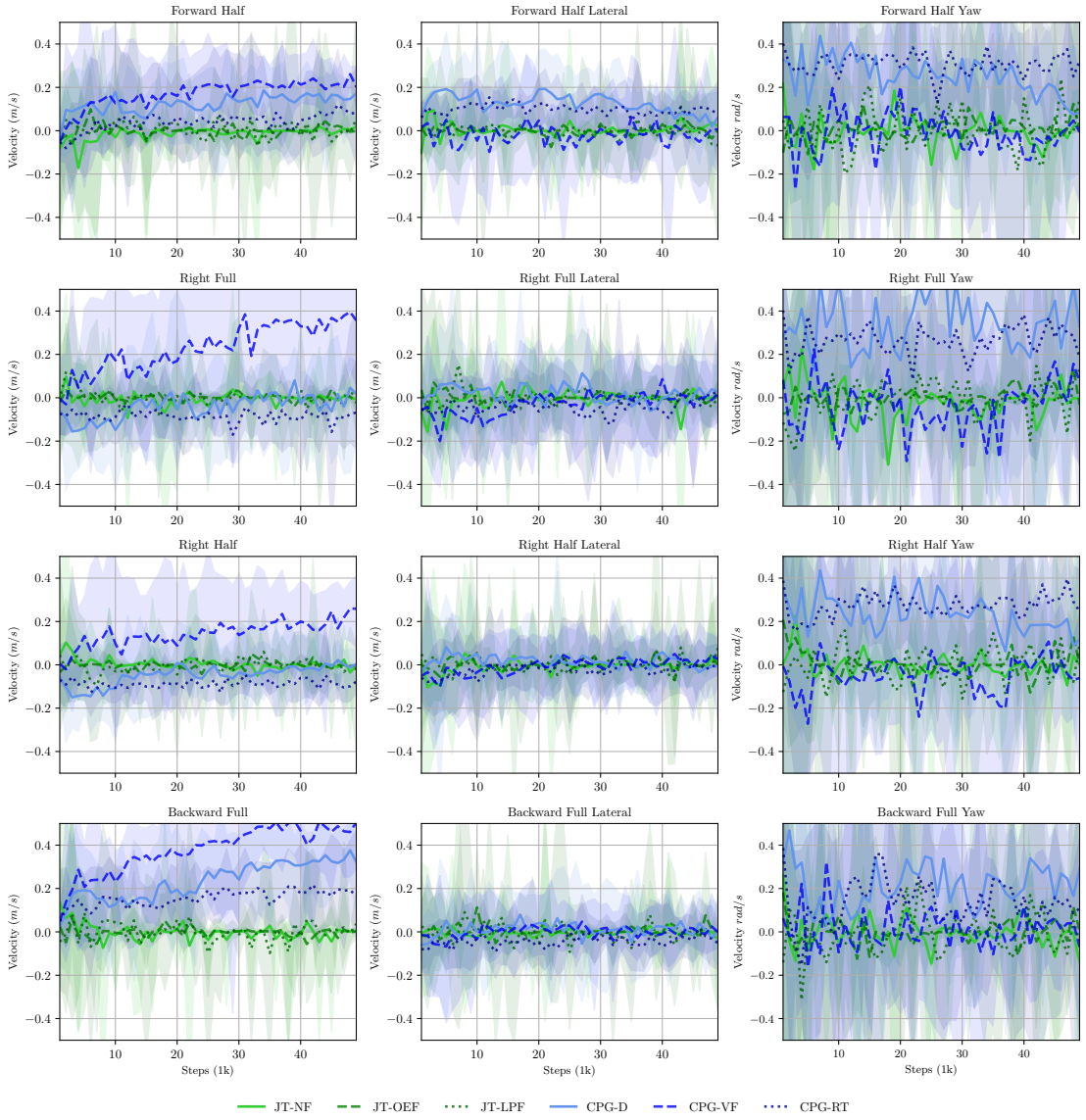


Figure 5.10: The evaluation includes deterministic policy runs conducted every 1000 training steps. These runs test performance across various directions, including forward, backward, left, and right, at both full and half target velocities.

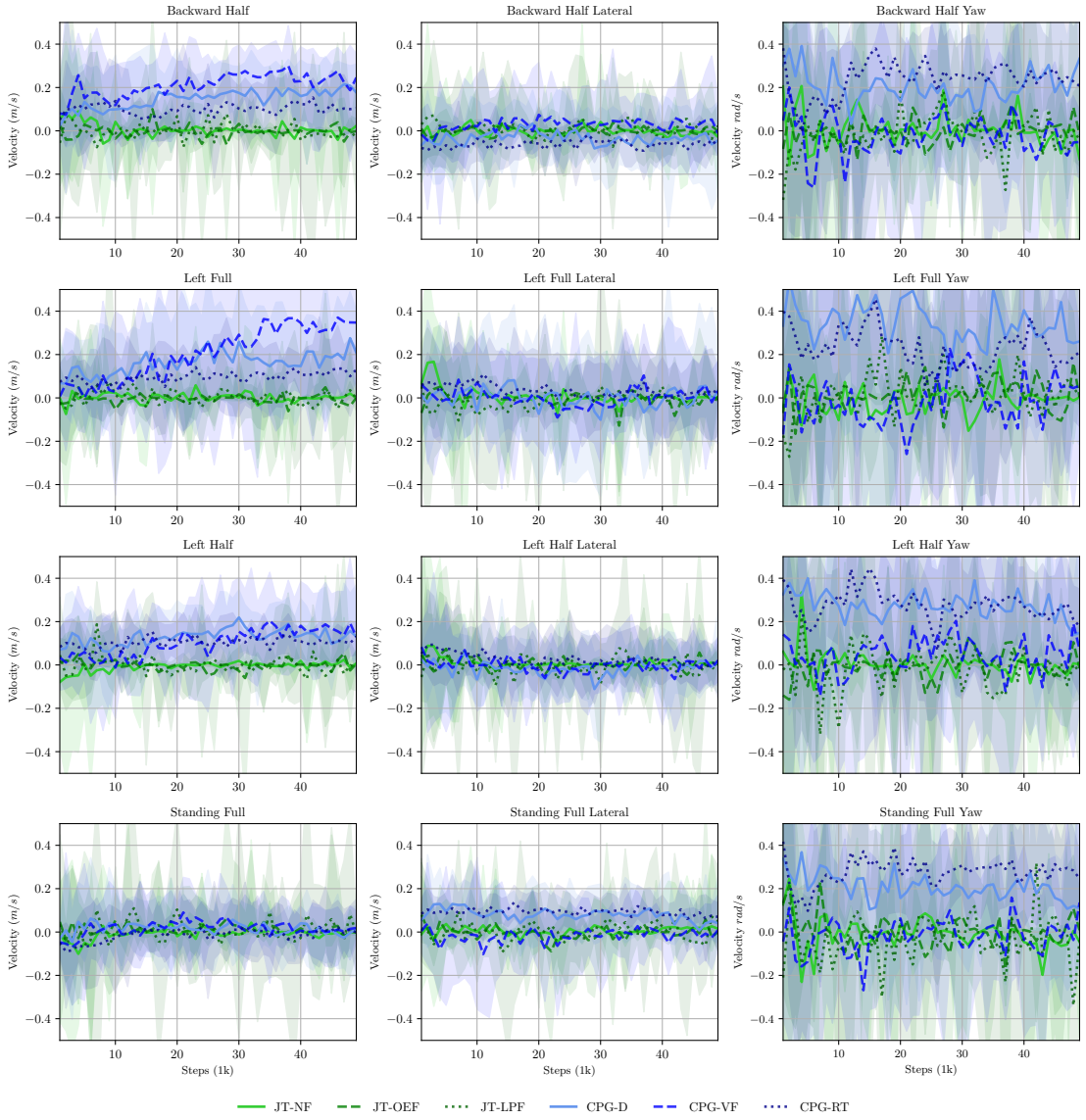


Figure 5.11: The results illustrate the velocity achieved in the desired direction, lateral velocity in the perpendicular direction, and yaw velocity.

5.1.5 Omnidirectional Locomotion with Progressive Directional Expansion Curriculum

This experiment builds upon the previous Omnidirectional Locomotion with Random Target Command by incorporating a structured curriculum, termed the Progressive Directional Expansion Curriculum, to enhance learning efficiency and performance. The goal remains consistent: to enable the agent to navigate freely in all horizontal directions while adhering to specific target velocities. By systematically expanding the agent's range of explored directions, the curriculum seeks to address the inherent challenges of learning omnidirectional locomotion, as outlined in the dedicated curriculum section 4.5.1. Unlike the Random Target sampling in the prior experiment, this approach introduces a progression-based methodology designed to mitigate the limitations of purely stochastic exploration.

A key addition to this experiment is the Curriculum Progress plot, which visually represents the agent's learning trajectory across different directional bins. This plot tracks how much of the directional circle, divided into two half-circles and further partitioned into bins, has been successfully learned. It provides a clear indication of how the curriculum facilitates balanced and systematic learning compared to the Random Target approach. The primary advantage of this curriculum lies in its ability to ensure adequate training in each direction, preventing neglect of specific areas and achieving a baseline performance threshold in all directions before advancing.

One major limitation of the Random Target approach is its inefficiency in uniformly training the agent across all directions. Without a structured curriculum, certain directions, such as lateral or diagonal movements, may remain undertrained or exhibit suboptimal performance. The curriculum addresses this by systematically introducing new directions only after meeting specific performance criteria in the current bins, thereby fostering gradual skill acquisition and minimizing training redundancy.

The analysis focuses exclusively on Central Pattern Generator-based methods, as learning omnidirectional locomotion from scratch remains a significant challenge for Joint Target approaches. In general, the agent lacks sufficient generalization capabilities to mirror motions for opposite directions, necessitating independent learning for each direction. This learning process is time-intensive, making CPG-based methods advantageous. With a predefined gait encoded in the pattern

generator, the neural network is freed to focus on refining control and transforming the encoded gait into precise, task-specific velocities.

In comparing the performance of CPG-based approaches, it is notable that neither approach resulted in falls during training, demonstrating their inherent stability. Among the evaluated methods, CPG-VF achieved the best overall performance, as observed in the previous experiment. The curriculum further enhanced its capabilities, particularly in lateral movements, where it exhibited marked improvements in leftward and rightward walking. Similarly, CPG-D and CPG-RT also showed improvements over their Random Target function counterparts, with better adherence to lateral and yaw velocity constraints, demonstrating the curriculum's effectiveness in refining these behaviors.

While CPG-VF achieved the highest velocity across all directions, CPG-D and CPG-RT displayed more consistent improvements under the curriculum. Specifically, CPG-D showed balanced performance with slightly better leftward motion compared to rightward, while CPG-RT demonstrated improved control in forward and backward movements. Despite its advancements, CPG-RT retained some directional inconsistencies, particularly in lateral directions, suggesting that further refinements may be necessary for uniform performance.

The yaw velocity plots reveal a general improvement across all CPG approaches under the curriculum. CPG-VF maintained a tight control range of approximately -0.2 to 0.2, while CPG-D and CPG-RT exhibited reduced variability compared to their Random Target counterparts. These findings underscore the curriculum's role in enhancing not only directional accuracy but also stability in yaw control.

Although the curriculum improved the agents overall performance compared to the Random Target approach, none of the approaches achieved the target velocity of 0.5 m/s in all directions. This is evident from the Velocity Desired MAE, which remains at 0.05 m/s, reflecting the persistent challenge of achieving perfect velocity tracking in omnidirectional locomotion tasks.

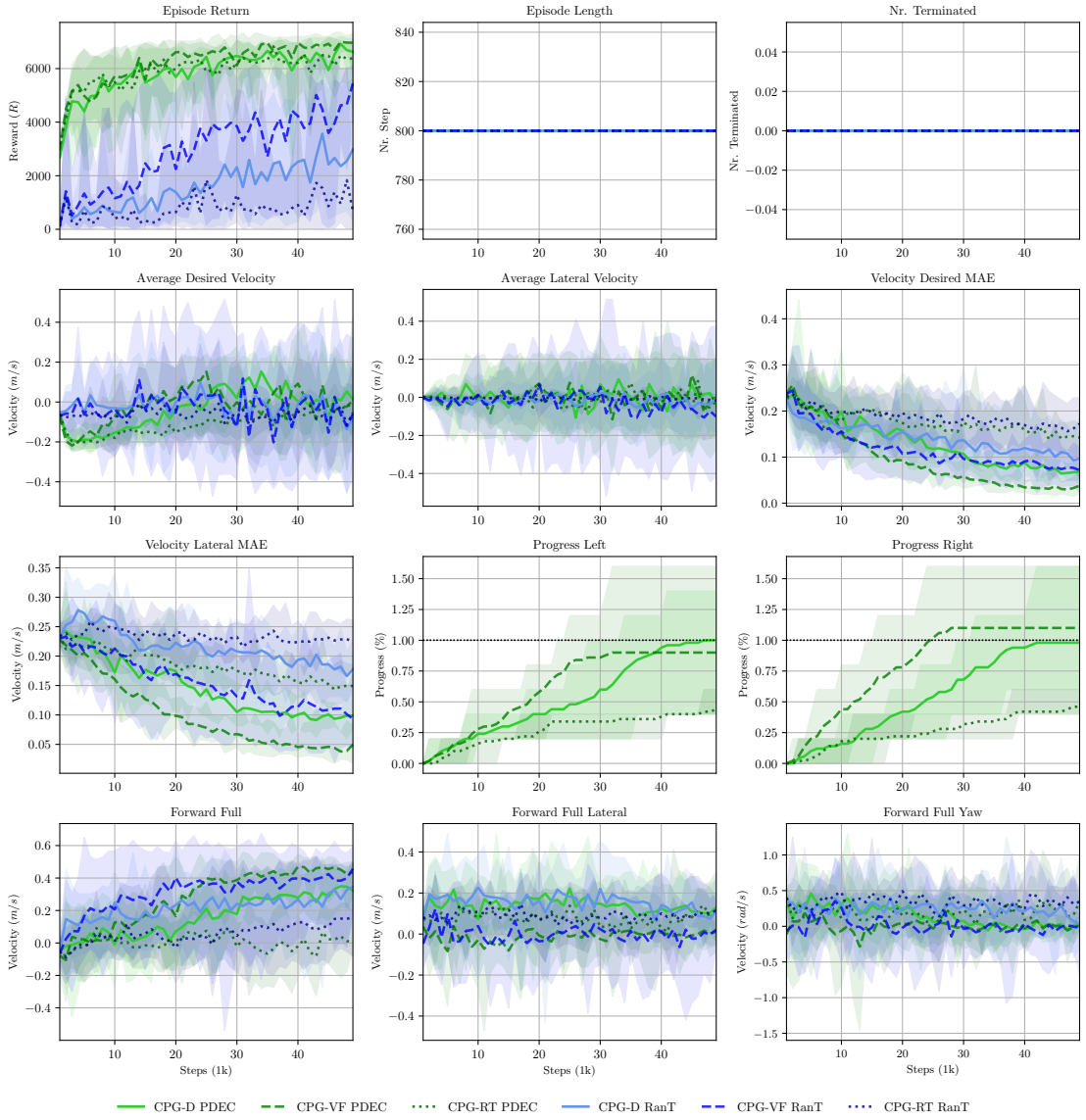


Figure 5.12: Comparison in the Omnidirectional Locomotion with Progressive Directional Expansion Curriculum Experimen (PDEC) with Omnidirectional Locomotion with Random Target (RanT).

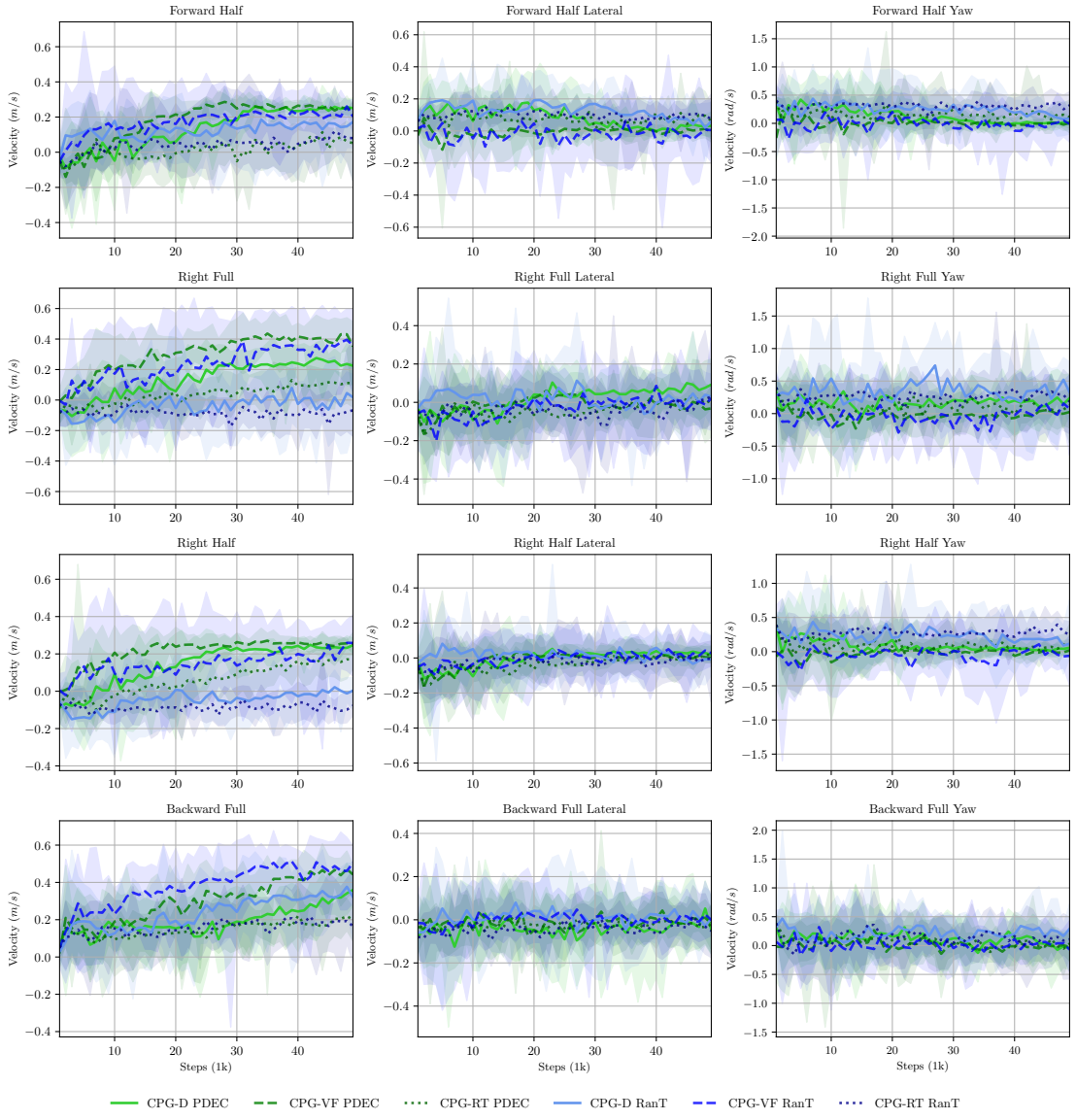


Figure 5.13: The evaluation includes deterministic policy runs conducted every 1000 training steps. These runs test performance across various directions, including forward, backward, left, and right, at both full and half target velocities.

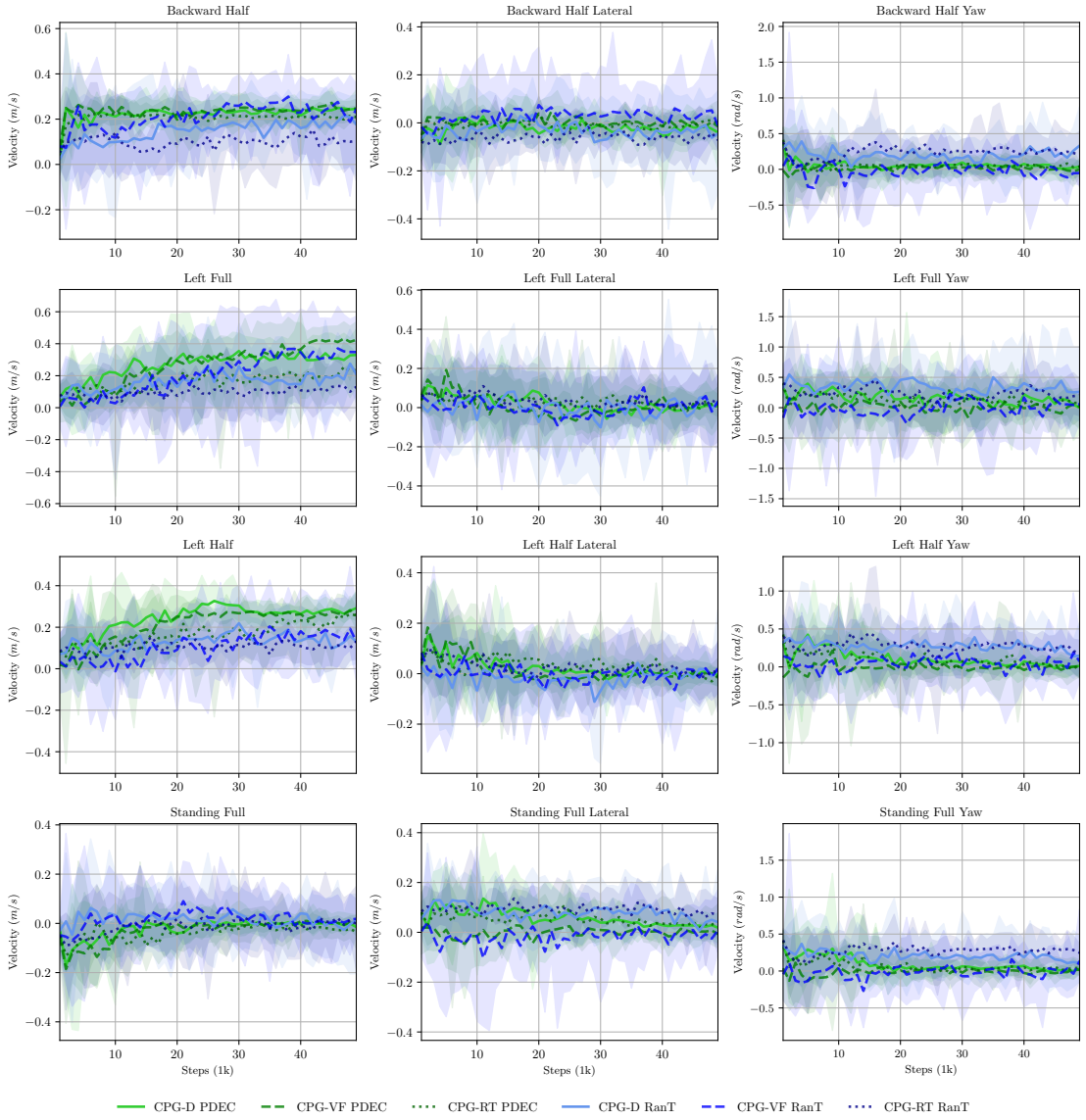


Figure 5.14: The results illustrate the velocity achieved in the desired direction, lateral velocity in the perpendicular direction, and yaw velocity.

5.2 Real World Experiments

Real-world experiments are critical to validate the findings from simulation and assess the applicability of the trained policies under physical constraints. While simulation provides a controlled environment with precise state estimation, efficient exploration, and rapid iteration, transferring these results to the real robot introduces numerous challenges. In simulation, agents often demonstrate excellent sample efficiency, learning effective behaviors within a constrained number of steps. However, reality poses a markedly different scenario, with inherent noise, uncertainties, and wear introducing complexities that can impede the learning process. These factors show the need to evaluate the training process on the physical robot to ensure their robustness and reliability.

In simulation, agents were trained for Forward Locomotion with Target Velocity, Omnidirectional Locomotion with Random Target Command, and curriculum-based approaches such as Omnidirectional Locomotion with Progressive Directional Expansion Curriculum. These experiments explored the agent's ability to move stably in predefined directions, navigate across all directions, and incrementally expand its behavioral repertoire. The real-world experiments aimed to translate these findings by evaluating the agent's ability to perform Forward Locomotion and Omnidirectional Locomotion in two distinct environments: the smooth and constrained office environment and the more complex outdoor environment featuring uneven cobblestones and environmental variability. These experiments bridge the gap between simulation and physical systems by testing the agent's adaptability to different levels of terrain complexity and friction coefficients.

The challenges of real-world experiments extend beyond the physical constraints of the robot itself. Noise in sensor measurements, such as inaccurate state estimation, can degrade the observation and reward signals used during training. Wear and tear on hardware components, such as the robot's feet, introduce additional uncertainty over time, while fluctuations in battery levels affect actuator responsiveness. The outdoor environment adds further complications, such as wind disturbances and a three degree terrain inclination. Safety measures were employed to protect the robot, including automatic disengagement of training when orientation thresholds were exceeded or joint limits were violated.

In such cases, the P-gains were set to zero, and the D-gains gains were set to three to minimize the impact forces.

Internal velocity estimation, a critical factor for training, was found to systematically underestimate the robot's actual velocity, introducing inaccuracies in both observation and reward signals. This limitation made curriculum-based learning approaches infeasible, as the transition criteria between curriculum levels depend on accurate performance assessment. Consequently, real-world experiments were limited to testing Forward Locomotion and Omnidirectional Locomotion without the curriculum.

Real-world experiments excluded two approaches—Joint Target Prediction without filtering (JT-NF) and Central Pattern Generator with Residual Targets (CPG-RT), due to poor performance during real-world testing. These approaches exhibited over 100 falls within the first 20k steps, leading to excessive hardware wear and safety risks. Testing focused on more robust approaches to ensure meaningful evaluation without risking significant damage to the robot.

The Forward Locomotion experiments were conducted in both the office and outdoor environments to evaluate performance across terrains of varying complexity. Omnidirectional locomotion experiments were conducted exclusively outdoors, as the limited space in the office environment restricted the agent's ability to explore multiple directions effectively. This decision ensured a comprehensive evaluation of the agent's capacity to navigate complex, unstructured environments.

The experiments were conducted using the MAB Robotics HoneyBadger 4.0 robot, depicted in figure 4.1 [70], connected via Wi-Fi to a MacBook Pro 2023 equipped with an M3 Pro processor (12-core CPU and 18-core GPU) and 18 GB of unified RAM. The computations were accelerated using RL-X [71] and JAX [72] with jax-metal version 0.1.0. Communication was implemented using ROS 2 Humble [73] with RoboStack [74], and a custom remote interface, developed in React and utilizing WebSockets, enabled real-time system monitoring and asynchronous control from various devices. These tools and methods facilitated efficient experimentation while maintaining robust communication between the robot and the control system.

5.2.1 Forward Locomotion in Office Environment

This experiment evaluates the real-world applicability of the findings from the Forward Locomotion with Target Velocity experiment conducted in simulation. The primary goal is to train the robot to walk forward with a target velocity of 0.25 m/s, using the same reward and command functions as in simulation. The experiments are conducted for 20k steps to assess how quickly and effectively the approaches can learn to achieve stable locomotion. The experimental setup was constrained by the physical limitations of the office environment. The available space was restricted, requiring frequent manual relocation of the robot, which significantly increased the overall experimental runtime. Additionally, the floor surface in the office was smooth, which reduced the robot's struggle to lift its legs during motion but introduced other challenges related to stability. For the CrossQ-based methods, Joint Target Prediction approaches utilizing either a one Euro filter or a low-pass filter were evaluated. However, the Joint Target Prediction approach without any filtering was excluded due to its instability and high risk of damaging the robot, as it led to frequent falls during preliminary tests. Similarly, the Central Pattern Generator with residual targets was omitted because it caused over 100 terminations within the first 10k steps, rendering it unsuitable for real-world evaluation. A significant challenge during these experiments was the inaccuracy of the robot's state estimation. The internal velocity estimation substantially underestimated the actual velocity, leading to an observation signal that differed markedly from the real-world dynamics. Consequently, the perceived velocity only showed slight improvements during training, even as the robot's actual performance improved. This discrepancy caused the cumulative reward to be

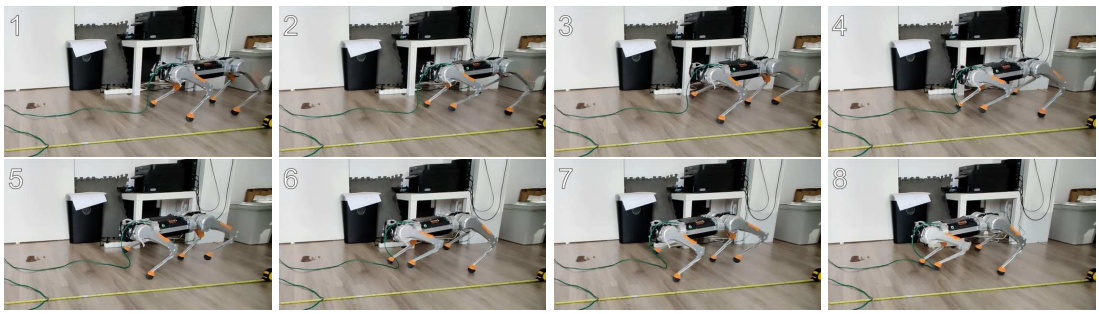


Figure 5.15: Evaluation run of internal MPC controller showing a stable trot gait.

dominated by episode length rather than velocity tracking, making it difficult to evaluate the effectiveness of different approaches purely based on reward metrics. Figure 5.15 illustrates a run of the internal MPC controller, demonstrating its use of a trot gait for forward locomotion. The controller effectively navigates the uneven terrain without falling or generating any significant yaw velocity.

SAC UTD 20 JT-LPF The experiment with SAC UTD 20 and low-pass filter took more than 40 minutes to complete. There are two reasons for this: this experiment was done with a reduced frequency of 20 Hz and therefore it took more time to collect the same amount of steps. Secondly, the Number of Terminations ended up to be 43, which is comparably a high number. One reason for this is that the safety measures were prematurely triggered a number of times. The strategy of the agent was to heavily jump with the front legs. This resulted in a harsh movement and the robot was not able to grip the ground properly. The robot was not able to learn a proper gait and the velocity was not improving over time. Also, the evaluation run showed (depicted in figure 5.16) that the agent was not able to walk in a straight line. Despite the jumping while learning, it shows signs of a trot gait and has connected the front left and rear right leg multiple times but is brushing the ground while the other legs are used for stabilization. Resulting in a very low forward velocity, travelling about half a meter in 40 seconds. Additionally, it has very low control over yaw and lateral velocity.

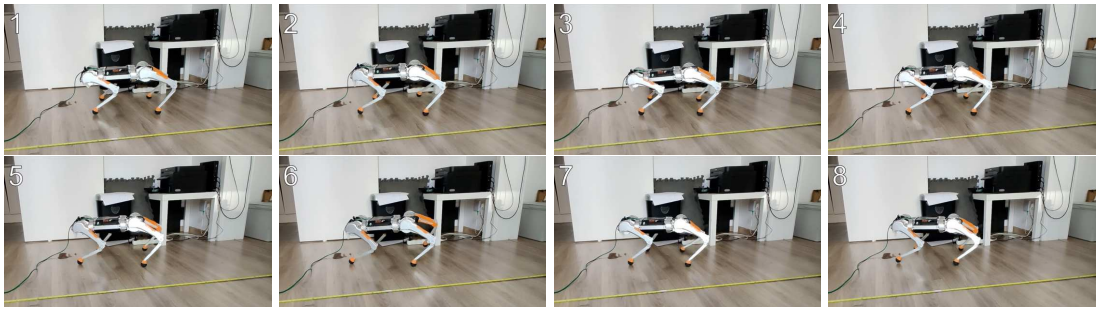


Figure 5.16: Evaluation run of SAC UTD 20 with low-pass filter, staying on the spot for the duration of 10 seconds.

CrossQ JT-OEF This experiment needed 25 minutes to conclude the required samples. Within the first three minutes, the agent was able to make little steps while keeping balance. This led to more exploration and consecutive falling. After 4.5 minutes, the agent was able to take bigger consecutive steps without falling directly at a low speed. This was followed by increasing the steps and lifting the leg from the floor to optimize the needed torque. At the same time the agent picked up that the velocity can be reduced to prevent falling down. Additionally, it also learned at this stage that a trot gait gives good balance while improving velocity. Moreover, the agent has already learned a higher walking speed than the given target velocity. After 6 minutes it is still increasing the velocity until it peaks at about 12:30 minutes. The agent develops a strategy of falling backwards when it is threatened to terminate. After this, the agent tries to improve robustness and yaw velocity restriction. In the end, it also learns to recover from falls by catching itself, but this only works when the agent tips in a shallow angle and one of its front feet are placed in the front of the robot. The evaluation run (see figure 5.17) showed that the agent capable of performing a stable trot gait and is able to reach up to 0.85 m/s for short distances of 1 m. This is due to the fact, that the agent is then falling to the back to prevent falling. For longer distances, the agent is able to reach a velocity of about 0.65 m/s. The agent struggles to walk in a completely straight line and occasionally changes its direction but learns a trot gait that lifts the leg. Despite the fact that the agent was trained for the target velocity of 0.25 m/s, it has exceeded this velocity. This is due to the fact that the internal velocity estimation is bounded, but the agent picks up that a higher velocity results in a higher reward and therefore intuitively aims for higher velocities even though

Figure 5.17: Evaluation run of CrossQ JT-OEF showing a trot gait and achieving a velocity of 0.65 m/s and 0.85 m/s for short distances.

these higher velocities only lead to minor reward increases. Then the learned policy was tested in simulation, it was possible to achieve velocities up to 1.5 m/s.

CrossQ JT-LP The experiment was completed in approximately 20 minutes, during which the agent exhibited a maximum velocity of less than 0.4 m/s. Initially, for the first two minutes, the agent displayed no forward movement. During this time, it performed very small, rapid steps that provided negligible progress. These steps were slightly larger than jittering movements but still resulted in minimal forward progress. The application of a low-pass filter to the action outputs aimed to suppress jittering and excessively small movements, yet the agent exploited the residual movement allowance to generate these tiny steps.

Around the five-minute mark, the agent began to demonstrate more assertive movements, combining small vibrating motions with occasional large, inconsistent steps. These larger steps introduced a jerky forward progression, with frequent foot dragging on the ground. To stabilize itself against forward imbalance, the agent often adopted a rapid backward-leaning motion, which sometimes led to larger, sporadic steps.

From the eleventh minute onward, the agent transitioned to a gait dominated by medium forward jumps, engaging all four legs simultaneously. This strategy improved speed and alignment, allowing the agent to sustain faster movement. In simulation, the policy was able to reach a velocity of 0.65 m/s.

The evaluation run following training, depicted in figure 5.18, showcased the resulting gait. The agent adopted a common strategy of executing medium

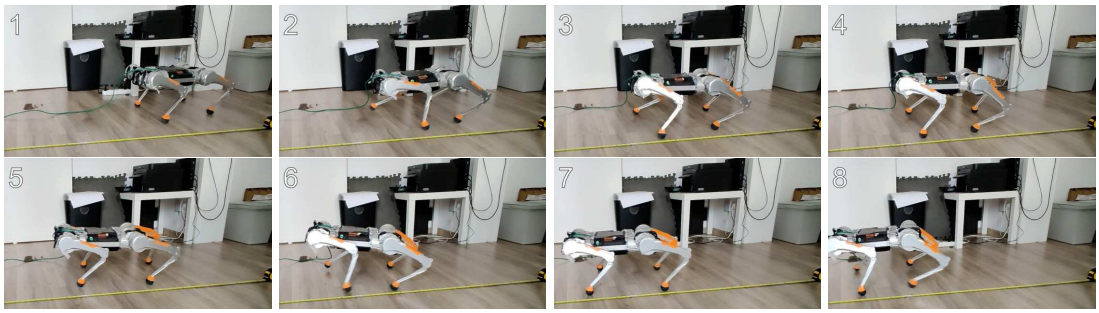


Figure 5.18: Evaluation run of CrossQ JT-LPF achieving a velocity of 0.4 m/s.

forward jumps with synchronized front and rear legs. While this gait demonstrated improved coordination compared to earlier phases of training, the achieved velocities remained modest, highlighting the constraints imposed by real-world dynamics and the filtering approach.

CrossQ CPG-D The training process for this experiment lasted approximately 30 minutes, with a total of 38 terminations recorded. These terminations were primarily caused by the safety measures, which frequently triggered before the robot could fall. This early intervention were partially caused by the swing height being set to $0.2\,\mathrm{m}$ with an additional possible z-offset of up to $0.1\,\mathrm{m}$. These parameters were configured to accommodate very rough terrain and maintained consistently across all runs for comparability. While this setup ensured the safety of the robot, reducing these parameters could potentially lead to better performance.

During the initial phase of training, the agent focused on exploring the boundaries of the safety measures and optimizing the trunk's tilting angle to prevent premature termination. This was followed by the agent quickly learning to take forward steps. Throughout the training process, the agent consistently tested the limits of the safety parameters, seeking to refine its gait and achieve efficient forward movement. Concurrently, it developed the ability to maintain a level trunk despite the elevated swing height, demonstrating robust compensation strategies.

The evaluation run, illustrated in figure 5.19, revealed that this approach was capable of walking forward with a consistent speed without falling. However, yaw velocity remained an unresolved issue, indicating room for improvement in

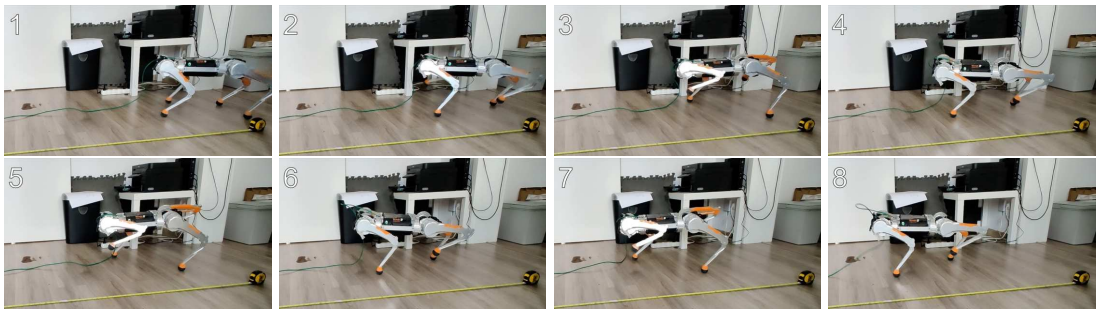


Figure 5.19: Evaluation run of CrossQ CPG-D achieving a velocity of 0.3 m/s.

directional stability. The evaluation run measured a maximum velocity of $0.3\,\mathrm{m/s}$, while the agent achieved approximately $0.6\,\mathrm{m/s}$ in simulation.

CrossQ CPG-VF It took about 45 minutes to complete. Safety measures were triggered prematurely, partly due to the agent's control over its gait frequency. This flexibility resulted in an initially unstable learning phase, as the agent had to identify the boundaries of the safety constraints. Adjusting the swing height parameter could potentially improve stability.

In the first 20 minutes, the agent primarily focused on learning how to avoid activating the safety measures. By the 25-minute mark, it was able to move forward slowly, though frequent falls still occurred. Following this, the agent began exploring faster walking patterns. Around 32 minutes, it reduced the number of falls, increased its speed, and adjusted its gait frequency.

The agent initially discovered that a very high frequency provided more stability. Over time, it learned to walk faster at a lower frequency but would revert to a higher frequency in situations with a high risk of falling. By the end of the evaluation (see figure 5.20), the agent had learned to walk consistently without falling. The evaluation runs also showed good yaw control throughout. The evaluation achieved a velocity of approximately 0.3 m/s, both in real and simulated runs.

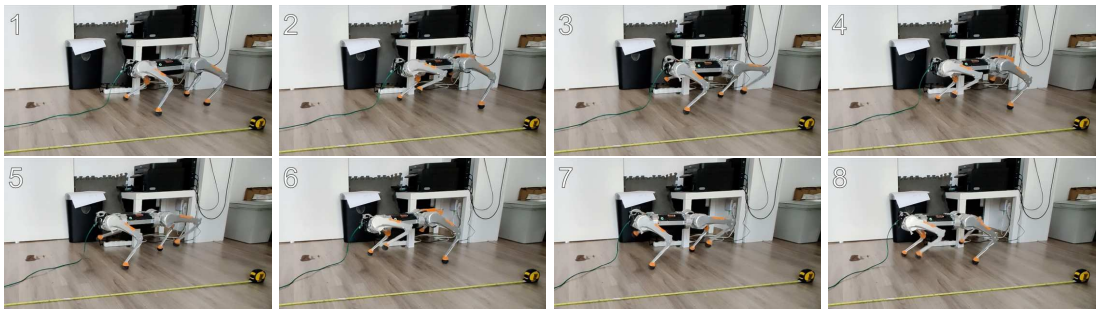


Figure 5.20: Evaluation run of CrossQ CPG-VF achieving a velocity of 0.3 m/s.

Conclusion The Forward Locomotion in Office Environment experiment revealed the challenges associated with real-world testing in constrained indoor environments, including limited space and smooth, low-friction surfaces. The comparison of the key metrics can be found in table 5.3 and a more detailed view of the training process is given by figure 5.21. These factors necessitated frequent manual intervention to relocate the robot, prolonging the experimental duration. Among the tested approaches, CrossQ JT-LPF exhibited modest performance, achieving forward velocities of less than $0.4\,\text{m/s}$. The agent adopted an inconsistent gait, characterized by small, rapid steps and occasional forward dashes, but struggled to achieve sustained or stable movement. CrossQ JT-OEF, on the other hand, demonstrated the most dynamic performance, achieving a peak velocity of $0.85\,\text{m/s}$ during evaluation runs, albeit with occasional backward falls and slight deviations from a straight trajectory. This approach showcased the potential for on-robot learning despite the limitations of manual resets and inaccurate state estimation. For the CPG-based methods, CrossQ CPG-D showed the most robust behavior, achieving consistent forward movement and maintaining a level trunk throughout the evaluation. However, its maximum velocity of $0.3\,\text{m/s}$ fell short of expectations. CrossQ CPG-VF demonstrated stability but relied heavily on high-frequency movements during training, leading to a slower progression and a final velocity comparable to that of CrossQ CPG-D.

Approach	Eval. Speed (m/s)	Yaw Control	Nr. Falls	Duration (min)
SAC UTD 20 JT-LPF	0.0125	poor	43	40
CrossQ JT-OEF	0.85	medium	15	25
CrossQ JT-LP	0.4	medium	16	20
CrossQ CPG-D	0.3	good	38	30
CrossQ CPG-VF	0.3	very good	52	45

Table 5.3: Comparison of the Forward Locomotion in Office Environment experiment. Velocity and yaw control are based on the agent's performance during the evaluation run. Yaw control: **Very Good** (<$10°$), **Good** ($10° - 30°$), **Medium** ($30° - 60°$), and **Poor** (> $60°$). The number of falls and duration indicate the agent's learning progress and stability.

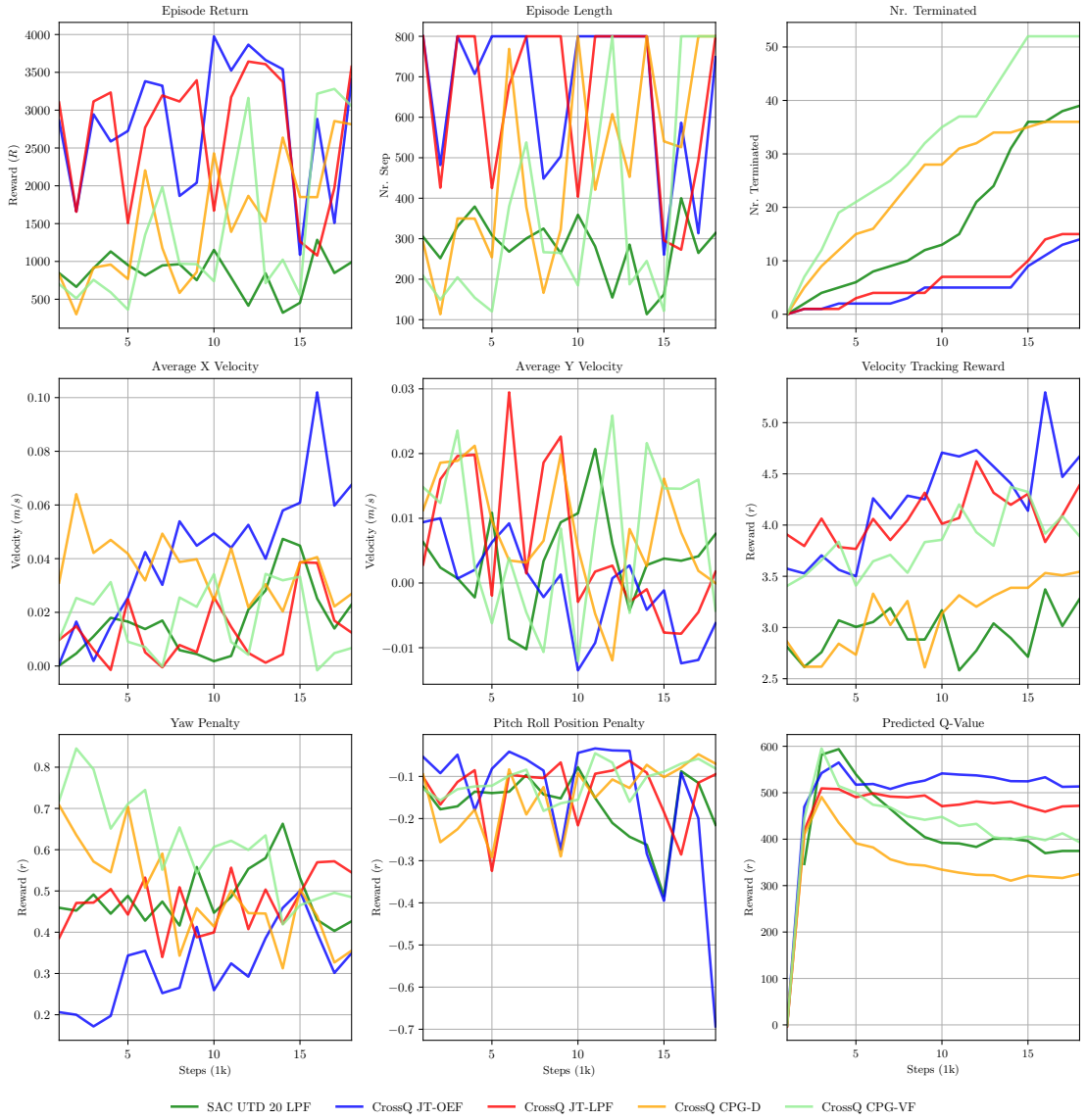


Figure 5.21: Training progress of the Forward Locomotion in the office environment. Due to the wrong velocity estimation, the velocity tracking reward is an insufficient indicator of the performance.

5.2.2 Forward Locomotion in Outdoor Environment

This experiment extends the findings from the Forward Locomotion with Target Velocity simulation experiment and the Forward Locomotion in Office Environment real-world experiment to a more challenging outdoor setting. The primary objective remains consistent: training the robot to walk forward with a target velocity of $0.25\,\mathrm{m/s}$ using the same reward and command functions as employed in simulation. The experiments are conducted for 20k steps, aiming to evaluate how effectively the approaches can learn stable locomotion under more demanding real-world conditions.

The outdoor environment posed additional challenges compared to the office setting. The surface consisted of uneven cobblestones with varying friction levels and a slight inclination, introducing more complex dynamics for the robot to adapt to. However, the increased training area reduced the need for frequent manual relocation of the robot, thereby accelerating the overall training process.

Joint Target Prediction approaches using either a one Euro filter or a low-pass filter were tested. As in the office environment experiments, the Joint Target Prediction approach without filtering and CPG with residual targets were excluded due to instability and high risk of damage, as it led to over 100 terminations within the first 10k steps in earlier tests, making it unsuitable for real-world deployment.

A key challenge in these experiments was the inaccuracy of the internal state estimation system. The velocity estimation consistently underestimated the robot's actual velocity, leading to observation signals that poorly reflected the

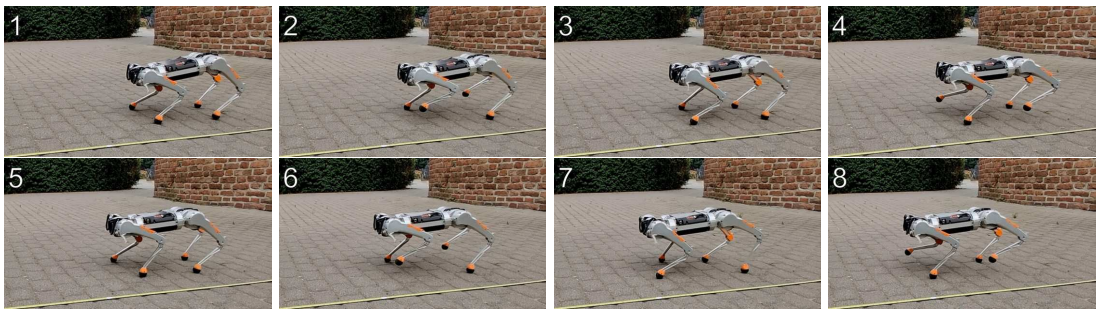


Figure 5.22: Evaluation run of MPC controller.

real-world dynamics. As a result, perceived velocity showed only marginal improvements during training, despite actual performance gains. Consequently, the cumulative reward was dominated by episode length rather than velocity tracking, complicating the evaluation of approach effectiveness based solely on reward metrics.

In figure 5.22 a run of the internal MPC controller can be seen as it is using a trot gait to walk forward. It is able to negotiate the uneven terrain without falling over and exerting any yaw velocity.

CrossQ JT-OEF The training for this experiment concluded in under 19 minutes, including manual resets and 19 total falls. After approximately 1 minute and 30 seconds of training, the agent initiated its first forward steps. By the 2:30-minute mark, it had developed a balancing mechanism by deliberately falling backward to avoid tipping forward, an adaptive strategy in response to the uneven terrain.

The training took place on challenging terrain characterized by varying surface heights and high friction. These conditions required the agent to continually adapt its movement, resulting in frequent falls as it learned to balance and traverse the surface effectively. The uneven ground introduced significant disturbances, demanding continuous adjustments in the agent's gait and balance strategies.

Throughout the training, the agent focused on exploring strategies to improve walking speed and stability. Despite the challenging environment, it demonstrated progressive improvements in both aspects. By the end of the training process (see figure 5.23), the agent had developed a distinctive gait pattern. This gait involved

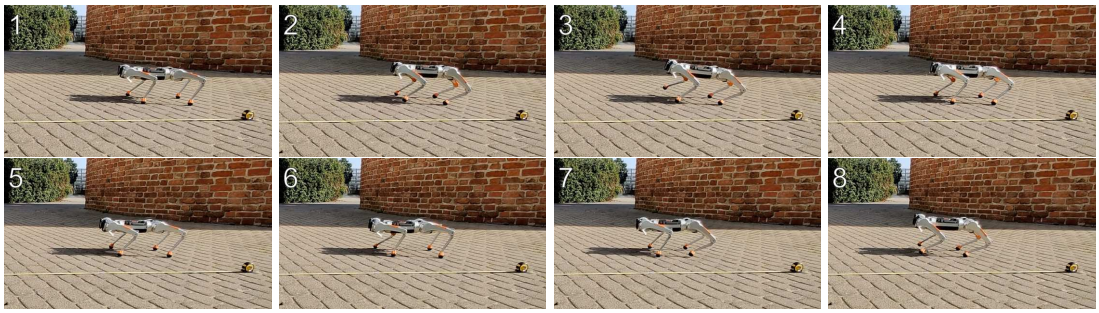


Figure 5.23: Evaluation run of CrossQ JT-OEF achieving a velocity of 0.25 m/s.

alternating steps with the front legs or simultaneously pushing with both front legs while pulling with both rear legs. This combination of gaits allowed the agent to maintain balance and navigate the terrain efficiently.

In the evaluation phase, the agent achieved a walking speed of approximately 0.25 m/s. The evaluation run showcased the learned gait, highlighting the agent's ability to adapt to the challenging terrain conditions while maintaining consistent forward locomotion.

CrossQ JT-LPF The training session for this experiment was completed in about 14 minutes and with 13 terminations. After roughly 5 minutes and 30 seconds, the agent began to take its first steps. During early exploration, it adopted a balancing technique where it periodically allowed itself to fall backward, which proved effective in preventing forward tumbles.

This training aimed to enhance the agent's walking speed on terrain with high friction and irregular surface heights. The constantly changing ground levels demanded frequent adaptive responses from the agent, leading to a continual adjustment of its gait to maintain balance.

As training progressed, the agent developed a distinctive gait optimized for stability on the challenging terrain. By the end (see figure 5.24), it had settled into a rhythm of small, controlled jumps, coordinating all four legs together while occasionally allowing itself to fall backward. This pattern provided a stable forward movement by countering the uneven surface and high friction with calculated adjustments.

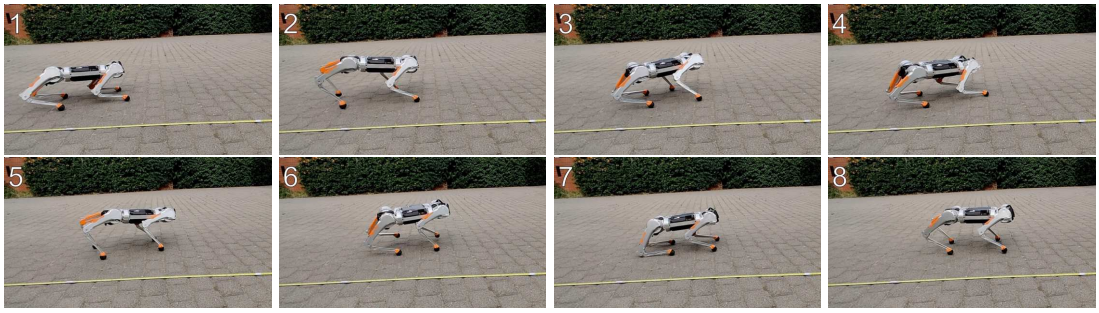


Figure 5.24: Evaluation run of CrossQ JT-LPF achieving a velocity of 0.23 m/s.

In the evaluation phase, the agent achieved a steady walking speed of around $0.23\,\mathrm{m/s}$. This pace reflects its approach of prioritizing balance over speed, as it carefully adjusted to the terrain demands while preserving stability.

CrossQ CPG-D The training session was completed in roughly 17 minutes, during which the agent experienced a total of 39 terminations, including from overly sensitive safety measures. Within the first minute, the agent managed to take its initial steps. Early in the session, it learned to avoid triggering unnecessary safety measures and began working on balancing its trunk to prevent excessive tipping while advancing forward.

After approximately 8 minutes, the agent demonstrated the ability to recover from falls by using its limbs to stabilize itself. However, the agent faced ongoing challenges with steeper sections of the terrain, often leading it to fall backward or even move in reverse as it attempted to navigate these areas.

In the evaluation phase, the agent achieved a walking speed of approximately $0.33\,\mathrm{m/s}$. It maintained a level trunk position throughout its movement, demonstrating that the agent successfully overcame the uneven terrain, leveraging the robustness of the CPG to navigate the challenging environment effectively. The evaluation gait, as illustrated in figure 5.25, highlights the generated pattern of CPG.

CrossQ CPG-VF The training concluded in around 19 minutes, during which the agent encountered a total of 81 terminations, including the ones from false-

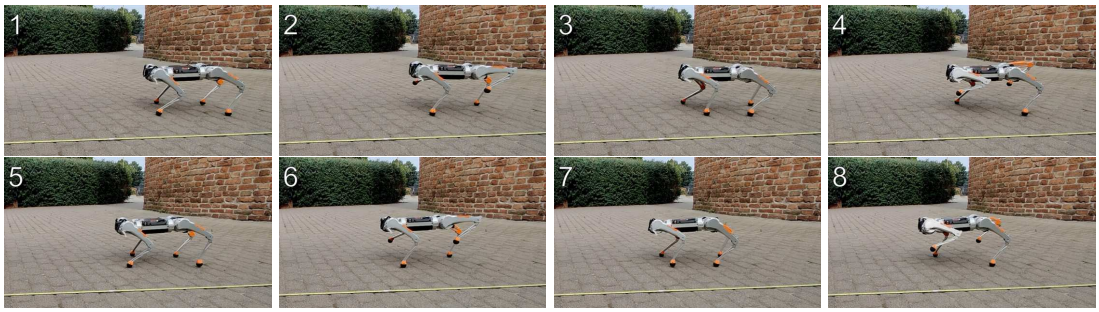


Figure 5.25: Evaluation of CrossQ CPG-D achieving a velocity of 0.33 m/s.

positive triggers of safety measures. The variable frequency of its movement proved challenging at the start, making it difficult for the agent to avoid these unintended terminations. Initially, much of the agent's learning involved adapting to avoid triggering these safety constraints, a skill it only fully developed by the end of the session.

After roughly 5 minutes of training, the agent began to move forward more consistently, though it still encountered challenges with safety measure activation and tipping over. The agent also struggled to overcome steeper terrain sections, often falling backward or even reversing direction when the slope proved too challenging.

Despite the difficulties, the agent learned to adapt its movement frequency during training to its advantage, alternating between slower movements when stable and faster movements when at risk of falling. However, in the evaluation run, that can be seen in figure 5.26, the agent adopted a different strategy, relying solely on high-frequency movements, which led it to rotate in place rather than moving forward.

This outcome contrasts with the results observed in the Forward Locomotion in Office Environment experiment. In that scenario, the same approach demonstrated a good walking ability, achieving consistent forward locomotion. This discrepancy highlights the increased complexity and instability posed by outdoor environments, which introduced uneven terrain and additional challenges that hindered effective gait application during evaluation.

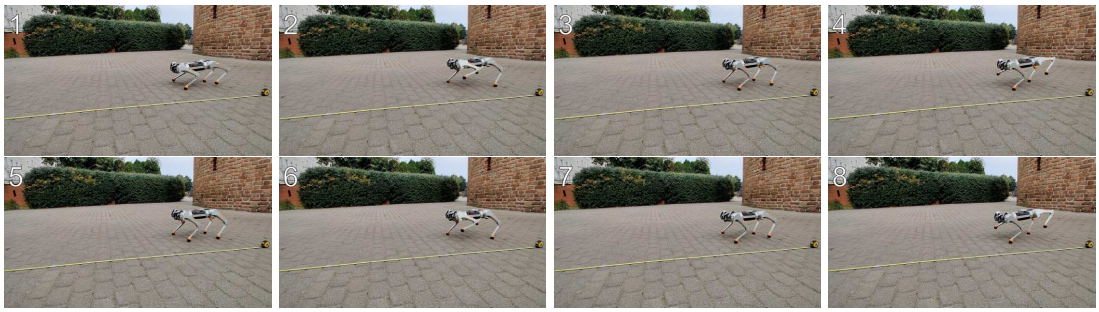


Figure 5.26: Evaluation run of CrossQ CPG-VF remaining on the spot.

Conclusion The Forward Locomotion in Outdoor Environment experiment highlights the increased complexity of real-world outdoor conditions, such as uneven cobblestone terrain, high friction, and inclinations. A summary of the primary metrics is provided in table 5.4, while figure 5.27 offers a detailed visualization of the training progression. Compared to the Forward Locomotion in Office Environment experiment, these factors posed significant challenges for all tested approaches.

The CrossQ JT-LPF approach achieved modest forward velocities, prioritizing balance over speed, while CrossQ JT-OEF demonstrated a more dynamic gait with higher speeds but frequent falls during training. Among the CPG-based methods, CrossQ CPG-D showed the best performance, maintaining a stable trunk and achieving the highest forward velocity, effectively adapting to the uneven terrain. In contrast, CrossQ CPG-VF struggled during evaluation, relying on high-frequency movements that caused it to rotate in place rather than progress forward.

Overall, the results highlight the challenges of transferring learned behaviors to demanding outdoor environments and the need for enhanced adaptability in real-world conditions.

Approach	Eval. Speed (m/s)	Yaw Control	Nr. Falls	Duration (min)
CrossQ JT-OEF	0.25	good	19	19
CrossQ JT-LP	0.23	medium	13	14
CrossQ CPG-D	0.33	very good	39	17
CrossQ CPG-VF	N/A	poor	81	19

Table 5.4: Comparison of the Forward Locomotion in Outdoor Environment experiment. Velocity and yaw control are evaluated based on the agent's performance during the evaluation run. Yaw control categorizes the robot's ability to maintain directional stability: **Very Good** indicates minimal deviations (<$10°$), **Good** allows moderate drift ($10° - 30°$), **Medium** reflects noticeable drift ($30° - 60°$), and **Poor** shows significant instability (> $60°$). **N/A** applies when yaw control is not evaluated. The number of falls and duration indicate the agent's learning progress and stability.

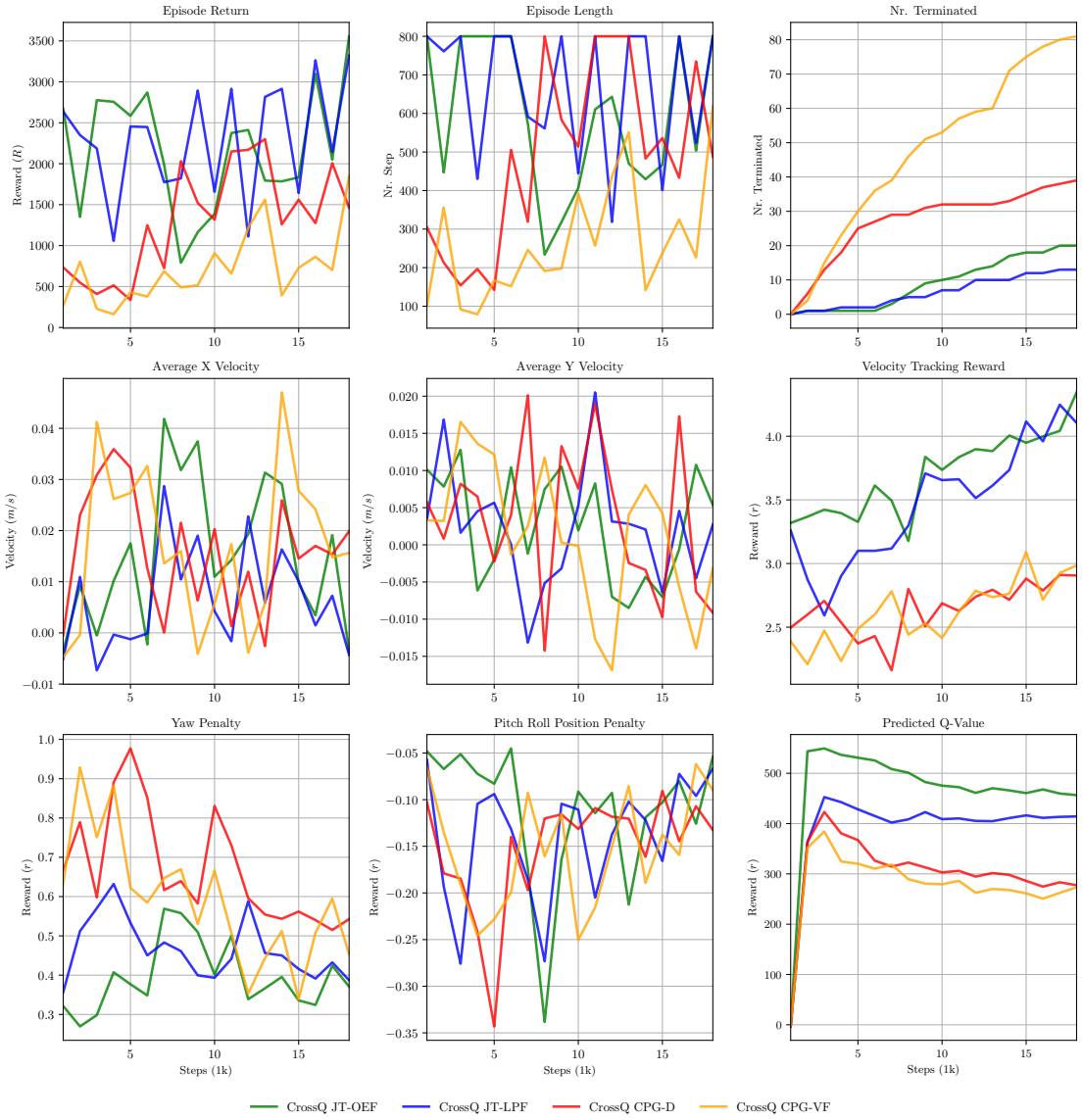


Figure 5.27: Training progress of Forward Locomotion in the Outdoor Environment. Due to the incorrect velocity estimation, the velocity tracking reward is an insufficient indicator of the performance.

5.2.3 Omnidirectional Locomotion in Outdoor Environment

This experiment evaluates the real-world applicability of the findings from the Omnidirectional Locomotion with Random Target Command simulation experiment. The primary objective was to train the robot to navigate in two-dimensional space with random target directions, using the same reward terms described in 4.4 and the Omnidirectional Locomotion with Random Target Command. The experiments were conducted for 50k steps to assess the agent's ability to adapt to challenging outdoor conditions while achieving stable movement in both x- and y-directions.

The experimental setup introduced additional complexity compared to the Forward Locomotion experiments. The uneven cobblestone terrain, combined with varying friction levels and inclinations, created significant challenges for maintaining balance and executing precise movements. While curriculum-based learning could have accelerated training by introducing progressively harder tasks, it was not feasible due to the inaccuracy of the internal state estimation system. This inaccuracy prevented the computation of reliable performance metrics, which are essential for transitioning between curriculum stages.

For the CrossQ-based methods, both Joint Target Prediction and CPG-based approaches were considered. Among the JT-based methods, the one Euro filter was selected as the most promising approach to validate the hypothesis that it could achieve directional movement under real-world conditions. JT-based approaches, such as those using a low-pass filter, struggled to produce directional movement and failed to achieve meaningful progress, consistent with simulation observations detailed in 5.1.4. Similarly, CPG-based approaches such as CPG-VF and CPG-RT were omitted due to frequent terminations caused by safety measures during preliminary testing. Consequently, only CrossQ JT-OEF and CPG-D were evaluated in this experiment.

The challenges of this experiment were exacerbated by the limitations of the robot's internal state estimation, which consistently underestimated actual velocities. This discrepancy made it difficult to evaluate the agent's performance based on cumulative rewards, as perceived velocity improvements were minimal despite genuine progress. The high termination rates observed during training further complicated the learning process, highlighting the need for enhanced stability and adaptability in real-world settings.

The results of the training process are presented in figure 5.28. The findings underscore the difficulties of transferring learned behaviors from simulation to complex real-world environments while highlighting the potential of robust methodologies to adapt to challenging conditions.

CrossQ JT-OEF The CrossQ JT-OEF experiment took 25 minutes to collect 50k steps. During this time, the agent was unable to achieve movement in any direction and did not respond to directional conditioning signals. Throughout the training process, the agent remained in one place, jittering some of its limbs without making significant progress. This behavior indicated a lack of focus on any specific direction, leading the agent to adopt a mean movement strategy, which effectively resulted in standing still.

This outcome mirrors the behavior observed in simulation experiments, where the agent similarly failed to achieve directional movement under random directional commands. The Progressive Directional Expansion Curriculum, as described in the introduction, was not implemented in the real-world trials due to inaccuracies in the state estimation system, which prevented the computation of performance metrics required to advance curriculum stages. However, in simulation, this curriculum significantly improved the agent's performance by introducing structured progression, suggesting that its application could potentially have mitigated the limitations observed in this experiment.

CrossQ CPG-D The CrossQ CPG-D experiment required 33 minutes to complete 50k training steps. Initially, the agent frequently triggered the safety mechanisms due to instability, resulting in 43 terminations over the course of the training. These terminations occurred primarily in the early stages, highlighting the challenges posed by the uneven terrain and the agent's need to learn self-stabilization strategies.

As training progressed, the agent demonstrated significant improvement, eventually learning to avoid triggering the safety mechanisms by adapting its movements to the environmental conditions. The agent developed the capability to recover from unstable situations and effectively navigate less steep areas of the terrain. However, it struggled to maintain stability and performance on steeper inclinations. Movement speed was noticeably slower when moving sideways

compared to forward locomotion, indicating limitations in lateral adaptability. Although the agent learned to move backward during training, it refrained from doing so during the evaluation phase, suggesting a potential gap in behavior generalization.

In the evaluation runs, the agent successfully executed forward, left, and right movements (which can be seen in figures 5.29 to 5.31), albeit with some degree of yaw rotation, resulting in trajectories that were not perfectly straight. The maximum forward velocity achieved was approximately $0.2\,\text{m/s}$, while the maximum sideways velocity reached around $0.15\,\text{m/s}$. These results reflect the limitations of the current approach in achieving high speeds, particularly in lateral directions. A curriculum, such as the Omnidirectional Locomotion with Progressive Directional Expansion Curriculum, could have potentially improved performance by structuring the learning process and gradually introducing directional complexity. This approach might have mitigated the challenges observed during training, especially in stabilizing the agent's lateral movements and addressing the inconsistencies in trajectory control.

Conclusion The Omnidirectional Locomotion in Outdoor Environment experiment emphasized the challenges of adapting learned behaviors to real-world conditions, including uneven terrain, variable friction, and inclinations. These factors significantly increased the complexity of achieving stable omnidirectional locomotion compared to the Forward Locomotion experiments. Key performance metrics are outlined in table 5.5, with a comprehensive depiction of the training dynamics illustrated in figure 5.28. The CrossQ JT-OEF approach failed to achieve any meaningful directional movement, resorting instead to stationary behavior. This outcome underscores the limitations of Joint Target approaches in complex real-world scenarios without structured training strategies, such as the Progressive Directional Expansion Curriculum. The CrossQ CPG-D approach demonstrated the best performance but still showed considerable room for improvement. After overcoming early instability, which resulted in 43 terminations, the agent adapted to the outdoor environment, achieving forward, left, and right movements during evaluation. However, trajectories were not perfectly straight, and backward movement, learned during training, was not observed in evaluation, reflecting limitations in behavior generalization. Maximum velocities reached $0.2\,\text{m/s}$ forward and $0.15\,\text{m/s}$ laterally, falling short

of desired performance levels. Overall, while CPG-D showed promise in handling real-world challenges, the results highlight the need for enhanced training methodologies, improved state estimation, and structured curricula to enable robust and consistent omnidirectional locomotion in complex environments.

Approach	F. Spd. (m/s)	L. Spd. (m/s)	Yaw Ctl.	Nr. Falls	Durat. (min)
CrossQ JT-OEF	N/A	N/A	N/A	6	25
CrossQ CPG-D	0.2	0.15	poor	43	33

Table 5.5: Comparison of the Omnidirectional Locomotion in Outdoor Environment experiment. F. Spd. and L. Spd. denote forward and lateral speeds, respectively. Durat. is the duration of the training until all of the samples are collected. Velocity and yaw control are evaluated based on the agent's performance during the evaluation run. Yaw Ctl. indicates the quality of yaw control and categorizes the robot's ability to maintain directional stability: **Very Good** indicates minimal deviations (<10°), **Good** allows moderate drift ($10^\circ - 30^\circ$), **Medium** reflects noticeable drift ($30^\circ - 60^\circ$), and **Poor** shows significant instability (> 60°). **N/A** applies when yaw control is not evaluated. The number of falls and duration indicate the agent's learning progress and stability.

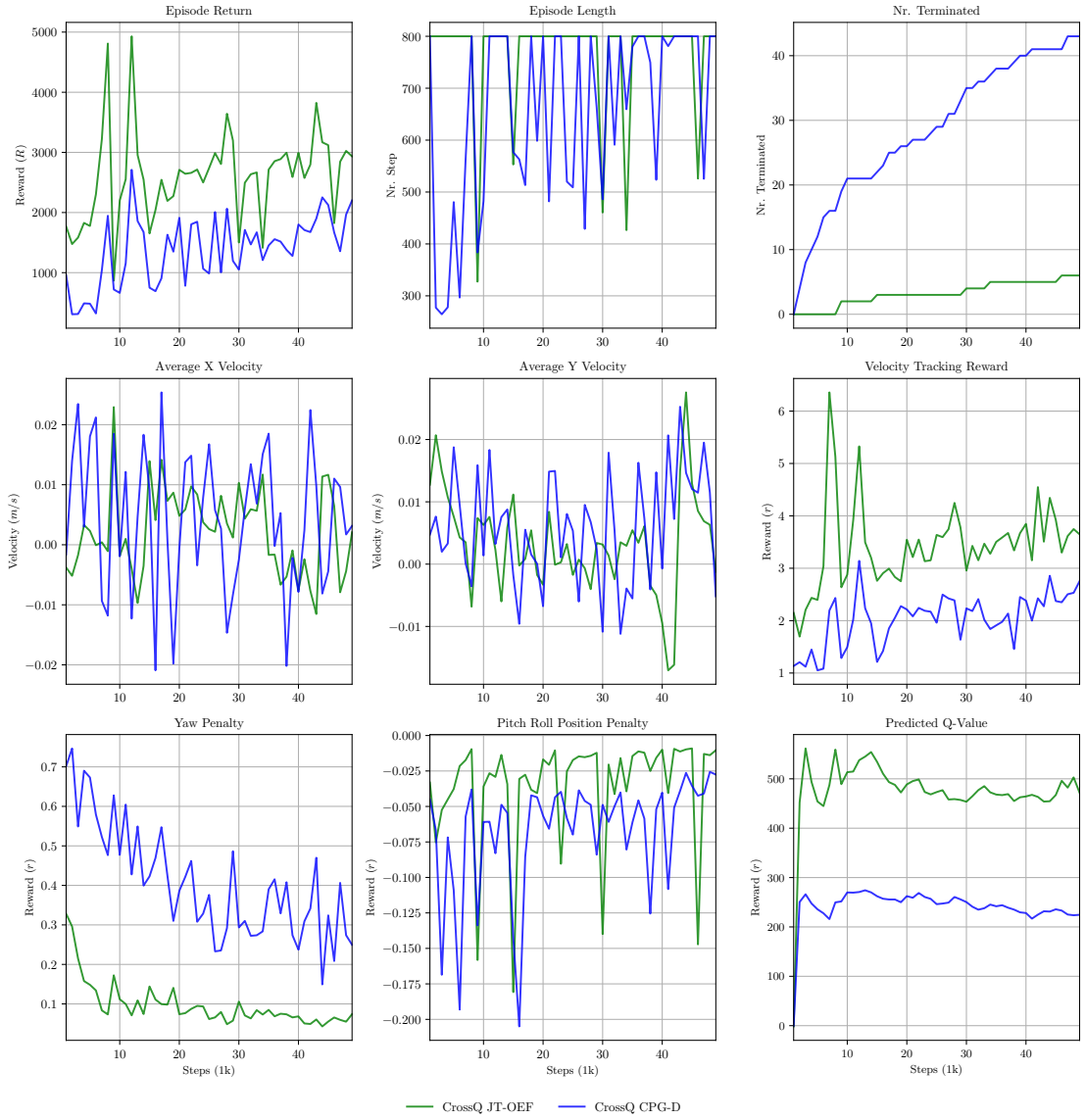


Figure 5.28: Performance of the JT-OEF and CPG-D for Omnidirectional Locomotion in the outdoor environment. The incorrect velocity estimation makes the velocity tracking reward an unreliable indicator of the approaches' performance.

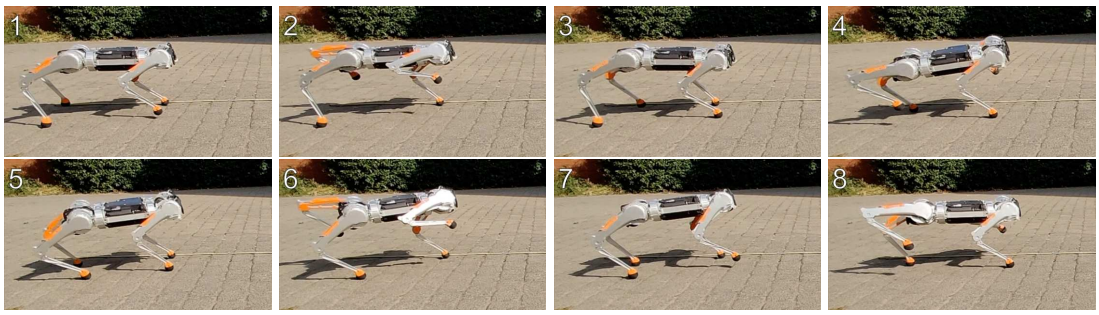


Figure 5.29: Evaluation run of CrossQ CPG-D, moving forward with a velocity of 0.2 m/s.

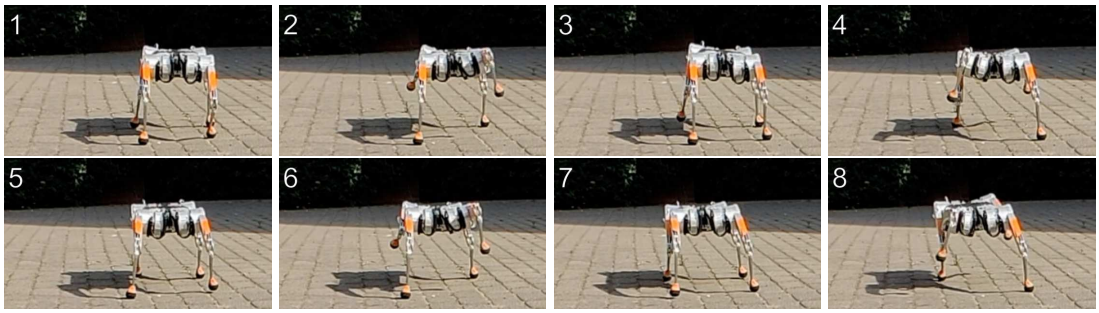


Figure 5.30: Evaluation run of CrossQ with CPG-D achieving a velocity of 0.15 m/s to the left.

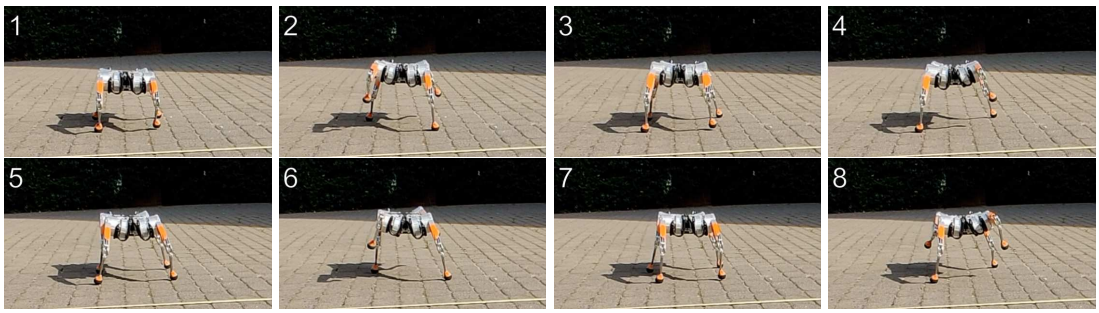


Figure 5.31: Evaluation run of CrossQ with CPG-D achieving a velocity of 0.15 m/s to the right.

6 Discussion

This thesis evaluates the performance of on-robot learning approaches for quadruped locomotion, emphasizing sample efficiency, robustness, and adaptability in both simulated and real-world environments. The discussion is organized into a comparison of reinforcement learning algorithms, joint target prediction methods, and Central Pattern Generators, culminating in an analysis of their relative strengths and limitations.

6.1 Comparison of High Sample-Efficient SAC Algorithms

Various high sample-efficient reinforcement learning algorithms, including SAC, SAC UTD 20, DroQ, and CrossQ, were compared in simulation, to determine the most effective algorithm for on-robot learning. CrossQ outperformed the other methods, achieving the target velocity within 2 minutes and completing training in 21 minutes due to its higher operating frequency of 40 Hz, which enabled faster data collection. CrossQ also exhibited greater stability with an average termination rate of 14 falls, compared to DroQ's 32 falls.

The computational efficiency of CrossQ not only enhanced the learning process but also made it feasible to conduct training on a MacBook equipped with an integrated GPU, rather than requiring a laptop with a dedicated GPU. This reduced hardware dependency further highlights CrossQ's potential for practical on-robot learning scenarios, where accessibility and portability are critical factors.

6.2 Comparison of Joint Target Approaches

Joint Target Prediction methods were evaluated using no filter (NF), one Euro filter (OEF), and low-pass filter (LPF). Table 6.1 summarizes the results across simulation, office, and outdoor environments.

JT Approaches	Simulation	Office	Outdoor
No Filter (NF)	2.2	N/A	N/A
One Euro Filter (OEF)	1.3	0.85	0.25
Low-Pass Filter (LPF)	1.15	0.4	0.23

Table 6.1: Performance of Joint Target Prediction methods in Forward Locomotion experiments (velocity in m/s). Simulation results show maximal achievable velocity. Real-world results use target velocities of 0.25 m/s.

In simulation, all methods achieved the target velocity of $0.5\,\text{m/s}$ within 20k steps, but differences became apparent in maximal velocity experiments. The NF approach achieved the highest velocity ($2.2\,\text{m/s}$) but at the cost of significant instability and jittering in early training phases. In contrast, OEF and LPF achieved similar maximal velocities ($1.3\,\text{m/s}$ and $1.15\,\text{m/s}$, respectively), but OEF converged to its velocity much faster, stabilizing within 20k steps, whereas LPF required significantly more steps to approach this performance.

The choice of filter plays a crucial role in balancing stability and responsiveness during joint angle predictions. The OEF operates as an adaptive smoothing mechanism that dynamically adjusts its filtering properties based on the rate of change of the input signal. This dynamic adjustment allows the OEF to preserve rapid changes when necessary while suppressing high-frequency noise, making it particularly suitable for tasks requiring both stability and responsiveness. In contrast, the LPF applies a fixed smoothing factor to the entire signal, heavily attenuating high-frequency components regardless of their relevance to the control task. While this approach effectively reduces noise, it can hinder the robot's ability to respond to abrupt changes, leading to slower convergence and less dynamic locomotion. The NF approach entirely avoids smoothing, passing the

raw signal directly to the controller. While this maximizes responsiveness and enables fast adaptations, it also amplifies noise and jittering, particularly during early training phases when the policy is still exploring the state-action space. This often results in erratic joint movements and high termination rates, as observed in preliminary experiments.

In the real-world Forward Locomotion in Office Environment experiment, OEF outperformed LPF, achieving 0.85 m/s compared to 0.4 m/s. The OEF method also demonstrated the ability to learn a stable trot gait, showcasing its adaptability to the low-friction surface. By restricting jitter while permitting rapid adjustments, the OEF approach enabled smoother and faster locomotion than LPF, which overly constrained the robot's movement, resulting in slower velocities and less dynamic gaits. The NF method, despite being one of the best-performing approaches in simulation, was excluded from real-world tests due to its high termination rates and erratic performance during preliminary evaluations. This discrepancy could be attributed to inaccurate state estimation, but it is also likely influenced by the nature of the training process. The NF approach exhibited significant jittering and instability during early training phases until a notion of a gait was developed. While this progression allowed the NF method to achieve unparalleled performance in simulation, its reliance on overcoming these early issues and its high degree of freedom in movement led to very unstable performance in the real world, making convergence either infeasible or excessively demanding.

When comparing these results to the baseline SAC UTD 20 algorithm, it becomes apparent that SAC UTD 20 was unable to perform effectively on the given system. This failure could be attributed to the limitations of the state estimation system, which introduced inaccuracies in velocity feedback, making learning more challenging. Additionally, the significant performance of JT-OEF (0.85 m/s) highlights the benefits of using the CrossQ algorithm as its seems that it is inherently more capable of handling complex locomotion tasks. The integration of the one Euro filter and the higher control frequency of 40 Hz further supported these improvements.

Compared to prior work by *Smith et al.* [67], our approach demonstrated more efficiency and adaptability. *Smith et al.*'s method, which operated on the Unitree A1 robot with automatic resetting, required 80k steps and 80 minutes to achieve a maximum velocity of 0.65 m/s. In contrast, our CrossQ-OEF method achieved 0.85 m/s within 20k steps (approximately 20 minutes), without relying on

automatic resetting or sophisticated techniques like Adaptive Policy Regularization (APRL) or network resetting. This performance increase, achieved despite the use of noisy velocity estimation and the absence of foot contact sensors, highlights the effectiveness of the OEF filter in enabling rapid and stable learning.

The Forward Locomotion in Outdoor Environment experiment posed additional challenges due to uneven terrain and increased surface friction. OEF reached 0.25 m/s, while LPF achieved 0.23 m/s. These reduced velocities, compared to the office environment, are likely due to the increased difficulty of lifting feet off high-friction surfaces, which requires more precise and forceful movements. The robot's lack of exteroceptive feedback further exacerbated these difficulties, leading to frequent foot dragging and collisions with uneven terrain. Despite these challenges, OEF maintained a slight advantage over LPF, demonstrating its robustness across different terrains.

6.3 Comparison of CPG Approaches

The evaluation of Central Pattern Generator-based approaches included constant frequency, variable frequency, and residual target configurations, with their results summarized in table 6.2.

CPG Approaches	Simulation	Office	Outdoor
CPG-D	0.55	0.3	0.3
CPG-VF	0.75	0.3	0
CPG-RT	0.55	N/A	N/A

Table 6.2: Performance of CPG-based methods in Forward Locomotion experiments (velocity in m/s). Simulation results show maximal achievable velocity. Real-world results use target velocities of 0.25 m/s.

In simulation, CPG-based methods demonstrated rapid learning, converging within 5k–10k steps for the target velocity experiment. Their robustness was reflected

by achieving zero terminations throughout the simulation experiments, showing their stability under controlled conditions. The maximal velocity tests revealed that CPG-VF achieved the highest velocity of $0.75\,\text{m/s}$, followed by CPG-D and CPG-RT, both at $0.55\,\text{m/s}$. However, the predefined gait configurations limited further velocity improvements, as the agent's flexibility in adapting the gait was restricted.

In the Forward Locomotion in Office Environment experiment, both CPG-D and CPG-VF achieved $0.3\,\text{m/s}$, exceeding the desired target velocity of $0.25\,\text{m/s}$. Despite their success, their performance was significantly hindered by premature safety triggers, which caused frequent terminations. The CPG-apporaches were configured with parameters suitable for rough terrain, including a high swing height and additional z-offsets, to ensure compatibility across diverse environments. Adjusting these parameters for smoother indoor conditions could improve trunk stability and reduce the number of terminations, thereby enhancing overall learning efficiency and reducing wall-clock training time.

CPG-RT, like JT-NT, presented an approach with significant initial freedom, making it challenging to pick up the notion of a gait during early training phases. While this high degree of flexibility allows for potentially powerful adaptations once a gait is established, it also increases the difficulty of learning and stability in environments with high uncertainty. In simulation, CPG-RT performed comparably or better to other CPG configurations, achieving a maximal velocity of $0.55\,\text{m/s}$. However, in real-world scenarios, these advantages turned into disadvantages due to increased environmental uncertainty and noisy state estimations. Moreover, the performance gap between simulation and reality could also be attributed to variations in the residual target implementation. In simulation, the residual target was applied with torque-based adjustments, whereas in real-world experiments, a joint angle-based approach was used to ensure compatibility with the robot's hardware constraints. This change, while necessary, likely influenced the dynamics of the learned gait and its stability, further contributing to the challenges faced by CPG-RT in real-world environments.

In the Forward Locomotion in Outdoor Environment experiment, CPG-D maintained its velocity of $0.3\,\text{m/s}$, demonstrating consistent performance across varied terrains. In contrast, CPG-VF exhibited difficulties during evaluation, focusing primarily on maintaining stability rather than forward motion.

For the Omnidirectional Locomotion experiments, CPG-VF was excluded due to its high termination rates and the associated risk to the robot's safety, similar to the exclusion of CPG-RT. CPG-D, while struggling initially with safety mechanisms, eventually adapted to multiple directions. However, its evaluation phase revealed limitations in generalization, as it refused to move backward while successfully executing movements in the forward, left, and right directions. These observations suggest that a Progressive Directional Expansion Curriculum could address such challenges by incrementally increasing task complexity, thereby enabling more effective generalization to dynamic, multi-directional tasks.

6.4 Comparison of Joint Target and CPG Approaches

The experiments highlighted distinct advantages and limitations of JT and CPG approaches, each excelling in specific environments and tasks.

JT approaches demonstrated superior adaptability and higher velocities in smooth indoor environments. For instance, JT-OEF achieved an impressive velocity of $0.85\,\mathrm{m/s}$ in the office setting, outperforming CPG methods. This performance can be attributed to the greater flexibility of JT methods, which allow the agent to dynamically adjust its movements without adhering to predefined gait patterns. However, this adaptability came at the cost of increased terminations during the exploratory phases of training. These terminations often required manual resets, posing a risk to the robot's hardware and significantly extending wall-clock training times.

On the other hand, CPG approaches excelled in robustness, particularly in unstructured outdoor terrains. The predefined patterns inherent to CPG methods reduced the complexity of the learning problem by providing a reliable framework for step generation. This allowed the agent to focus on higher-level locomotion tasks rather than learning how to generate individual steps. In outdoor experiments, CPG-D maintained a steady velocity of $0.3\,\mathrm{m/s}$, outperforming JT methods, which struggled with the increased friction and uneven terrain. Moreover, CPG terminations were primarily triggered by safety mechanisms rather than falls, enabling training to resume automatically without requiring human intervention.

One notable limitation of JT methods was their reliance on noisy proprioceptive feedback, which significantly hindered performance on challenging terrains. Without additional sensory inputs, such as foot contact sensors or visual data, the agent lacks the capability to accurately perceive and adapt to varying terrain conditions. Consequently, the agent tends to lift its legs as little as possible, often resulting in collisions with cobblestones or dragging feet over the surface. This behavior highlights the challenge of finding a single viable policy that can address all such scenarios simultaneously, rather than dynamically adapting to the specific demands of the terrain. Incorporating additional sensors could enable the agent to better evaluate the terrain and adjust its strategy accordingly, improving both efficiency and stability.

CPG methods, while robust, were constrained by their predefined gait structures, which limited their ability to achieve higher velocities. Although allowing the agent to dynamically adjust the frequency, as tested with the CPG-VF configuration, was hypothesized to enhance performance, it did not lead to any significant improvement. Instead, the increased flexibility introduced additional instability, highlighting the challenges of balancing adaptability and robustness.

In conclusion, the findings suggest that JT and CPG approaches have complementary strengths. JT methods are better suited for environments where adaptability and velocity are critical, while CPG methods excel in robustness and structured environments. A potential fusion of these approaches, such as the CPG-RT configuration, aimed to leverage the benefits of both by using CPG patterns as a foundation while allowing for dynamic adjustments. However, in practice, CPG-RT proved too challenging to train effectively in the real world and had to be discarded due to its inability to converge to a viable walking policy. This highlights the inherent difficulty of combining the complexity of JT learning with the structured nature of CPGs.

7 Conclusion and Outlook

This thesis investigated on-robot deep reinforcement learning methods for quadruped locomotion, focusing on enhancing the sample efficiency, robustness, and adaptability required for effective on-robot learning. By evaluating and comparing several approaches, including advanced soft actor-critic algorithms and central pattern generator frameworks, this research has contributed to improving quadruped locomotion performance and adaptability, especially in challenging and unstructured environments.

CrossQ was investigated in the context of on-robot learning, combined with a one Euro filter and an increased control frequency of 40 Hz, leading to remarkable performance gains. This combination enabled the quadruped robot to reach walking speeds of up to $0.85\,\mathrm{m/s}$ within less than 20 minutes of training, demonstrating a stable trot gait. Additionally, the design of the reward function was optimized to facilitate more efficient exploration, incorporating torque, pitch, and roll information to enhance the agent's stability and forward progression. The results showed that this reward structure, in conjunction with the improved filtering and control strategy, produced a reliable gait in a minimal amount of training time, confirming CrossQ's value in real-world applications.

Another contribution is the exploration of CPG-based approaches, an area previously underutilized in on-robot reinforcement learning for quadruped control. Among the tested CPG configurations, the approach with a constant frequency emerged as the most promising, demonstrating faster learning and greater robustness than the joint target prediction methods. This CPG configuration provided smooth, rhythmic locomotion patterns that allowed the robot to quickly adapt to directional commands. The experiments confirmed that CPG-based control methods could achieve coordinated multi-directional walking,

a notable step forward for on-robot learning applications where resilience to unstable ground conditions and rapid learning is crucial.

Looking forward, this research identifies several areas where future advancements could further enhance quadruped locomotion performance and adaptability. Improved state estimation, through techniques like visual SLAM (ViSLAM) [75] or foot contact-based velocity estimation, would be a key next step. ViSLAM could provide a robust external velocity reference, while foot contact sensing would refine the robot's proprioceptive velocity estimation. This foot contact information would also help optimize gait adjustments by detecting foot slippage or dragging, thereby minimizing footwear and increasing locomotion efficiency. A potential avenue for this sensory feedback is AnySkin [76], a technology that could offer seamless integration of foot contact data into the robot's control algorithms.

Additionally, integrating visual information from onboard cameras to perceive the terrain could further improve the quadruped's adaptability on uneven surfaces. This external sensory data would enable the robot to anticipate obstacles or sudden height changes, facilitating smoother and more stable walking across complex terrains. By combining visual perception, enhanced state estimation, and CPG-based control, future work could push the boundaries of on-robot learning for quadruped locomotion, setting new standards for robotic agility and autonomy in real-world settings.

Bibliography

[1] N. Li, J. Cao, and Y. Huang, "Fabrication and testing of the rescue quadruped robot for post-disaster search and rescue operations," in *2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI)*, pp. 723–729, 2023.

[2] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5761–5768, 2018.

[3] C. Dario Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3359–3365, 2017.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[5] M. Mathieu, S. Ozair, S. Srinivasan, C. Gulcehre, S. Zhang, R. Jiang, T. L. Paine, R. Powell, K. Żołna, J. Schrittwieser, D. Choi, P. Georgiev, D. Toyama, A. Huang, R. Ring, I. Babuschkin, T. Ewalds, M. Bordbar, S. Henderson, S. G. Colmenarejo, A. van den Oord, W. M. Czarnecki, N. de Freitas, and O. Vinyals, "Alphastar unplugged: Large-scale offline reinforcement learning," 2023.

[6] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," 2022.

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, Oct. 2020.

[8] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," 2019.

[9] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi8022, 2024.

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

[11] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," in *Proceedings of The 2nd Conference on Robot Learning* (A. Billard, A. Dragan, J. Peters, and J. Morimoto, eds.), vol. 87 of *Proceedings of Machine Learning Research*, pp. 270–282, PMLR, 29–31 Oct 2018.

[12] Z. Li, T. Chen, Z.-W. Hong, A. Ajay, and P. Agrawal, "Parallel q-learning: Scaling off-policy reinforcement learning under massively parallel simulation," 2023.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, IEEE, 1998.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[16] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[19] A. Radford, "Improving language understanding by generative pre-training," 2018.

[20] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[21] R. Bellman, "Dynamic programming," *science*, vol. 153, no. 3731, pp. 34–37, 1966.

[22] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[24] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[25] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning* (A. Billard, A. Dragan, J. Peters, and J. Morimoto, eds.), vol. 87 of *Proceedings of Machine Learning Research*, pp. 651–673, PMLR, 29–31 Oct 2018.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, PMLR, 10–15 Jul 2018.

[27] M. Hussing, C. Voelcker, I. Gilitschenski, A. massoud Farahmand, and E. Eaton, "Dissecting deep rl with high update ratios: Combatting value divergence," 2024.

[28] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka, "Dropout q-functions for doubly efficient reinforcement learning," 2022.

[29] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," *arXiv preprint arXiv:2101.05982*, 2021.

[30] A. Bhatt, D. Palenicek, B. Belousov, M. Argus, A. Amiranashvili, T. Brox, and J. Peters, "Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity," *CoRR*, vol. abs/1902.05605, 2019.

[31] S. Ioffe, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[33] K. Ota, D. K. Jha, and A. Kanezaki, "Training larger networks for deep reinforcement learning," *CoRR*, vol. abs/2102.07920, 2021.

[34] S. Butterworth *et al.*, "On the theory of filter amplifiers," *Wireless Engineer*, vol. 7, no. 6, pp. 536–541, 1930.

[35] G. Casiez, N. Roussel, and D. Vogel, "1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), p. 2527–2530, Association for Computing Machinery, 2012.

[36] P. Agarwal, S. Kumar, J. Ryde, J. Corso, V. Krovi, N. Ahmed, J. Schoenberg, M. Campbell, M. Bloesch, M. Hutter, M. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, R. Siegwart, J. Brookshire, S. Teller, M. Bryson, M. Johnson-Roberson, O. Pizzaro, S. Williams, N. Colonnese, A. Okamura, D. Dey, T. Y. Liu, M. Hebert, J. A. Bagnell, M. R. Dogar, K. Hsiao, M. Ciocarlie, S. Srinivasa, B. Douillard, N. Nourani-Vatani, C. Roman, I. Vaughn, G. Inglis, A. Dragan, J. Gillula, C. Tomlin, K. Gilpin, D. Rus, G. Gioioso, G. Salvietti, M. Malvezzi, D. Prattichizzo, M. Goodrich, S. Kerman, B. Pendleton, P. B.

Sujit, C. Hartmann, J. Boedecker, O. Obst, S. Ikemoto, M. Asada, K. Hauser, T. Kanda, L. Jaillet, J. M. Porta, A. Jain, C. Crean, C. Kuo, H. V. Bremen, S. Myint, D. Joho, G. D. Tipaldi, N. Engelhard, C. Stachniss, W. Burgard, B. Julian, S. L. Smith, M. Kazemi, J.-S. Valois, N. Pollard, M. Kovac, M. Bendana, R. Krishnan, J. Burton, M. Smith, R. Wood, M. Kuderer, H. Kretzschmar, C. Sprunk, S. Kuindersma, R. A. Grupen, A. G. Barto, T. Kunz, M. Stilman, A. Kushleyev, V. Kumar, D. Mellinger, M. Laffranchi, N. Tsagarakis, D. Caldwell, Y. Latif, C. C. Lerma, J. Neira, M. Li, A. Mourikis, S. Lim, C. Sommer, E. Nikolova, L. Liu, D. Shell, A. Long, K. Wolfe, M. Mashner, G. Chirikjian, I. Lysenkov, V. Eruhimov, G. Bradski, W. Maddern, M. Milford, G. Wyeth, B. Marthi, J. Nakanishi, S. Vijayakumar, E. Olson, P. Agarwal, I. Paprotny, C. Levey, P. Wright, B. R. Donald, Q.-C. Pham, Y. Nakamura, M. Phillips, B. Cohen, S. Chitta, M. Likhachev, F. Qian, T. Zhang, C. Li, A. Hoover, P. Masarati, P. Birkmeyer, A. Pullin, R. Fearing, D. Goldman, K. Rawlik, M. Toussaint, C. Robin, S. Lacroix, E. Rombokas, M. Malhotra, E. Theodorou, Y. Matsuoka, E. Todorov, S. Sarid, B. Xu, H. Kress-Gazit, A. Schepelmann, H. Geyer, M. Taylor, L. Sentis, J. Petersen, R. Philippsen, C. Taylor, A. Cowley, S. Tellex, P. Thaker, R. Deits, D. Simeonov, T. Kollar, N. Roy, P. Vernaza, V. Narayanan, H. Wang, G. Hu, S. Huang, G. Dissanayake, Z. Wang, M. P. Deisenroth, H. B. Amor, D. Vogt, B. Schölkopf, J. Peters, R. Wilcox, S. Nikolaidis, J. Shah, E. Wolff, U. Topcu, R. Murray, C. Yoo, R. Fitch, S. Sukkarieh, D. Zarubin, V. Ivan, T. Komura, D. Zelazo, A. Franchi, F. Allgöwer, H. Bülthoff, and P. R. Giordano, *State Estimation for Legged Robots: Consistent Fusion of Leg Kinematics and IMU*, pp. 17–24. 2013.

[37] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[38] R. Labbe, "Kalman and bayesian filters in python," *Chap*, vol. 7, no. 246, p. 4, 2014.

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

[40] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, "State estimation for legged robots on unstable and slippery

terrain," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6058–6064, 2013.

[41] S. Agarwal, A. Mahapatra, and S. S. Roy, "Dynamics and optimal feet force distributions of a realistic four-legged robot," *IAES International Journal of Robotics and Automation*, vol. 1, no. 4, p. 223, 2012.

[42] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[43] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[44] Duckietown, "lib-dt-apriltags," 2023. GitHub repository.

[45] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research. The Eleventh International Symposium: With 303 Figures*, pp. 561–572, Springer, 2005.

[46] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1047–1062, 2014.

[47] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, *et al.*, "Anymal-toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, 2017.

[48] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, Jan. 2019.

[49] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, Jan. 2022.

[50] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," 2022.

[51] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," 2023.

[52] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," 2023.

[53] S. Li, Y. Pang, P. Bai, Z. Liu, J. Li, S. Hu, L. Wang, and G. Wang, "Learning agility and adaptive legged locomotion via curricular hindsight reinforcement learning," 2023.

[54] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," 2018.

[55] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," 2022.

[56] J. Chen, J. Frey, R. Zhou, T. Miki, G. Martius, and M. Hutter, "Identifying terrain physical parameters from vision – towards physical-parameter-aware locomotion and navigation," 2024.

[57] D. Vogel, R. Baines, J. Church, J. Lotzer, K. Werner, and M. Hutter, "Robust ladder climbing with a quadrupedal robot," 2024.

[58] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," 2023.

[59] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 194–201, 2013.

[60] A. Sharma and K. Kitani, "Phase-parametric policies for reinforcement learning in cyclic environments," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[61] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," 2019.

[62] M. Kasaei, M. Abreu, N. Lau, A. Pereira, and L. P. Reis, "A cpg-based agile and versatile locomotion framework using proximal symmetry loss," 2021.

[63] G. Bellegarda and A. Ijspeert, "Cpg-rl: Learning central pattern generators for quadruped locomotion," 2022.

[64] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," 2020.

[65] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "Daydreamer: World models for physical robot learning," 2022.

[66] L. Smith, I. Kostrikov, and S. Levine, "A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," *arXiv preprint arXiv:2208.07860*, 2022.

[67] L. Smith, Y. Cao, and S. Levine, "Grow your limits: Continuous improvement with real-world rl for robotic locomotion," 2023.

[68] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

[69] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[70] "MAB Robotics | Legged Robotics Technology — mabrobotics.pl." `https://www.mabrobotics.pl/robotics`. [Accessed 01-10-2024].

[71] N. Bohlinger and K. Dorer, "Rl-x: A deep reinforcement learning library (not only) for robocup," in *Robot World Cup*, pp. 228–239, Springer, 2023.

[72] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[73] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[74] T. Fischer, W. Vollprecht, S. Traversaro, S. Yen, C. Herrero, and M. Milford, "A robostack tutorial: Using the robot operating system alongside the conda and jupyter data science ecosystems," *IEEE Robotics and Automation Magazine*, 2021.

[75] T. Taketomi, H. Uchiyama, and S. Ikeda, “Visual slam algorithms: A survey from 2010 to 2016,” *IPSJ transactions on computer vision and applications*, vol. 9, pp. 1–11, 2017.

[76] R. Bhirangi, V. Pattabiraman, E. Erciyes, Y. Cao, T. Hellebrekers, and L. Pinto, “Anyskin: Plug-and-play skin sensing for robotic touch,” 2024.