One Laptop, One Policy, Many Terrains: Efficient Training of Generalisable Quadruped Locomotion Policies via Adaptive Rewards

Sándor Felber[1,2], Thomas Corbères[1], Jiayi Wang[1], Antonin Bretagne[1], Steve Tonneau[1]

***Abstract*— Navigating unstructured terrains with quadruped robots requires sophisticated locomotion capable of adapting to diverse environmental conditions. Existing RL-based single-policy methods work well when fine-tuned for a subdomain of terrains and obstacles but do not generalise well to others. Hierarchical multi-policy approaches comprising several individual locomotion policies trained for each type of terrain with a higher level neural network to switch between low-level policies exhibit greater adaptability at the cost of intensive computational resource requirements for retraining and onboard deployment which results in a labour-intensive hand-tuning and distillation process. We propose the adoption of a terrain adaptive reward shaping method that leverages privileged height observations during training encapsulated in a singular locomotion policy. This offers robust adaptability to a wide range of terrains, and through independent terrain-specific rewards and efficient and easily adaptable training. Our method is trainable on a single commercial laptop GPU in a few hours and leverages real-time inferences from the elevation map of the robot's surrounding terrain for the classification of terrain type. We find that our method improves generalisability of single-policy locomotion to challenging terrains without the computational overhead of hierarchical multi-policy approaches which often require days to train. We validate the training efficiency, generalisability, and robustness of our method on existing and newly generated terrain types in simulation and in real-world experiments. The project website can be found here: `https://adaptive-rewards.github.io`**

I. Introduction

Research on quadruped locomotion aims at proposing robust means to control the motion of a legged robot in arbitrary terrain. In recent years, this research has been driven by industry and the motivation to tackle dangerous and repetitive tasks such as inspection of off-shore oil rigs or participating in first-response to natural disasters [1]–[3]. This industrial interest has resulted in a variety of affordable and reliable quadruped platforms that benefit the research effort [4]–[7].

The contemporary adoption of Deep Reinforcement Learning (DRL) as a means to control quadruped robots offers a seducing alternative to traditional control techniques, with the promise of learning a control policy offline that can be queried efficiently at runtime, rather than solving a computationally expensive trajectory optimisation problem online. Various learning-based frameworks have successfully enabled quadrupeds with agile locomotion skills resembling human parkour as they climb stairs twice their height [8] and jump or leap over gaps twice their length [9], [10], or robustly perform dynamic gaits and handshakes [11], [12].

[1]University of Edinburgh [2]Massachusetts Institute of Technology `felber@mit.edu`

One area of progress regarding DRL frameworks concerns the ability of a learned policy to generalise to new environments without sacrificing performance on existing ones as the learning progresses [13]. Another challenge is identifying the optimal set of reward functions to fulfil the role of an information guide to the actor-critic learning framework [14]. Overall, DRL often struggles with effective exploration and generalisation of rewards, particularly in environments where rewards are sparse and less frequent.

Multi-policy approaches, such as those by [8], [15]–[17], have demonstrated greater success in generalising across diverse terrains. These methods involve training multiple individual locomotion policies for each terrain type and utilising a high-level neural network to switch between them. However, increasing the number of networks also escalates complexity and training time, as exemplified by Zhuang et al.'s method [15], which trains individual policies for various terrain types necessitating a relatively long training and at least two GPUs for policy distillation. Additionally, large multi-stage policies can demand substantial on-board computing resources [18], [19].

Approaches using a single DRL-policy [9], [10], [20]–[23] work well for a subdomain of terrains and obstacles that they have been carefully fine-tuned for but their generalisability to additional new terrains remains a challenge. Relevant recent work is summarised in Table I.

Regardless of the policy training approach, sim-to-real transfer comes with the limitation that simulation is often unable to encompass all possible terrains that will be encountered at deployment, leading to unmodeled scenarios and ultimately failure to generalise to one of the environments in the target application. This necessitates the capability of a locomotion policy to be adaptable and retrainable at the deployment site quickly and with minimal computational requirements.

However, adding a new terrain type or obstacle into an existing framework with optimised rewards requires a new set of rewards that will work for both the existing environments and the new one. To find an optimal set of reward functions for the new set of terrains, iterative reward shaping can be attempted, a laborious and ineffective tuning process that ultimately leads to a suboptimal compromise between the two optimal sets of rewards for the old terrains and the new terrain. When a new terrain requires completely opposite rewards to ones beneficial for traversing others, it is often impossible to successfully train a policy for both

TABLE I: Comparison of various methods on challenging locomotion tasks relative to the robot's body proportions. Skills in Continuous Sequence refer to the policy's ability to tackle all challenges in one run one after the other. [1]

Method	Robot	Locomotion Skills						Skills in Sequence	# of NNs	Flat (Only) Training Time*	Training Time*
		Climb	Gap	Ramp	Crouch	Trench Tilt	Bridge Tilt				
Rudin et. al [20]	AnymalC	1.1	0.75	×	×	×	×	✓	1	3 min	18 min
Hoeller et. al [8]	AnymalC	2	1.5	×	×	×	×	✓	8	N/A	N/A
Zhuang et. al [15]	Unitree-A1	1.6	1.5	×	0.76	0.93	×	×	5	6 h	82 h
Cheng et. al [10]	Unitree-A1	2	2	37°	×	×	×	✓	1	N/A	26 h
Chane-Sane et. al [22]	SOLO12	0.93	×	24°	0.77	×	×	✓	1	10 min	N/A
Adaptive Sim (ours)	SOLO12	0.73	0.3	32°	×	0.83	0.8	✓	1	23 min	3 h
Adaptive Real (ours)	SOLO12	0.25	×	15°	×	0.8	0.8*	✓	1	23 min	3 h

terrains as shown in the supplementary video.

To address the above and the problem of computationally efficient generalisability in quadruped locomotion, we propose to use terrain adaptive reward shaping, which decouples the reward functions and weights to any distinguishable terrain type (within the limits of the simulation and policy observations) while leaving the reward function on the original terrain unchanged. This allows the integration of new terrains without compromising performance on existing terrains as the rewards and the training for existing terrains can remain untouched, while the new rewards can be tailored to the new terrain without constraints from the previous terrain. New rewards can be added and existing reward functions and scales can be disabled or enabled when the agent encounters the specific terrain type, allowing for a higher degree of customisability of the trained behaviour with minimal computational overhead yielding a similar training time to existing single-policy methods. The proposed architecture for testing and validating our approach is displayed in Figure 1.

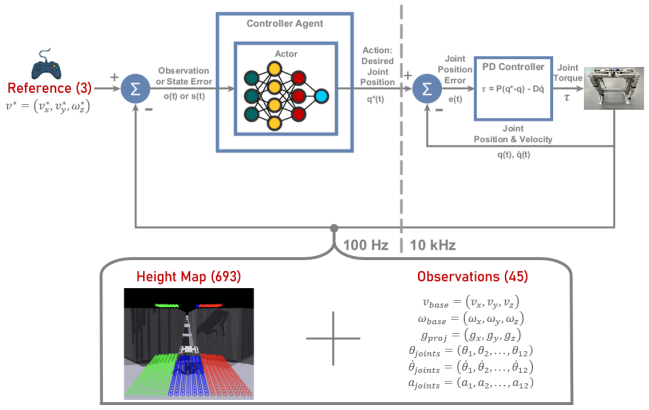


Fig. 1: Training Architecture

Our method enables the training of a single locomotion policy that is able to adopt rewards during massively parallel training specific to the environment on any new distinguishable terrain type. It is easily trainable on a single commercially available laptop for new terrains at deployment. This has been validated through simulations in IsaacGym and PyBullet, as well as through empirical testing in real-world experiments on the SOLO12 quadruped. These experiments confirm that our method yields an agile and robust DRL policy transferable to real-world applications that require generalisation to terrains with competing reward landscapes. Our method enables generalisation and sequential, continuous traversing across a range of obstacles which previously has only been seen in segmented, multi-policy controllers [8], [18] but with the computational efficiency and simplicity offered by single-policy controllers [10], [15].

1) **An adaptive reward shaping training framework integrating massively parallel DRL with curriculum learning:** to demonstrate the enhanced adaptability of the locomotion policy, we create and add two new obstacles (narrow trenches and bridges) requiring different, competing rewards against the existing baseline terrains in [20].
2) **A generalisable locomotion policy efficiently trainable on a commercial laptop:** we show that our method enables the efficient training of a single continuous locomotion policy traversing various obstacles of unstructured environments in a single run without resets or restarts. We evaluate efficiency by evaluating training time in Table I.
3) **Evaluation of sim-to-real performance:** we evaluate performance gaps between simulation and real-world experiments through comparing success rates and reporting the measured control loop frequencies.

II. Related Work

A. Optimal Control

Optimal control-based strategies have predominantly guided legged mobile robotics, demonstrated by various trajectory optimisation and model-predictive control (MPC) methods on quadrupeds such as MIT's Cheetah and MiniCheetah [24], [25], ETH's Anymal, and StarlETH [26]–[30], among others [31]–[36]. Although these methods have demonstrated capability for enabling agile and autonomous locomotion in controlled settings, their generalisable deployment across diverse environments is still constrained [29], [37]–[39].

B. Learning-based Methods

More recently learning based methods have demonstrated a wide range of skills on quadruped robots, including stairs

[1] All training times refer to training the locomotion policy (without policy distillation) tested with NVIDIA RTX 4090 Mobile GPU (16GB VRAM). Adaptive Rewards Sim refers to IsaacGym. Our real-world experiments were completed without perception with synthetic height maps. *To avoid hardware damage, bridge hardware tests were done on flat terrain.

[40], [41], trotting and running [42]–[44], jumping [9], [45], [46], crouching [22], [47] and various other parkour-like skills [8], [10], [15] and even interacting with dynamic objects, such as dribbling a ball [48] but transitions between discrepant terrains remains a challenge.

Notably, Margolis et. al [21] proposed encoding multiple locomotion strategies trained on flat ground into a single policy while providing a human operator a set of tunable parameters. The operator can manually tune the gait pattern enabling robust locomotion in unseen environments including stair climbing, crouching, and traversing sudden drops. Limitations include requiring an operator at deployment and reduced performance on individual skills like high-speed running.

C. Training Efficiency

Considering efficient training, many recent works focus on optimising training efficiency for an idealised quasi-planar surface with some smaller obstructions. Authors of [20], [49]–[51] report training times in the domain of minutes. More sophisticated control systems able to tackle more challenging non-planar terrains have time-consuming training pipelines that train separately for each skill [15]. For instance, [8] requires 8 different neural networks some of which are interdependent. Such multi-stage hierarchical frameworks require GPU clusters to train and extended training times for all neural networks. Another example is [18], where deploying the trained DRL policy requires a large amount of computational resources exceeding the available onboard computational capabilities of the quadruped for agile motion making real-world deployment impossible without additional computers mounted onto the robot or remotely aiding it.

D. Adapting Rewards

Two notable ways for the application of dynamic rewards include using a customised reward function for each different environment relying on domain-specific knowledge [52], [53] or the automated [54]–[57] encoding of knowledge into a (possibly potential) reward function [58]. In this work, we propose and validate a method under the scope of the former via incorporating adaptable reward functions triggered by inferring the terrain type surrounding each individual agent during training.

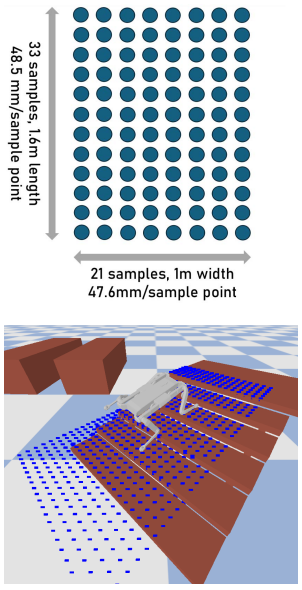


Fig. 2: Height Map (Top) & Visualisation in PyBullet (Bottom).

To further enhance the agent's adaptability and learning efficiency, we incorporate curriculum learning. This strategy involves starting with simpler tasks and progressively increasing the difficulty, a method shown to improve learning outcomes [20], [59]–[61].

TABLE II: Summary Table of Observations

Feature	Description
Base Linear Velocity	$v_{\text{lin}} = (v_x, v_y, v_z) \in \mathbb{R}^3$
Base Angular Velocity	$\omega = (\omega_{\text{roll}}, \omega_{\text{pitch}}, \omega_{\text{yaw}}) \in \mathbb{R}^3$
Projected Gravity on Base	$g_{\text{proj}} = (g_x, g_y, g_z) \in \mathbb{R}^3$
Commands	$\text{cmd}^* = (v_x^*, v_y^*, \omega_{\text{yaw}}^*) \in \mathbb{R}^3$
Joint Positions	$\theta = (\theta_1, \ldots, \theta_{12}) \in \mathbb{R}^{12}$
Joint Velocities	$\dot{\theta} = (\dot{\theta}_1, \ldots, \dot{\theta}_{12}) \in \mathbb{R}^{12}$
Previous Actions	$a_{\text{prev}} = (a_1, \ldots, a_{12}) \in \mathbb{R}^{12}$
Surrounding Elevation Map	$\mathbf{H} \in \mathbb{R}^{33\times 21}$

III. Preliminaries

Shown in Figure 1, the proposed control system architecture integrates a neural network-based controller (actor) and a proportional-derivative (PD) controller for robotic joint actuation. It processes a reference velocity against current joint states to determine errors, which the actor uses to set desired positions. These are then refined by the PD controller to compute the necessary torques at high frequencies (PD controller at 10 kHz and actor at 100 Hz). We track reference linear and angular velocities and yaw while robustly navigating challenging terrains without necessitating policy swaps, thus ensuring the generalisability and computational efficiency of the method.

A. States and Observations

The control system's input comprises 48 proprioceptive and 693 height measurements provided as privileged information during training. The height map grid is made up of approximately equally spaced sampling points arranged in a grid of 33x21 over the 1.6 m long and 1 m wide area around the centre of the robot's base. This yields a longitudinal sampling rate of approximately 48.5 mm and a lateral sampling rate of 47.6 mm as shown in Figure 2. The observations serving as the input of the policy's neural network are summarised in Table II.

While authors of [62] argue that torque control is more robust to larger external disturbances than position control, [63] showed that it is harder to learn torque control directly which is supported by other works in the literature opting for position control [41], [48], [64]–[66].

B. Actions

The action space comprises a 12-dimensional vector, $\mathbf{a} \in \mathbb{R}^{12}$, representing the radial displacement (angle) for the i^{th} rotational joint. The actions in $\mathbf{a}$ are the learned displacements (Δq_t) and the target joint positions can be computed as:

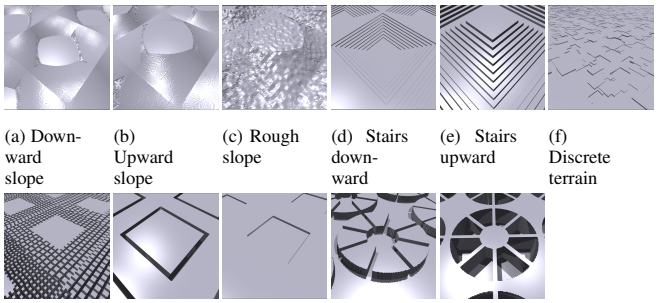
$$q_t^{\text{target}} = q_{\text{init}} + \lambda_q \Delta q_t, \tag{1}$$

where q_{init} are the robot's nominal joint configuration around which the policy actions are centred [66]. λ_q is defined as a constant scaling factor for the output actions before adding to q_{init} and subsequently clipping to avoid out-of-bound control commands that could damage the hardware or the environment. The base position and rotation are indirectly controlled and measured via SOLO's built-in IMU through the steps of the output learned action.

C. Training Setup

Curriculum learning structures the learning process progressively, starting with simpler tasks and gradually introducing more complex ones as the agent's capabilities improve for better exploration of the solution space and avoidance of local minima [60], [67]–[69]. The success of curriculum learning has been demonstrated for quadruped robots in [20], [41], [42], [70], [71].

To demonstrate the plasticity of our method we add two new environments to the 2D-matrix terrain tile configuration proposed by [20]. The environment includes twenty terrain tile columns and ten rows increasing in difficulty, from smooth surfaces to near-impossible obstacles as shown in Fig 3. Narrow passages are simulated with concentric radial trench and bridge terrains, with 1x1m spawning areas and headings quantised to the nearest 45°. Difficulty adjusts automatically based on traversal success as we train for 10-15,000 policy update iterations. No negative rewards are applied for 500 iterations to encourage exploration, followed by a gradual increase in penalties over 4500 updates following $P(x) = x^{1.5}$, where x denotes the episode count.



(a) Downward slope (b) Upward slope (c) Rough slope (d) Stairs downward (e) Stairs upward (f) Discrete terrain

(g) Stepping stones (h) Gap terrain (i) Pit terrain (j) Trench (k) Bridge

Fig. 3: Terrain Types Used In Training

Domain randomisation (noise parameters in the training) is introduced to simulate real-world perturbations during training as described in Table III.

Ground friction coefficients are varied between 0.5 and 1.25, and random dynamic pushes are exerted every 15 seconds to mimic unexpected external forces encountered in application environments.

TABLE III: Noise Parameters

Noise Type	Scaling
Overall Noise Level	1.0
Position Disturbance (DOF)	0.01
Velocity Disturbance (DOF)	1.5
Linear Velocity Noise	0.1
Angular Velocity Noise	0.2
Gravity Variations	0.05
Height Measurement Noise	0.1
Classification Noise	0.05

IV. Adaptive Reward Shaping

We combine adaptive reward shaping with privileged height map information during training, curriculum learning, and massively parallel DRL using Proximal Policy Optimization.

We utilise a set of 21 reward functions and penalties available on the project website. We build on the rewards proposed by [66], which provide a tested foundation for robust locomotion on smooth and rough flat terrains. We then enhance this to adapt to inclined terrains, stairs, and steps. When the agent infers from the elevation map that it is about to enter a specific terrain type, such as a narrow bridge, then a flag for this specific terrain type triggers a swap of rewards or weights for this terrain, which in turn changes the trained behaviour (such as collision aversion, stance width, or step height) of the quadruped. The comprehensive reward function parametrised by time t is defined as:

$$r_t = \mathbf{w} \cdot \mathbf{r} = \begin{bmatrix} w_{\text{linear velocity}} & w_{\text{foot clearance}} & \cdots & w_{\text{termination}} \end{bmatrix} \cdot \begin{bmatrix} r_{\text{linear velocity}} \\ r_{\text{foot clearance}} \\ \vdots \\ r_{\text{termination}} \end{bmatrix} \quad (2)$$

The vector $\mathbf{w}$ contains weights for each reward component, allowing precise adjustments to tailor the optimisation landscape, thereby influencing the behaviour of the robot directly. Our framework proposes dynamically adjusting both the weight and the function based on terrain type, enhancing training plasticity and eliminating competing reward functions to ensure optimal performance. In the following section, we demonstrate our method through an example.

A. Example Terrain Adaptive Reward: Foot Clearance

To be able to effectively approach and enter a narrow passage, the quadruped needs to reduce the step height to smaller, more nuanced steps that are more suitable for this terrain type. By separating the original reward into two conditional rewards, both the reward function and its weight can be freely changed without affecting the default reward to facilitate this.

An example penalty function pair for the original terrain with scaling factor 25 and the new terrain (scaling factor of 5) is:

$$-25 * r_{\text{clearance}} = \begin{cases} 0, & \text{in trench or on bridge} \\ -25 * \sum_{i=1}^{n} (z_{\text{foot}_i} - z_{\text{target}})^2 \sqrt{v_{\text{foot}_i}^2}, & \text{otherwise} \end{cases} \quad (3)$$

$$-5 * r_{\text{clearance}^*} = \begin{cases} -5 * \sum_{i=1}^{n} (z_{\text{foot}_i} - z_{\text{target}})^2 \sqrt{v_{\text{foot}_i}^2}, & \text{in trench or on bridge} \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

When entering narrow passages, where lateral space is highly constrained, the default foot height reward needs to be reduced to mitigate penalties associated with divergence from the reference point. The lower foot clearances enable the quadruped's front or rear legs to fit through the entrance of the narrow trench and navigate it more effectively, otherwise, the quadruped struggles to traverse the narrow passage. It is important to note that the low foot clearance would hinder locomotion on uneven rough terrain, and elevations such as stairs and steps.

Shaping the foot clearance reward, there are two ways to achieve the desired terrain-specific behaviour without changing the reference: 1. tuning the weight of the reward to enable a broader exploration of the parameter space without significant rewards or penalties incurred during the training, or 2. change the terms of the function defining the foot height reward, z_{target}, to scale differently fostering a decreased step height together with quicker steps. By utilising

TABLE IV: Success Rates on Different Terrains across Platforms. Ours includes both the privileged height map and adaptive rewards in simulation and synthetic height maps in the real world.

Terrain Type	**IsaacGym Simulation**			**PyBullet Simulation**			**Real-World Deployment**		
	Baseline	+Height Map	Ours	Baseline	+Height Map	Ours	Baseline	+Height Map	Ours
Planar Terrain	100%	100%	100%	100%	100%	100%	100%	100%	100%
Planar with Random Obstacles	60%	100%	100%	60%	80%	100%	60%	100%	100%
Inclined Terrain & Stairs	0%	80%	100%	0%	60%	60%	0%	60%	60%
Trench/Bridge	0%	0%	90%	0%	0%	80%	0%	0%	80%
All but Inclines/Stairs Continuously	0%	0%	90%	0%	0%	90%	0%	0%	80%
All Continuously	0%	0%	100%	0%	0%	80%	0%	0%	40%

separate definitions for the default and the trench-specific foot clearance reward as in Eq. 3 and 4 respectively, changing the scale and the function terms at once is not a mutually exclusive operation—any weight could be assigned to any reward function. During training, the actors interact with the various terrain types as they approach, traverse, and leave them, and in doing so they automatically learn a seamless transition between the various optimisation landscapes of the different reward functions. By adjusting the weights and the terms of the reward function, we enable the RL agent to exhibit fundamentally different behaviours tailored to specific environmental scenarios provided that the agent can identify which terrain it is traversing from a function.

B. Inferring Terrain Type from The Elevation Map

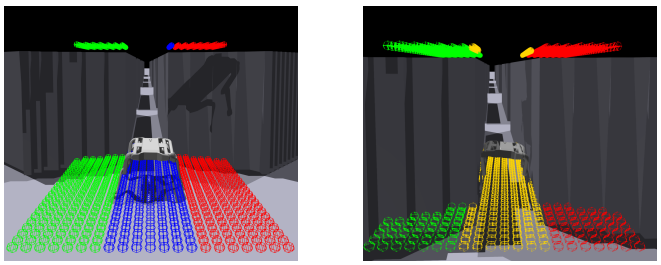


Fig. 4: Agent before entering a trench (left) and in trench (right), with amber colour indicating the change of rewards.

For example, the algorithm compares mean heights from two longitudinal side stripes (shown in green and red in Figure 4) to that of the middle stripe to recognise trench-like conditions at each episode rollout encompassing approximately 0.5 s of simulation time. This comparison informs the agent when to adjust its gait and tilting strategy for optimal navigation through narrow passages. If the policy is played in simulation, the change of terrain causes the agent to transition to the gait learned under the new reward paradigm specific to that terrain. The implementation of this dynamic tilting reward function ensures that the robot can effectively modify its posture and gait in real-time, enabling it to traverse narrow passages.

V. Simulation and Hardware Experiments

A. Training Efficiency

Using 4096 environments in parallel training for 10000 iterations and a batch size of 98304 on a terrain matrix of 20x10, the complete training can be completed in less than 3 hours locally on a commercially available laptop with i9-13900H CPU and NVIDIA RTX 4090 Mobile (16GB VRAM) GPU and in approximately 5-6 hours on a 2080 Ti Desktop GPU with 12GB of VRAM. This is significantly more efficient than existing methods. We slightly varied the terrain ratios, curriculum start and end, and rewards scales across 20 training runs, all of which trained successfully.

B. Evaluation in Simulation and Real-World

Adapting gait to new environments can be observed from simulation results: note the narrow stance of the agent combined with its base being tilted (roll) in Fig. 4, allowing it to fit into the narrow trench. Collision reward adoption is shown in the supplementary video.

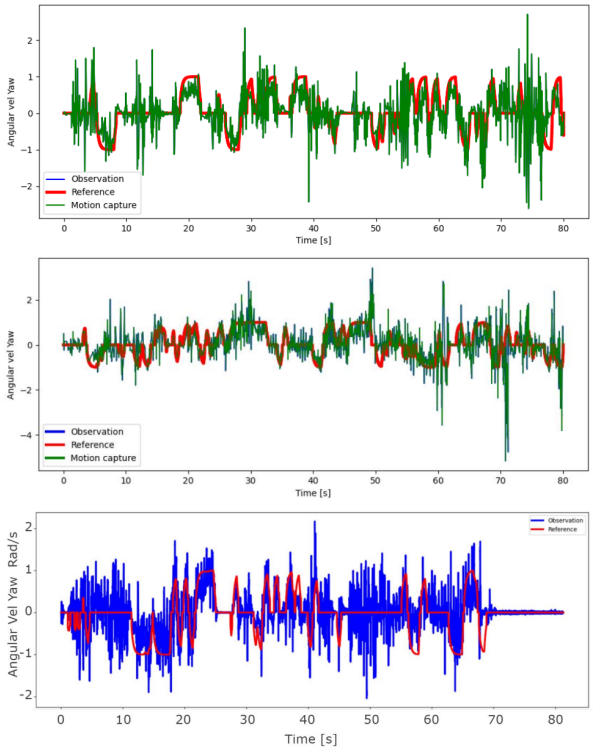


Fig. 5: Angular velocity (yaw) [rad/s] tracking in simulation (PyBullet) without (top) and with (middle) privileged height map information, and in the real world (bottom). While the tracking is noisy attributable to taking steps to and from elevated surfaces without perception in the real world (falling and colliding with walls), it remains accurate and follows the reference. (Motion capture and observation measurements overlap in all cases.)

To evaluate the performance of the policy, an obstacle

course was created involving rough inclined terrain, gravel, and stairs to climb together with a narrow bridge and trench to demonstrate the adaptability of the method shown in the supplementary video. We define success rate as successfully traversing the given individual obstacle in 20s or all of the obstacles in 2 minutes without getting stuck. Fig 5 demonstrates a successful sim-to-real transfer. The reported trench width is 0.22m in simulation and 0.25m (SOLO12 is 0.3m wide, but its "normal" walking gait is approx. 0.4m wide) where we find a similar success rate.

The robustness of the locomotion policies was assessed through 10 attempts of evaluation with the reference commands of Table V spanning simulations in IsaacGym, PyBullet, and real-world field tests summarised in Table IV demonstrating the usefulness of incorporating privileged height maps and adaptive rewards into the baseline policy. Note that the lack of perception makes tackling inclined terrains and stairs challenging.

TABLE V: Command parameter ranges.

Command Parameter	**Minimum Value**	**Maximum Value**
Linear Velocity X [m/s]	-1.0	1.0
Linear Velocity Y [m/s]	-0.8	0.8
Angular Velocity Yaw [rad/s]	-0.65	0.65
Heading [rad]	-3.14	3.14

C. Transferability

Although the trained policy transfer from IsaacGym to PyBullet to SOLO12 produces similar success rates, when playing the trained policy, we find a slight, but noticeable difference in the apparent gait of the quadruped between simulation and the real world. However, the velocity tracking remains uncompromised as shown in Fig 5. A notable discrepancy between PyBullet and hardware experiments is the peak in computation time at initialisation as shown in Table VI. Overall, the policies trained transferred successfully to the real robot with a small sim-to-real gap. This is supported by the findings of [36] attributing the relatively small sim-to-real gap to the lightweight form factor, low inertia actuators, and high control bandwidth.

D. Neural Network Computation and Control Loop Performance

Results in Table VI indicate that while the simulation environment can achieve high average performance, it exhibits substantial variability under worst-case conditions which can be attributed to the computational load of the PyBullet Simulator. In contrast, the hardware experiment implementation which does not rely on the simulator demonstrates consistent performance with less severe worst-case degradation maintaining an inference frequency of 320.51 Hz, which is more than 3 times the frequency of the control loop at 100 Hz. On average, hardware inference runs at 0.17 ms (5882 Hz), yielding a high-frequency control loop yielding a responsive, agile, and robust quadruped walking controller.

TABLE VI: Summary of results of 5-5 tests with a duration of 5 seconds in PyBullet simulator and on the SOLO12 quadruped measuring the frequency of neural network inference and of the full control loop.

Average Results					
	NN Computation Time [ms]		Control Loop Duration Frequency [Hz]		NN % of Loop
	Avg	Peak	Avg	Peak	
Simulation	0.43	2.11	2326	474	43%
Hardware	0.17	1.68	5882	595	17%
Worst Results					
	Avg	Peak	Avg	Peak	
Simulation	63.48	19.40	14.60	50.55	98%
Hardware	3.12	0.32	320.51	3125.00	19%

VI. Discussion and Conclusion

Regarding limitations, we find the lack of on-board or external perception as the main limiting factor of hardware experiments since the SOLO12 is unable to carry such equipment due to its lightweight actuators and low-cost 3D-printed design. The use of a 2.5D height map for the observation matrix, assigning a single elevation value (z) to each (x, y) coordinate around the robot, constrains training scenarios that involve navigating beneath obstacles. While the current elevation map adequately represents terrain variations in height, this model does not support scenarios where a robot must crouch under overhangs, thus limiting the robot's exposure to complex three-dimensional manoeuvring challenges in simulated environments. While the SOLO12 robot can jump with its actuators producing a peak torque of 3 Nm, we found that its 3D-printed parts can break on larger impacts and did not pursue jumps further. To explore more dynamic tasks in the future we plan to integrate active perception via an external motion capture system.

To improve the training efficiency, differentiable simulation for better convergence could be explored as proposed by the recently released work in [51]. This could further enhance the training efficiency by using the true gradient of the loss instead of a surrogate approximation.

In conclusion, our adaptive reward shaping framework enhances the adaptability of quadruped locomotion policies by dynamically adjusting rewards based on real-time inferences about the robot's surroundings. We demonstrated the enhanced adaptability through the integration of two new terrains into an existing training framework. Our training method overcomes the limitations of competing reward functions thereby eliminating the training and deployment computational overhead of multi-policy frameworks. We train and test an example policy to adapt to various environmental conditions without extensive retraining on a single commercially available laptop. Validated on the SOLO12 quadruped robot, our approach demonstrates practical applicability in scenarios like search and rescue. Future work will aim to automate the trigger condition with pattern recognition to further ease the adoption of adaptive reward shaping.

For more information, example videos, and to get started using our adaptive reward framework, we recommend visiting our website hosted at https://adaptive-rewards.github.io

REFERENCES

[1] X. Hu, F. He, P. Xiao, T. Wang, D. Zhang, X. Zhou, and Y. Fan, "Design of a quadruped inspection robot used in substation," in *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, vol. 4, 2021, pp. 766–769.

[2] K. Hashimoto, T. Matsuzawa, T. Teramachi, K. Uryu, X. Sun, S. Hamamoto, A. Koizumi, and A. Takanishi, "A four-limbed disaster-response robot having high mobility capabilities in extreme environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5398–5405.

[3] H. Kolvenbach and M. Hutter, "Life extension: An autonomous docking station for recharging quadrupedal robots," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 545–557.

[4] J. He and F. Gao, "Mechanism, actuation, perception, and control of highly dynamic multilegged robots: A review," *Chinese Journal of Mechanical Engineering*, vol. 33, no. 79, 2020. [Online]. Available: https://doi.org/10.1186/s10033-020-00485-9

[5] R. W. Xu, K. Chin Hsieh, U. H. Chan, H. Un Cheang, W. K. Shi, and C. Tin Hon, "Analytical review on developing progress of the quadruped robot industry and gaits research," in *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, 2022, pp. 1–8.

[6] H. Chai, Y. Li, R. Song, G. Zhang, Q. Zhang, S. Liu, J. Hou, Y. Xin, M. Yuan, G. Zhang, and Z. Yang, "A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach," *Biomimetic Intelligence and Robotics*, vol. 2, no. 1, p. 100029, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667379721000292

[7] R. W. Xu, K. Chin Hsieh, U. H. Chan, H. Un Cheang, W. K. Shi, and C. Tin Hon, "Analytical review on developing progress of the quadruped robot industry and gaits research," in *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, 2022, pp. 1–8.

[8] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *arXiv preprint arXiv:2306.14874*, 2023.

[9] G. Bellegarda, C. Nguyen, and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," 2023.

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

[11] J. Viereck, A. Meduri, and L. Righetti, "Valuenetqp: Learned one-step optimal control for legged locomotion," in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, ser. Proceedings of Machine Learning Research, vol. 168. PMLR, June 2022, pp. 931–942. [Online]. Available: https://proceedings.mlr.press/v168/viereck22a.html

[12] A. Meduri, P. Shah, J. Viereck, M. Khadiv, I. Havoutis, and L. Righetti, "Biconmp: A nonlinear model predictive control framework for whole body motion planning," *IEEE Transactions on Robotics*, p. 1–18, 2023. [Online]. Available: https://arxiv.org/abs/2201.07601

[13] S. Gai, S. Lyu, H. Zhang, and D. Wang, "Continual reinforcement learning for quadruped robot locomotion," *Entropy*, vol. 26, no. 1, p. 93, 2024.

[14] Z.-Y. Fu, D.-C. Zhan, X.-C. Li, and Y.-X. Lu, "Automatic successive reinforcement learning with multiple auxiliary rewards," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 2336–2342.

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

[16] J. Wang, C. Hu, and Y. Zhu, "Cpg-based hierarchical locomotion control for modular quadrupedal robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7193–7200, 2021.

[17] Y. Kim, B. Son, and D. Lee, "Learning multiple gaits of quadruped robot using hierarchical reinforcement learning," 2021.

[18] W. Tan, X. Fang, W. Zhang, R. Song, T. Chen, Y. Zheng, and Y. Li, "A hierarchical framework for quadruped omnidirectional locomotion based on reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, 2023.

[19] J. Ren, Y. Dai, B. Liu, P. Xie, and G. Wang, "Hierarchical vision navigation system for quadruped robots with foothold adaptation learning," *Sensors*, vol. 23, no. 11, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/11/5194

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

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

[22] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard, "Cat: Constraints as terminations for legged locomotion reinforcement learning," 2024.

[23] M. Shafiee, G. Bellegarda, and A. Ijspeert, "Manyquadrupeds: Learning a single locomotion policy for diverse quadruped robots," 2024.

[24] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[25] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 2019.

[26] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[27] 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.

[28] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[29] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot," in *Field and Service Robotics: Results of the 12th International Conference*. Springer, 2021, pp. 247–260.

[30] M. Hutter, C. Gehring, M. A. Höpflinger, M. Blösch, and R. Siegwart, "Toward combining speed, efficiency, versatility, and robustness in an autonomous quadruped," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1427–1440, 2014.

[31] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021.

[32] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8484–8490.

[33] J. He, J. Shao, G. Sun, and X. Shao, "Survey of quadruped robots coping strategies in complex situations," *Electronics*, vol. 8, no. 12, p. 1414, 2019.

[34] P. Biswal and P. K. Mohanty, "Development of quadruped walking robots: A review," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 2017–2031, 2021.

[35] H. Taheri and N. Mozayani, "A study on quadruped mobile robots," *Mechanism and Machine Theory*, vol. 190, p. 105448, 2023.

[36] P.-A. Léziart, T. Corbères, T. Flayols, S. Tonneau, N. Mansard, and P. Souères, "Improved control scheme for the solo quadruped and experimental comparison of model predictive controllers," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9945–9952, 2022.

[37] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete, "Optimization-based control for dynamic legged robots," 2022.

[38] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2464–2470.

[39] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "TAMOLS: Terrain-aware motion optimization for legged systems," vol. 38, 2022.

[40] S. Chamorro, V. Klemm, M. d. l. I. Valls, C. Pal, and R. Siegwart, "Reinforcement learning for blind stair climbing with legged and wheeled-legged robots," *arXiv preprint arXiv:2402.06143*, 2024.

[41] 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, no. 62, p. eabk2822, 2022.

[42] G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 10 364–10 370.

[43] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," 2018.

[44] X. Han and M. Zhao, "Learning quadrupedal high-speed running on uneven terrain," *Biomimetics*, vol. 9, no. 1, p. 37, 2024.

[45] J. Qi, H. Gao, H. Su, L. Han, B. Su, M. Huo, H. Yu, and Z. Deng, "Reinforcement learning-based stable jump control method for asteroid-exploration quadruped robots," *Aerospace Science and Technology*, vol. 142, p. 108689, 2023.

[46] C. Zhang, J. Sheng, T. Li, H. Zhang, C. Zhou, Q. Zhu, R. Zhao, Y. Zhang, and L. Han, "Learning highly dynamic behaviors for quadrupedal robots," *arXiv preprint arXiv:2402.13473*, 2024.

[47] T. Miki, J. Lee, L. Wellhausen, and M. Hutter, "Learning to walk in confined spaces using 3d representation," 2024.

[48] Y. Ji, G. B. Margolis, and P. Agrawal, "Dribblebot: Dynamic legged manipulation in the wild," 2023.

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

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

[51] Y. Song, S. Kim, and D. Scaramuzza, "Learning quadruped locomotion using differentiable simulation," 2024.

[52] S. Devlin, D. Kudenko, and M. Grzes, "An empirical study of potential-based reward shaping and advice in complex, multi-agent systems," *Adv. Complex Syst.*, vol. 14, pp. 251–278, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:7654948

[53] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Villaseñor-Pineda, "Dynamic reward shaping: Training a robot by voice," in *Advances in Artificial Intelligence – IBERAMIA 2010*, A. Kuri-Morales and G. R. Simari, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 483–492.

[54] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Reward machines: Exploiting reward function structure in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 73, p. 173–208, Jan. 2022. [Online]. Available: http://dx.doi.org/10.1613/jair.1.12440

[55] M. Grzes and D. Kudenko, "Plan-based reward shaping for reinforcement learning," in *2008 4th International IEEE Conference Intelligent Systems*, vol. 2, 2008, pp. 10–22–10–29.

[56] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma, "Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics," 2023.

[57] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Reward shaping with language models for reinforcement learning," 2024.

[58] S. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, 2012, pp. 433–440.

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

[60] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," 2022.

[61] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," 2020.

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

[63] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, ser. SCA '17. ACM, July 2017. [Online]. Available: http://dx.doi.org/10.1145/3099564.3099567

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

[65] 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, no. 26, Jan. 2019. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.aau5872

[66] M. Aractingi, P.-A. Léziart, T. Flayols, J. Perez, T. Silander, and P. Souères, "Controlling the solo12 quadruped robot with deep reinforcement learning," *Scientific Reports*, vol. 13, no. 1, July 2023. [Online]. Available: http://dx.doi.org/10.1038/s41598-023-38259-7

[67] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[68] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[69] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *International conference on machine learning*. PMLR, 2018, pp. 1515–1528.

[70] X. Zhang, Y. Wu, H. Wang, F. Iida, and L. Wang, "Adaptive locomotion learning for quadruped robots by combining DRL with a cosine oscillator based rhythm controller," *Applied Sciences*, vol. 13, no. 19, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/19/11045

[71] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne, "Allsteps: curriculum-driven learning of stepping stone skills," in *Computer Graphics Forum*, vol. 39, no. 8. Wiley Online Library, 2020, pp. 213–224.