DTC: Deep Tracking Control

Fabian Jenelten,[1]* Junzhe He,[1] Farbod Farshidian,[2] and Marco Hutter[1]

[1] *Robotic Systems Lab, ETH Zurich, 8092 Zurich, Switzerland.*
[2] *Currently at Boston Dynamics AI Institute, 145 Broadway, Cambridge MA, USA*
[*] *Corresponding author: fabian.jenelten@ethz.ch*

This is the accepted version of Science Robotics Vol. 9, Issue 86, eadh5401 (2024)
DOI: 0.1126/scirobotics.adh5401

Legged locomotion is a complex control problem that requires both accuracy and robustness to cope with real-world challenges. Legged systems have traditionally been controlled using trajectory optimization with inverse dynamics. Such hierarchical model-based methods are appealing due to intuitive cost function tuning, accurate planning, generalization, and most importantly, the insightful understanding gained from more than one decade of extensive research. However, model mismatch and violation of assumptions are common sources of faulty operation. Simulation-based reinforcement learning, on the other hand, results in locomotion policies with unprecedented robustness and recovery skills. Yet, all learning algorithms struggle with sparse rewards emerging from environments where valid footholds are rare, such as gaps or stepping stones. In this work, we propose a hybrid control architecture that combines the advantages of both worlds to simultaneously achieve greater robustness, foot-placement accuracy, and terrain generalization. Our approach utilizes a model-based planner to roll out a reference motion during training. A deep neural network policy is trained in simulation, aiming to track the optimized footholds. We evaluate the accuracy of our locomotion pipeline on sparse terrains, where pure data-driven methods are prone to fail. Furthermore, we demonstrate superior robustness in the presence of slippery or deformable ground when compared to model-based counterparts. Finally, we show that our proposed tracking controller generalizes across different trajectory optimization methods not seen during training. In conclusion, our work unites the predictive capabilities and optimality guarantees of online planning with the inherent robustness attributed to offline learning.

INTRODUCTION

Trajectory optimization (TO) is a commonly deployed instance of optimal control for designing motions of legged systems and has a long history of successful applications in rough environments since the early 2010s [1, 2]. These methods require a model of the robot's kinematics and dynamics during runtime, along with a parametrization of the terrain. Until recently, most approaches have used simple models such as single rigid body [3] or inverted pendulum dynamics [4, 5], or have ignored the dynamic effects altogether [6]. Research has shifted towards more complex formulations, including centroidal [7] or full-body dynamics [8]. The resulting trajectories are tracked by a whole-body control (WBC) module, which operates at the control frequency and utilizes full-body dynamics [9]. Despite the diversity and agility of the resulting motions, there remains a considerable gap between simulation and reality due to unrealistic assumptions. Most problematic assumptions include perfect state estimation, occlusion-free vision, known contact states, zero foot-slip, and perfect realization of the planned motions. Sophisticated hand-engineered state machines are required to detect and respond to various special cases not accounted for in the modeling process. Nevertheless, highly dynamic jumping maneuvers performed by Boston Dynamics' bipedal robot Atlas demonstrate the potential power of TO.

Reinforcement learning (RL) has emerged as a powerful tool in recent years for synthesizing robust legged locomotion. Unlike model-based control, RL does not rely on explicit models. Instead, behaviors are learned, most often in simulation, through random interactions of agents with the environment. The result is a closed-loop control policy, typically represented by a deep neural network, that maps raw observations to actions. Handcrafted state-machines become obsolete because all relevant corner cases are eventually visited during training. End-to-end policies, trained from user commands to joint target positions, have been deployed successfully on quadrupedal robots such as ANYmal [10, 11]. More advanced teacher-student structures have substantially improved the robustness, enabling legged robots to overcome obstacles through touch [12] and perception [13]. Although locomotion across gaps and stepping stones is theoretically possible, good exploration strategies are required to learn from the emerging sparse reward signals. So far, these terrains could only be handled by specialized policies, which intentionally overfit to one particular scenario [14] or a selection of similar terrain types [15–18]. Despite promising results, distilling a unifying locomotion policy may be difficult and has only been shown with limited success [19].

Some of the shortcomings that appear in RL can be mitigated using optimization-based methods. While the problem of sparse gradients still exists, two important advantages can be exploited: First, cost-function and constraint gradients can be computed with a small

Fig. 1. Robust and precise locomotion in various indoor and outdoor environments. The marriage of model-free and model-based control allows legged robots to be deployed in environments where steppable contact surfaces are sparse (bottom left) and environmental uncertainties are high (top right).

number of samples. Second, poor local optima can be avoided by pre-computing footholds [5, 8], pre-segmenting the terrain into steppable areas [7, 20], or by smoothing out the entire gradient landscape [21]. Another advantage of TO is its ability to plan actions ahead and predict future interactions with the environment. If model assumptions are generic enough, this allows for great generalization across diverse terrain geometries [7, 21].

The sparse gradient problem has been addressed extensively in the learning community. A notable line of research has focused on learning a specific task while imitating expert behavior. The expert provides a direct demonstration for solving the task [22, 23], or is used to impose a style while discovering the task [24–26]. These approaches require collecting expert data, commonly done offline, either through re-targeted motion capture data [24–26] or a TO technique [22, 23]. The reward function can now be formulated to be dense, meaning that agents can collect non-trivial rewards even if they do not initially solve the task. Nonetheless, the goal is not to preserve the expert's accuracy but rather to lower the sample and reward complexity by leveraging existing knowledge.

To further decrease the gap between the expert and the policy performance, we speculate that the latter should have insight into the expert's intentions. This requires online generation of expert data, which can be conveniently achieved using any model-based controller. Unfortunately, rolling out trajectories is often orders of magnitude more expensive than a complete learning iteration. To circumvent this problem, one possible alternative is to approximate the expert with a generative model, for instance, by sampling footholds from a uniform distribution [15, 16], or from a neural network [17, 27, 28]. However, for the former group, it might be challenging to capture the distribution of an actual model-based controller, while the latter group still does not solve the exploration problem itself.

In this work, we propose to guide exploration through the solution of TO. As such data will be available both on- and offline, we refer to it as "reference" and not expert motion. We utilize a hierarchical structure introduced in deep loco [28], where a high-level planner proposes footholds at a lower rate, and a low-level controller follows the footholds at a higher rate. Instead of using a neural network to generate the foothold plan, we leverage TO. Moreover, we do not only use the target footholds as an indicator for a rough high-level direction but as a demonstration of optimal foot placement.

The idea of combining model-based and model-free approaches is not new in the literature. For instance, supervised [29] and unsupervised [30, 31] learning has been used to warm-start nonlinear solvers. RL has been used to imitate [22, 23] or correct [32] motions obtained by solving TO problems. Conversely, model-based methods have been used to check the feasibility of learned high-level commands [27] or to track learned acceleration profiles [33]. Compared to [32], we do not learn corrective joint torques around an existing WBC, but instead, learn the mapping from reference signals to joint positions in an end-to-end fashion.

To generate the reference data, we rely on an efficient TO method called terrain-aware motion generation for legged systems (TAMOLS) [21]. It optimizes over footholds and base pose simultaneously, thereby enabling the robot to operate at its kinematic limits. We let the policy observe only a small subset of the solution, namely planar footholds, desired joint positions, and the contact schedule. We found that these observations are more robust under the common pitfalls of model-based control, while still providing enough information to solve the locomotion task. In addition, we limit computational costs arising from solving the optimization problems by utilizing a variable update rate. During deployment, the optimizer runs at the fastest possible rate to account for model uncertainties and external disturbances.

Our approach incorporates elements introduced in [14], such as time-based rewards and position-based goal tracking. However, we reward desired foothold positions at planned touch-down instead of rewarding a desired base pose at an arbitrarily chosen time. Finally, we use an asymmetric actor-critic structure similar to [22], where we provide privileged ground truth information to the value function and noisified measurements to the network policy.

We trained more than 4000 robots in parallel for two weeks on challenging ground covering a surface area of more than $76000\,\mathrm{m}^2$. Throughout the entire training process, we generated and learned from about 23 years of optimized trajectories. The combination of offline training and online re-planing results in accurate, agile, and robust locomotion. As showcased in Fig. 1 and movie 1, with our hybrid control pipeline, ANYmal [34] can skillfully traverse parkours with high precision, and confidently overcome uncertain environments with high robustness. Without the need for any post-training, the tracking policy can be deployed zero-shot with different TO methods at different update rates. Moreover, movie 2 demonstrates successful deployment in search-and-rescue scenarios, which demand both accurate foot placement and robust recovery skills. The contributions of our work are therefore twofold: Firstly, we enable the deployment of model-based planners in rough and uncertain real-world environments. Secondly, we create a single unifying locomotion policy that generalizes beyond the limitations imposed by state-of-the-art RL methods.

RESULTS

In order to evaluate the effectiveness of our proposed pipeline, hereby referred to as Deep Tracking Control (DTC), we compared it with four different approaches: two model-based controllers, TAMOLS [21] and a nonlinear model predictive control (MPC) method presented in [7], and two data-driven methods, as introduced in [13] and [11]. We refer to those as baseline-to-1 (TAMOLS), baseline-to-2 (MPC), baseline-rl-1 (teacher/student policy), and baseline-rl-2 (RL policy), respectively. These baselines mark the state-of-the-art in MPC and RL prior to this work and they have been tested and deployed under various conditions. If not noted differently, all experiments were conducted in the real world.

Evaluation of Robustness

We conducted three experiments to evaluate the robustness of our hybrid control pipeline. The intent is to demonstrate survival skills on slippery ground, and recovery reflexes when visual data is not consistent with proprioception or is absent altogether. We rebuilt harsh environments that are likely to be encountered on sites of natural disasters, where debris might further break down when stepped onto, and construction sites, where oil patches create slippery surfaces.

In the first experiment, we placed a rectangular cover plate with an area of $0.78 \times 1.19\,\mathrm{m}^2$ on top of a box with the same length and width, and height $0.37\,\mathrm{m}$ (Fig. 2 A). The cover plate was shifted to the front, half of the box's length. ANYmal was then steered over the cover plate, which pitched down as soon as its center of mass passed beyond the edge of the box. Facing only forward and backward, the plate's movement was not detected through the depth cameras, and could only be perceived through proprioceptive sensors. Despite the error between map and odometry reaching up to $0.4\,\mathrm{m}$, the robot managed to successfully balance itself. This experiment was repeated three times with consistent outcomes.

In our second experiment (Fig. 2 B) we created an obstacle parkour with challenging physical properties. A large wooden box with a slopped front face was placed next to a wet and slippery whiteboard. We increased the difficulty by placing a soft foam box in front, and a rolling transport cart on top of the wooden box. The robot was commanded to walk over the objects with random reference velocities for

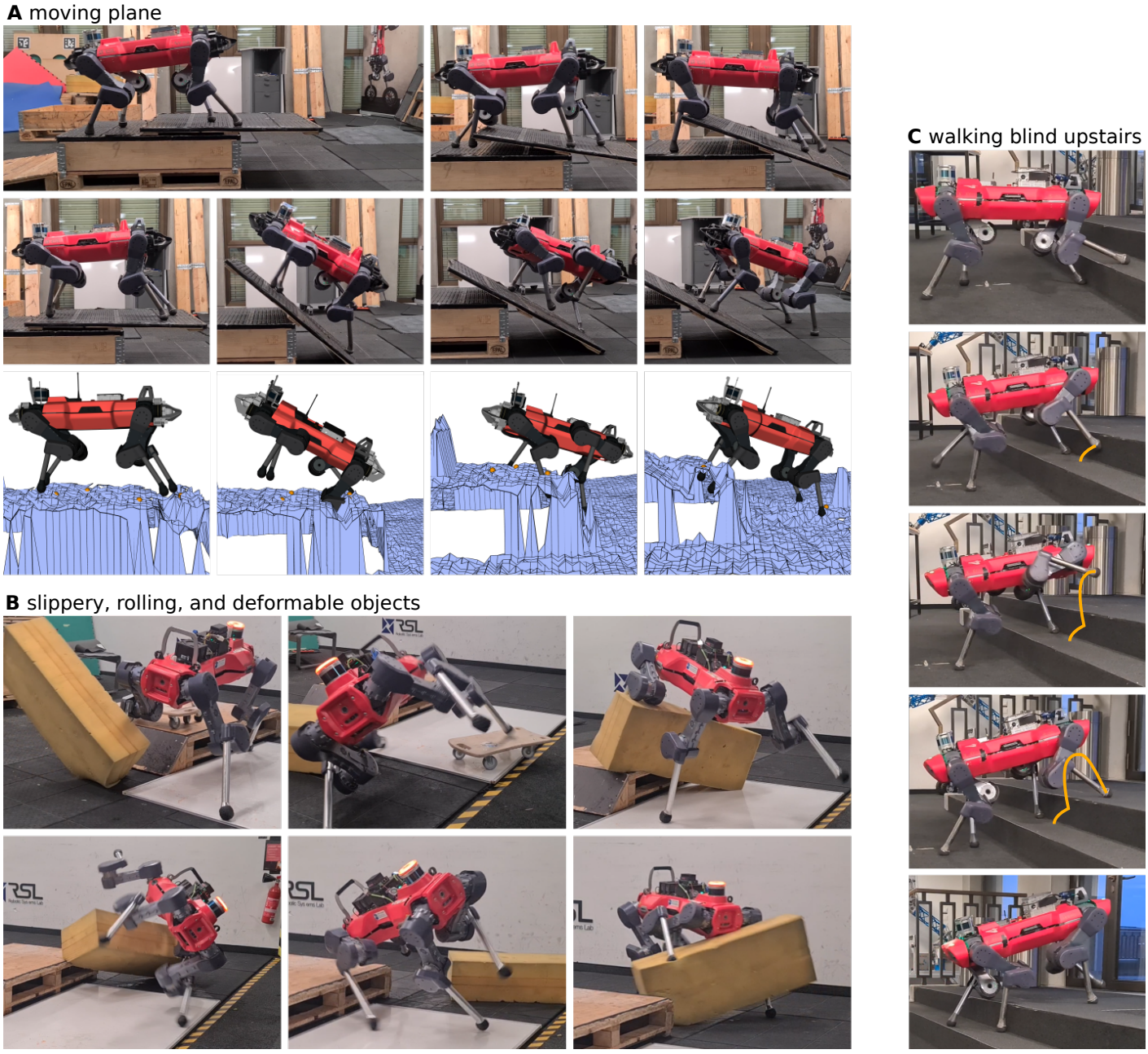

Fig. 2. Evaluation of robustness. **(A)** ANYmal walks along a loose cover plate that eventually pitches forward (left to right, top to bottom). The third row shows ANYmal's perception of the surroundings during the transition and recovery phase. **(B)** The snapshots are taken at critical time instances when walking on slippery ground, just before complete recovery. **(C)** ANYmal climbs upstairs with disabled perception (top to bottom). The collision of the right-front end-effector with the stair tread triggers a swing reflex, visualized in orange.

approximately 45 seconds, after which the objects were redistributed to their original locations to account for any potential displacement. This experiment was repeated five times. Despite not being trained on movable or deforming obstacles, the robot demonstrated its recovery skills in all five trials without any falls.

The tracking policy was trained with perceptive feedback, meaning that the policy and the motion planner had partial or complete insight into the local geometrical landscape. Nevertheless, the locomotion policy was still capable of overcoming many obstacles completely blind. To simulate a scenario with damaged depth sensors, we let ANYmal blindly walk over a stair with two treads, each 0.18 m high and 0.29 m wide (Fig. 2 C). The experiment was repeated three times up and down, with an increasing heading velocity selected from $\{\pm 0.5, \pm 0.75, \pm 1.0\}$ m/s. In some cases, a stair tread was higher than the swing motion of a foot. Thanks to a learned swing reflex, the stair set could be successfully cleared in all trials. We note that the same stair set was passed by a blindfolded version of baseline-rl-1 [13], which was trained in a complex teacher/student environment. In contrast, our method relies on an asymmetric actor/critics structure, achieving a similar level of robustness. Accompanying video clips can be found in the supplementary movie S1.

Evaluation of Accuracy

We demonstrate the precision of foothold tracking by devising a complex motion that requires the robot to perform a turn-in-place maneuver on a small surface area of $0.94 \times 0.44\,\mathrm{m}^2$. The robot was commanded to walk up a slope onto a narrow table, then to execute a complete 360 deg turn, and finally to descend onto a pallet. Some snapshots of the experiment are provided in Fig. 3 A, whereas the full video is contained in movie S2.

To evaluate the quality of the foothold tracking, we collected data while ANYmal walked on flat ground. Each experiment lasted for approximately 20 s and was repeated with eight different heading velocities selected from $\{\pm 1.0, \pm 0.8, \pm 0.6, \pm 0.4\}$ m/s. We measured the tracking error as the smallest horizontal distance between a foot and its associated foothold during a stance phase. As shown in Fig. 3 B, the footholds could be tracked with very high precision of 2.3 cm and standard deviation 0.48 cm when averaged over the broad spectrum of heading velocity commands.

Deployment with MPC

The maximum height that DTC in combination with TAMOLS can reliably overcome is about 0.40 m. The policy might hesitate to climb up taller objects due to the risk of potential knee joint collisions with the environment. This limitation is inherent to the chosen TO method, which only considers simplified kinematic constraints. We, therefore, deployed DTC with the planner of baseline-to-2, which takes into account the full kinematics of the system. To allow for zero-shot generalization, we implemented the same trotting gait as experienced during training. With this enhanced setup, ANYMal could climb up a box of height of 0.48 m. This is 50 % higher than what baseline-rl-1 could climb up, and 380 % more than what was reported for baseline-rl-2. The box climbing experiment was successfully repeated five times. The results are shown in movie S2, and for one selected trial in Fig. 3 D. Furthermore, we measured the tracking error on flat ground. Despite the wider stance configuration of baseline-to-2, the error was found to be only 0.03 m on average (Fig. 3 C).

The above two results seem to be surprising at first glance but are easy to explain when considering the observation space and the training environment. Although the base-pose trajectory is considerably more detailed for baseline-to-2 due to frequency-loop shaping and increased system complexity, the foothold patterns are nevertheless quite similar. Thus, good generalization is facilitated by the specific choice of observations, which hides the optimized base pose from the policy. Some terrains within the training environment can be seen as a combination of gaps and boxes, where each box is surrounded by a gap. During training, TAMOLS placed the footholds sufficiently far away from the box to avoid stepping into the gap. This allowed the policy to learn climbing maneuvers without knee joint collisions. Baseline-to-2, being aware of the spatial coordinates of the knees, naturally produces a similar foothold pattern, even in the absence of the gap.

Benchmark Against Model-Based Control

TO was proven to be effective in solving complex locomotion tasks in simulation, such as the parkour shown in Fig. 4 A. This parkour has been successfully traversed by ANYmal using baseline-to-1 and baseline-to-2, while it was found to be non-traversable for baseline-rl-1 and baseline-rl-2 [7]. With our proposed approach, ANYmal could cross the same obstacle parkour in simulation back and forth at a speed of 1 m/s, which was 20 % faster than baseline-to-1. The corresponding video clip can be found in movie S3.

Model-based controllers react sensitively to violation of model assumptions, which hinders applications in real-world scenarios, where, for instance, uncertainties in friction coefficients, contact states, and visual perception may be large. This issue is exemplified in Fig. 4 B, where baseline-to-1 was used to guide ANYmal over a flat floor with an invisible gap. When the right front foot stepped onto the trap, the planned and executed motions deviated from each other. This triggered a sequence of heuristic recovery strategies. For large mismatches, however, such scripted reflexes were not effective, and resulted in failure. DTC uses the same high-level planner but incorporates learned recovery and reflex skills. This allowed the robot to successfully navigate through the trap. The robustness is rooted in the ability to ignore both perception and reference motion while relying only on proprioception. Such behavior was learned in simulation by experiencing simulated map drift. The experiment was repeated five times with baseline-to-1, five times with baseline-to-2, and five times with our method, consistently leading to similar results. The video clips corresponding to the above experiments can be found in movie S3. The movie is further enriched with a comparison of baseline-to-2 against DTC on soft materials, which impose very similar challenges.

Benchmark Against RL Control

Although RL policies are known for their robustness, they may struggle in environments with limited interaction points. We demonstrate typical failure cases in two experiments utilizing baseline-rl-1. In the first experiment (Fig. 5 A), ANYmal was tasked to cross a small gap of 0.1 m with a reference heading velocity of 0.2 m/s. The model-free controller did not avoid the gap, and thus could not reach the other side of the platform. In the second experiment, we connected two elevated boxes with a 1.0 m-long beam of height 0.2 m (Fig. 5 B). The robot was commanded to walk from the left to the right box but failed to make use of the beam.

In comparison, our hybrid policy achieved a 100 % success rate for the same gap size over ten repetitions. To further demonstrate the locomotion skills of DTC, we made the experiments more challenging. We replaced the small gap with four larger gaps, each 0.6 m wide and evenly distributed along the path (Fig. 5 C). Similarly, we increased the length of the beam to a total of 1.8 m (Fig. 5 D). Despite the increased difficulty, our approach maintained a 100 % success rate across four repetitions of each experiment. Video clips of those experiments can be found in movie S4.

By using a specialized policy, ANYmal crossed a 0.6 m wide gap within a pre-mapped environment [14]. Most notably, our locomotion controller, not being specialized nor fine-tuned for this terrain type,

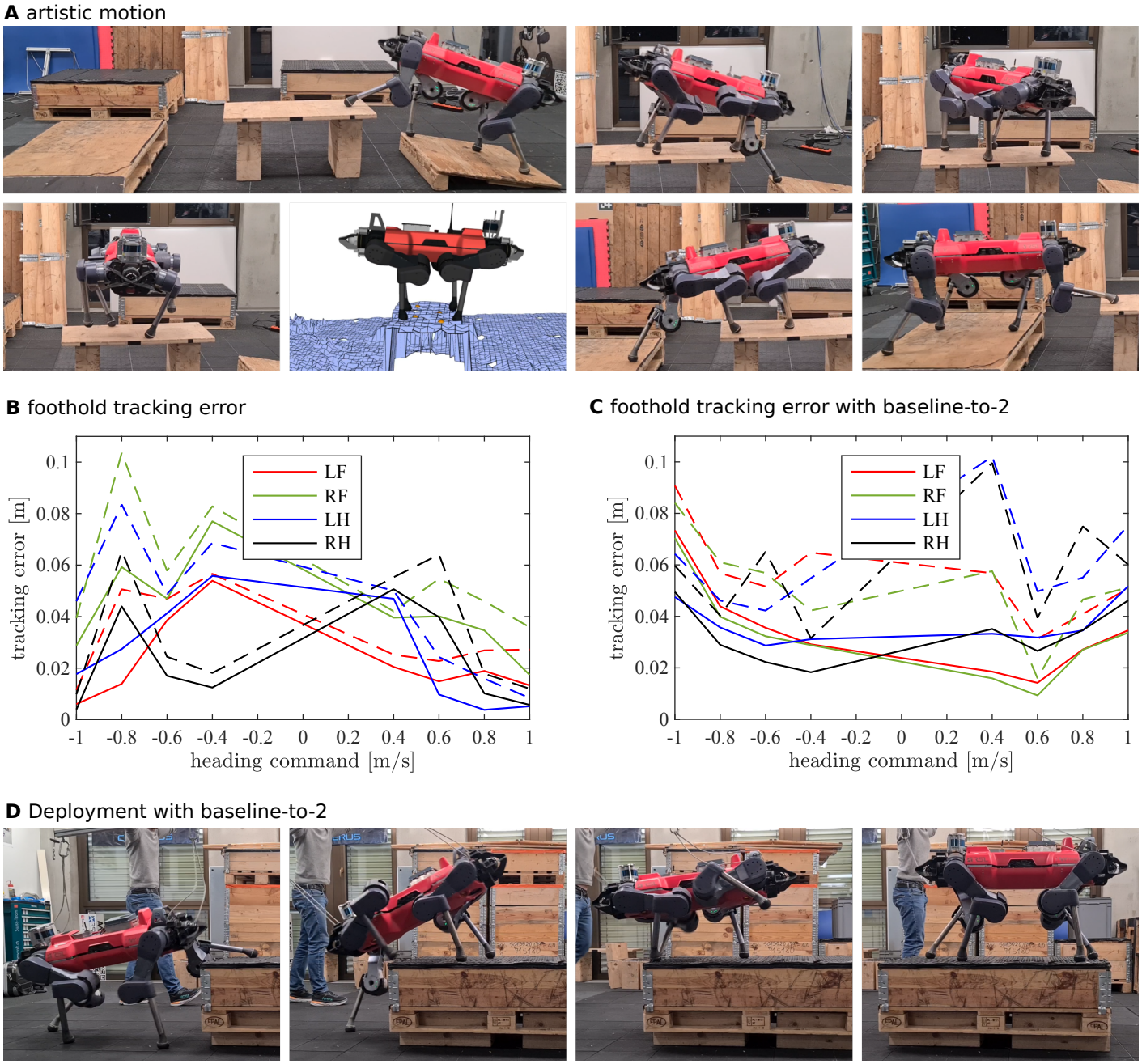

Fig. 3. Evaluation of tracking performance. **(A)** ANYmal climbs up a narrow table, turns, and descends back down to a box. The second image in the second row shows the robot's perception of the environment. **(B)** Euclidean norm of the planar foothold error, averaged over 20 s of operation using a constant heading velocity. The solid/dashed curves represent the average/maximum tracking errors. **(C)** Same representation as in (B), but the data was collected with baseline-to-2. **(D)** DTC deployed with baseline-to-2, enabling ANYMal to climb up a box of 0.48 m.

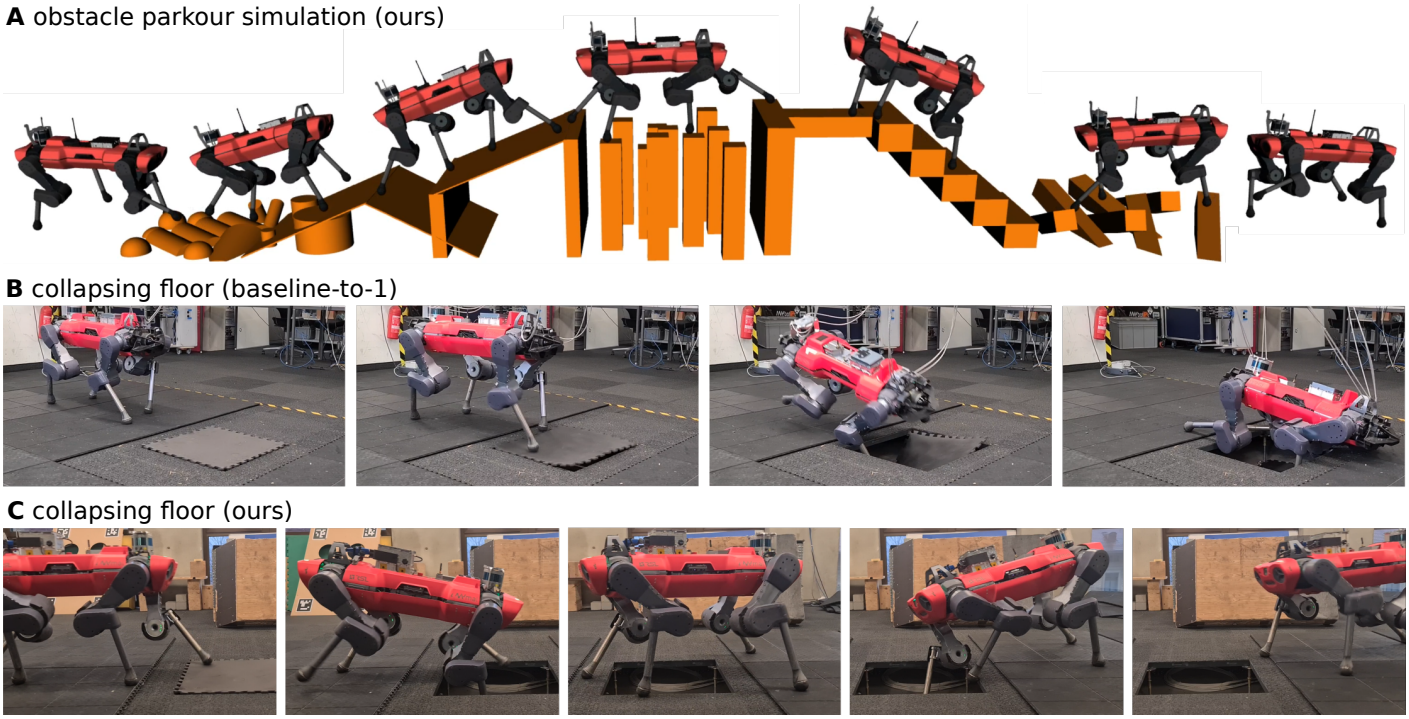

Fig. 4. Benchmarking against model-based control. **(A)** DTC successfully traverses an obstacle parkour (left to right) in simulation with a heading velocity of 1 m/s. **(B)** Baseline-to-1 falls after stepping into a gap hidden from the perception (left to right). **(C)** ANYmal successfully overcomes a trapped floor using our hybrid control architecture (left to right).

crossed a sequence of four gaps with the same width, whilst, relying on online generated maps only.

The limitations of baseline-rl-1 were previously demonstrated [7] on the obstacle parkour of Fig. 4 A, showing its inability to cross the stepping stones. We showcase the generality of our proposed control framework by conducting three experiments on stepping stones in the real world, each with an increased level of difficulty. The first experiment (Fig. 6 A) required ANYmal traversing a field of equally sized stepping stones, providing a contact surface of $0.2 \times 0.2\,\mathrm{m}^2$ each. The robot passed the 2.0 m long field 10 times. Despite the varying heading velocity commands, the robot accurately hit the correct stepping stones as indicated by the solution of the TO. For the second experiment (Fig. 6 B), we increased the height of two randomly selected stones. The parkour was successfully crossed four out of four times. In the final experiment (Fig. 6 C), we distributed three elevated platforms a, b, and c, connected by loose wooden blocks of sizes $0.31 \times 0.2 \times 0.2\,\mathrm{m}^3$ and $0.51 \times 0.2 \times 0.2\,\mathrm{m}^3$. This environment posed considerable challenges as the blocks may move and flip over when stepped on. Following the path $a \to b \to a \to b \to c \to a$, the robot missed only one stepping stone, which, however, did not lead to failure. Video clips of the stepping stones experiments are provided in movie S5.

Simulation-Based Ablation Study

During training, we computed a solution to the TO problem after variable time intervals, but mainly after each foot touch-down. Although such a throttled rate greatly reduced computational costs, it also leads to poor reactive behavior in the presence of quickly changing external disturbances, dynamic obstacles, or map occlusion. Moreover, the optimizer was updated using privileged observations, whereas, in reality, the optimizer is subject to elevation map drift, wrongly estimated friction coefficients, and unpredicted external forces. To compensate for such modeling errors, we deploy the optimizer in MPC-fashion. We investigated the locomotion performance as a function of the optimizer update rate. Using the experimental setup outlined in the supplementary methods (section "Experimental Setup for Evaluation of Optimizer Rate"), we collected a total of six days of data in simulation. A robot was deemed "successful" if it could walk from the center to the border of its assigned terrain patch, "failed" if its torso made contact with the environment within its patch, and "stuck" otherwise. We report success and failure rates in Fig. 7 A. Accordingly, when increasing the update rate from 1 Hz to 50 Hz, the failure rate dropped by 7.11 % whereas the success rate increased by 4.25 %.

In the second set of experiments, we compared our approach against baseline-rl-2 as well as against the same policy trained within our training environment. We refer to the emerging policy as baseline-rl-3. More details regarding the experimental setup can be found in the supplementary methods (section "Experimental Setup for Performance Evaluation"). As depicted in Fig. 7 B i, our approach exhibited a substantially higher success rate than baseline-rl-2. By learning on the same terrains, baseline-rl-3 could catch up but still did not match our performance in terms of success rate. The difference mainly originates from the fact that the retrained baseline still failed to solve sparse-structured terrains. To highlight this observation, we evaluated the performance on four terrain types with sparse ("stepping stones", "beams", "gaps", and "pallets"), and on four types with dense stepping locations ("stairs", "pit", "rough slope", and "rings"). On all considered terrain types, our approach outperformed baseline-rl-2 by a huge margin (Fig. 7 B ii), thereby demonstrating that learned locomotion generally does not extrapolate well to unseen scenarios. We performed equally well as baseline-rl-3 on dense terrains, but scored notably higher on sparse-structured terrains. This result suggests that the proposed approach itself was effective and that favorable locomotion skills were not encouraged by the specific training environment.

In an additional experiment, we investigated the effects of erroneous predictions of the high-level planner on locomotion performance. We did so by adding a drift value to the elevation map, sampled uniformly

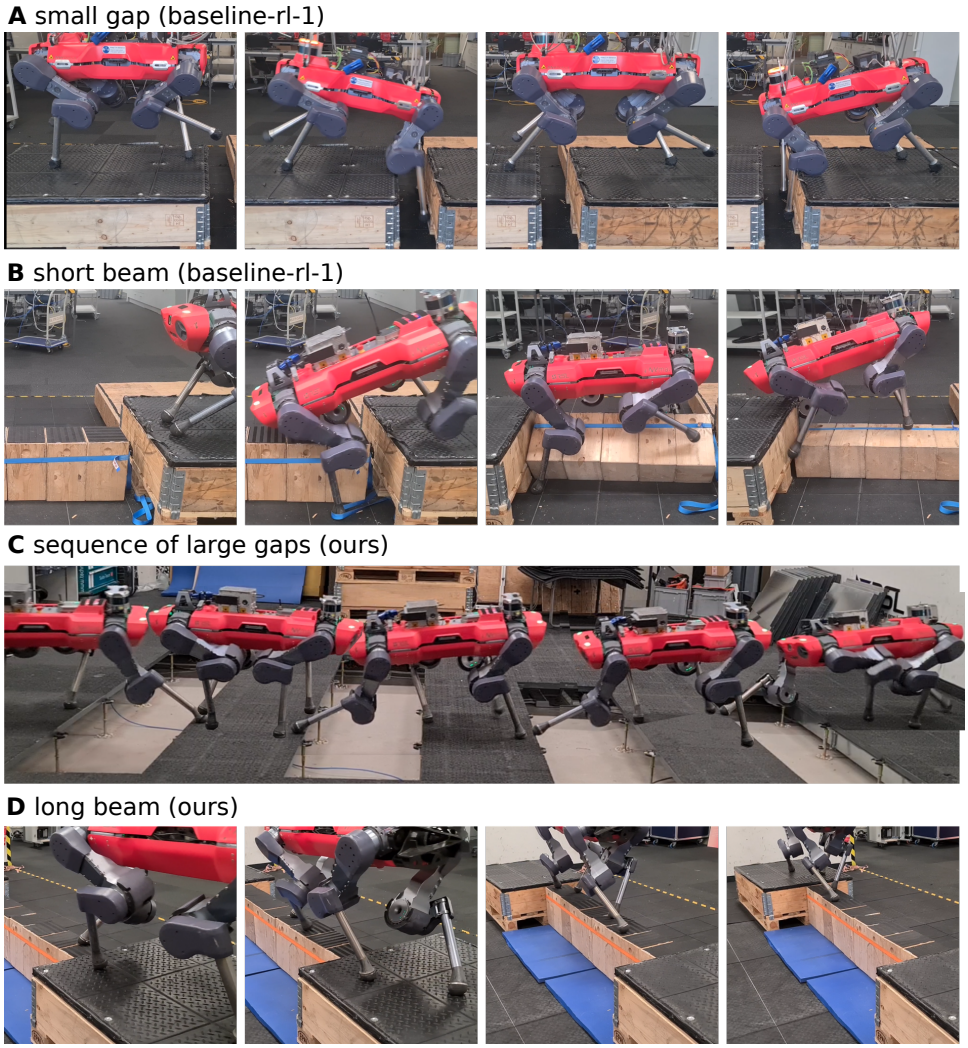

Fig. 5. Benchmarking against reinforcement learning. **(A)** Baseline-rl-1 attempts to cross a small gap. ANYmal initially manages to recover from miss-stepping with its front legs but subsequently gets stuck as its hind legs fall inside the gap. **(B)** Using baseline-rl-1, the robot stumbles along a narrow beam. **(C)** With DTC, the robot can pass four consecutive large gaps (left to right) without getting stuck or falling. **(D)** ANYmal is crossing a long beam using the proposed control framework.

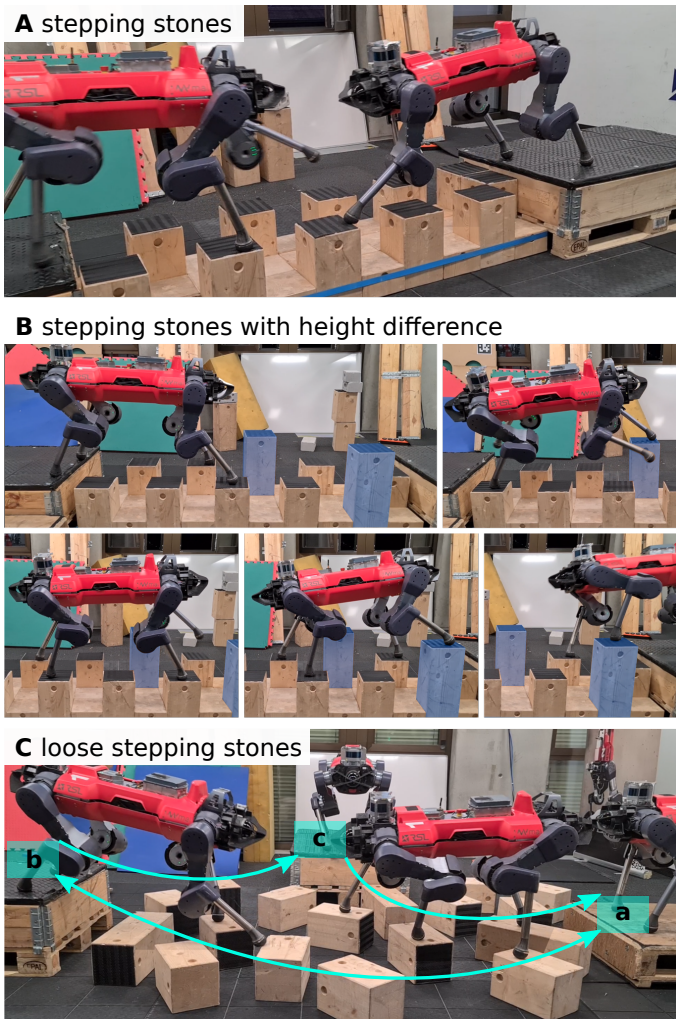

Fig. 6. Evaluation of the locomotion performance on stepping stones. **(A)** ANYmal reliably crosses a field of flat stepping stones (left to right). **(B)** The robot crosses stepping stones of varying heights (left to right). The two tall blocks are highlighted in blue. **(C)** ANYmal navigates through a field of loosely connected stepping stones, following the path $a \rightarrow b \rightarrow a \rightarrow b \rightarrow c \rightarrow a$.

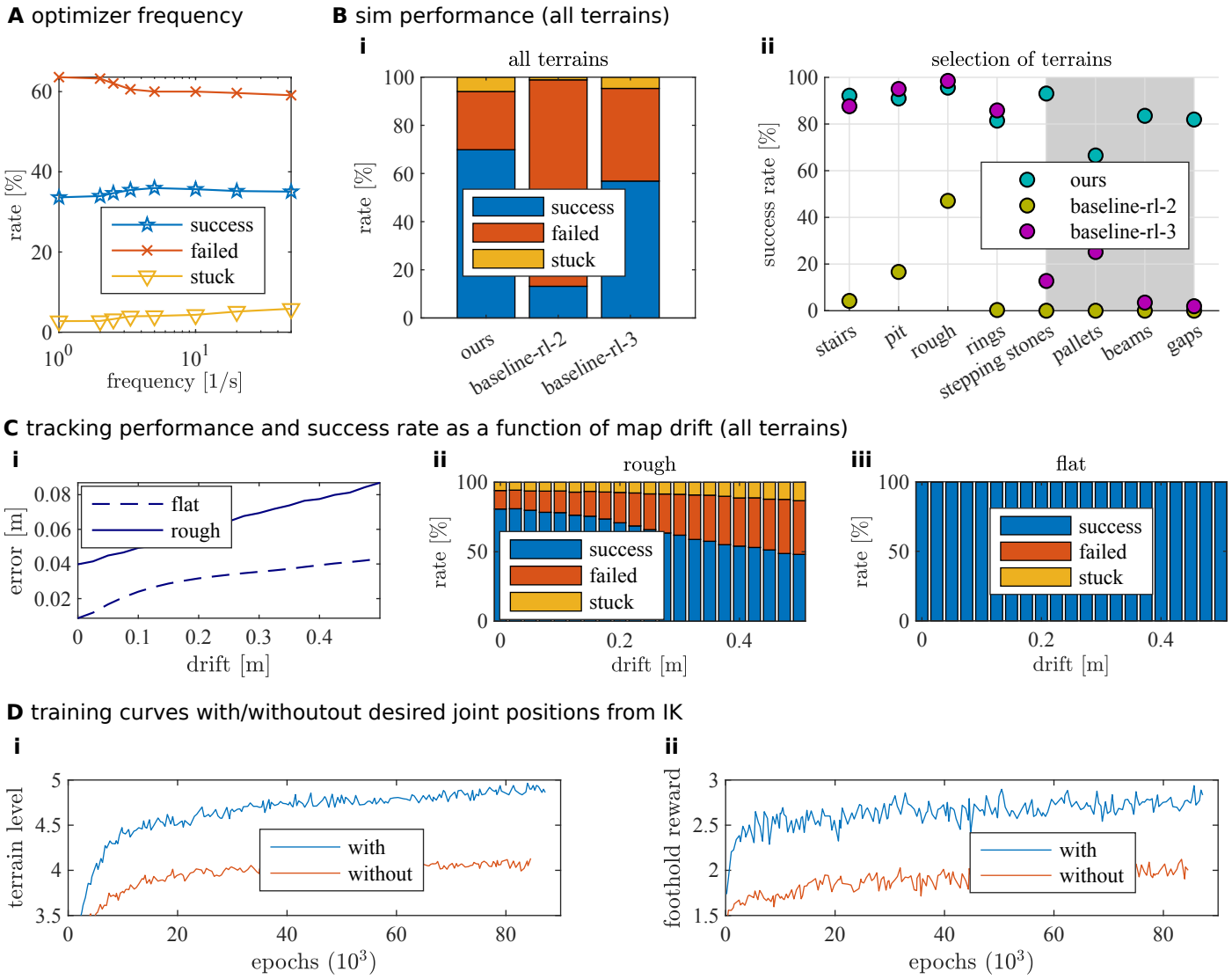

Fig. 7. Simulation results and ablation studies. **(A)** Success and failure rates of DTC, recorded for different update rates of the optimizer. The upper limit of 50 Hz is imposed by the policy frequency. **(B)** Comparison against baseline policies. **(i)** Evaluation on all 120 terrains. **(ii)** Evaluation on terrains where valid footholds are dense (white background) and sparse (gray background). **(C)** Influence of elevation map drift on the locomotion performance, quantified by tracking error **(i)**, success rate on rough **(ii)**, and on flat ground **(ii)**. **(D)** Average terrain level **(i)** and average foothold reward **(ii)** scored during training.

from the interval $\in (0, 0.5)$ m. Contrary to training, the motion was optimized over the perturbed height map. Other changes to the experimental setup are described in the supplementary methods (section "Experimental Setup for Performance Evaluation under Drift"). As visualized in Fig. 7 C, we collected tracking error, success, and failure rates with simulated drift on flat and rough ground. The tracking error grew mostly linearly with the drift value (Fig. 7 C i). On rough terrains, the success rate remained constant for drift values smaller than 0.1 m, and decreased linearly for larger values (Fig. 7 C ii). On the other hand, success and failure rates were not affected by drift on flat ground (Fig. 7 C iii).

We found that providing joint positions computed for the upcoming touch-down event greatly improved convergence time and foothold tracking performance. This signal encodes the foothold location in joint space, thus, providing a useful hint for foothold tracking. It also simplifies the learning process, as the network is no longer required to implicitly learn the inverse kinematics (IK). Evidence for our claims is given in Fig. 7 D, showing two relevant learning curves. Tracking accuracy is represented by the foothold rewards, whereas technical skills are quantified using the average terrain level [11]. Both scores are substantially higher if the footholds could be observed in both task and joint space.

DISCUSSION

This work demonstrates the potential of a hybrid locomotion pipeline that combines accurate foot placement and dynamic agility of state-of-the-art TO with the inherent robustness and reflex behaviors of novel RL control strategies. Our approach enables legged robots to overcome complex environments that either method alone would struggle with. As such terrains are commonly found in construction sites, mines, and collapsed buildings, our work could help advance the deployment of autonomous legged machines in the fields of construction, maintenance, and search-and-rescue.

We have rigorously evaluated the performance in extensive real-world experiments over the course of about half a year. We included gaps, stepping stones, narrow beams, and tall boxes in our tests, and demonstrated that our method outperformed the RL baseline controller on every single terrain. Next, we evaluated the robustness on slippery and soft ground, each time outperforming two model-based controllers.

Furthermore, we have shown that the emerging policy can track the motion of two different planners utilizing the same trotting gait. This was possible because the observed footholds seem to be mostly invariant under the choice of the optimizer. However, certain obstacles may encourage the deployed planner to produce footprint patterns that otherwise do not emerge during training. In this case, we would expect a degraded tracking performance.

In addition to our main contribution, we have demonstrated several other notable results. First, our policy, which was trained exclusively with visual perception, is still able to generalize to blind locomotion. Second, A simple multilayer perceptron (MLP) trained with an asymmetric actor/critics setup achieves similar robust behaviors as much more complex teacher/student trainings [12, 13]. Third, Our locomotion policy can handle a lot of noise and drift in the visual data without relying on complicated gaited networks, which might be difficult to tune and train [13].

Contrary to our expectations, the proposed training environment was found to not be more sample efficient than similar unifying RL approaches [11, 13]. The large number of epochs required for convergence suggests that foothold accuracy is something intrinsically complicated to learn.

In this work, we emphasized that TO and RL share complementary properties and that no single best method exists to address the open challenges in legged locomotion. The proposed control architecture leverages this observation by combining the planning capabilities of the former and the robustness properties of the latter. It does, by no means, constitute a universal recipe to integrate the two approaches in an optimal way for a generic problem. Moreover, one could even extend the discussion with self- and unsupervised learning, indirect optimal control, dynamic programming, and stochastic optimal control. Nevertheless, our results may motivate future research to incorporate the aspect of planning into the concept RL.

We see several promising avenues for future research. Many successful data-driven controllers have the ability to alter the stride duration of the trotting gait. We expect a further increase in survival rate and technical skills if the network policy could suggest an arbitrary contact schedule to the motion optimizer. Moreover, a truly hybrid method, in which the policy can directly modify the cost function of the planner, may be able to generate more diversified motions. Our results indicate that IK is difficult to learn. To increase the sample efficiency and improve generalization across different platforms, a more sophisticated network structure could exploit prior knowledge of analytical IK. Another potential research direction may focus on leveraging the benefits of sampling trajectories from an offline buffer. This could substantially reduce the training time and allow for the substitution of TAMOLS with a more accurate TO method, or even expert data gathered from real animals.

MATERIALS AND METHODS

Motivation

To motivate the specific architectural design, we first identify the strengths and weaknesses of the two most commonly used control paradigms in legged locomotion.

TO amounts to open-loop control, which produces suboptimal solutions in the presence of stochasticity, modeling errors, and small prediction windows. Unfortunately, these methods also introduce many assumptions, mostly to reduce computation time or achieve favorable numerical properties. For instance, the feet are almost always preselected interaction points to prevent complex collision constraints, contact and actuator dynamics are usually omitted or smoothed out to circumvent stiff optimization problems, and the contact schedule is often pre-specified to avoid the combinatorial problem imposed by the switched system dynamics. Despite a large set of strong assumptions, real-time capable planners are not always truly real-time. The reference trajectories are updated around 5 Hz [31] to 100 Hz [7] and realized between 400 Hz to 1000 Hz. In other words, these methods do not plan fast enough to catch up with the errors they are making. While structural [2] or environmental [7, 20] decomposition may further contribute to the overall suboptimality, they were found useful for extracting good local solutions on sparse terrains. Because the concept of planning is not restricted to the tuning domain, model-based approaches tend to generalize well across different terrain geometries [7, 21]. Moreover, since numerical solvers perform very cheap and sparse operations on the elevation map, the map resolution can be arbitrarily small, facilitating accurate foothold planning.

RL, on the other hand, leads to policies that represent global closed-loop control strategies. Deep neural networks are large capacity models, and as such, can represent locomotion policies without introducing any assumption about the terrain or the system (except from being Markovian). They exhibit good interpolation in-between visited states but do not extrapolate well to unseen environments. Despite their large size, the inference time is usually relatively small. The integration of an actuator model has been demonstrated to improve sim-to-real-transfer [10], while the stochasticity in the system dynamics and training environment can effectively be utilized to synthesize robust behaviors [12, 13].

Contrary toTO, the elevation map is typically down-sampled [11, 13] to avoid immense memory consumption during training.

In summary, TO might be better suited if good generalization and high accuracy are required, whereas RL is the preferred method if robustness is of concern or onboard computational power is limited. As locomotion combines challenges from both of these fields, we formulate the goal of this work as follows: RL shall be used to train a low-level tracking controller that provides markedly more robustness than classical inverse dynamics. The accuracy and planning capabilities of model-based TO shall be leveraged on a low level to synthesize a unifying locomotion strategy that supports diverse and generalizing motions.

Reference Motions

Designing a TO problem for control always involves a compromise, that trades off physical accuracy and generalization against good numerical conditioning, low computation time, convexity, smoothness, availability of derivatives, and the necessity of a high-quality initial guess. In our work, we generate the trajectories using TAMOLS [21]. Unlike other similar methods, it does not require terrain segmentation nor pre-computation of footholds, and its solutions are robust under varying initial guesses. The system dynamics and kinematics are simplified, allowing for fast updates. During deployment, we also compare against baseline-to-2, which builds up on more complex kinodynamics. Due to the increased computation time and in particular the computationally demanding map-processing pipeline, this method is not well-suited to be used directly within the learning process (the training time would be expected to be about eight times larger).

We added three crucial features to TAMOLS: First, we enable parallelization on CPU, which allows multiple optimization problems to be solved simultaneously. Second, we created a python interface using `pybind11` [35], enabling it to run in a python-based environment. Finally, we assume that the measured contact state always matches the desired contact state. This renders the TO independent of contact estimation, which typically is the most fragile module in a model-based controller.

The optimizer requires a discretized 2.5d representation of its environment, a so-called elevation map, as input. We extract the map directly from the simulator by sampling the height across a fixed grid. For both training and deployment, we use a fixed trotting gait with a stride duration of 0.93 s and swing phase of 0.465 s, and set the resolution of the grid map to $0.04 \times 0.04\,\mathrm{m}^2$.

Overview of the Training Environment

The locomotion policy $\pi(\boldsymbol{a} \mid \boldsymbol{o})$ is a stochastic distribution of actions $\boldsymbol{a} \in \mathcal{A}$ that are conditioned on observations $\boldsymbol{o} \in \mathcal{O}$, parametrized by an MLP. The action space comprises target joint positions that are tracked using a PD controller, following the approach in [10] and related works [12–14].

Given the state $\boldsymbol{s} \in \mathcal{S}$, we extract the solution at the next time step $\boldsymbol{x}'(\boldsymbol{s}) \in \mathcal{X} \subseteq \mathcal{S}$ from the optimizer, which includes four footholds $\boldsymbol{p}^*_{i=0,\dots,3}$, joint positions $\boldsymbol{q}^*$ at touch-down time, and the base trajectory evaluated at the next time step. The base trajectory consists of of base pose $\boldsymbol{b}^*(\Delta t)$, twist $\dot{\boldsymbol{b}}^*(\Delta t)$, and linear and angular acceleration $\ddot{\boldsymbol{b}}^*(\Delta t)$. More details can be found in Fig. 8 A. We then sample an action from the policy. It is used to forward simulate the system dynamics, yielding a new state $\boldsymbol{s}' \in \mathcal{S}$, as illustrated in Fig. 8 B.

To define a scalar reward $r(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{x}', \boldsymbol{a})$, we use a monotonically decreasing function of the error between the optimized and measured states, that is $r \propto \boldsymbol{x}'(\boldsymbol{s}) \ominus \boldsymbol{x}(\boldsymbol{s}')$. The minus operator $\ominus$ is defined on the set $\mathcal{X}$, the vector $\boldsymbol{x}'(\boldsymbol{s})$ is the optimized state, and $\boldsymbol{x}(\boldsymbol{s}')$ is the state of the simulator after extracting it on the corresponding subset. The policy network can also be understood as a learned model reference adaptive controller with the optimizer being the reference model.

In this work, we use an asymmetrical actor/critic method for training. The value function approximation $V(\boldsymbol{o}, \tilde{\boldsymbol{o}})$ uses privileged $\tilde{\boldsymbol{o}} \in \tilde{\mathcal{O}}$ as well as policy observations $\boldsymbol{o}$.

Observation Space

The value function is trained on policy observations and privileged observations, while the policy network is trained on the former only [22]. All observations are given in the robot-centric base frame. The definition of the observation vector is given below, whereas noise distributions and dimensionalities of the observation vectors can be found in the supplementary methods and Table 2.

Policy Observations

The policy observations comprise proprioceptive measurements such as base twist, gravity vector, joint positions, and joint velocities. The history only includes previous actions [11]. Additional observations are extracted from the model-based planner, including planar coordinates of foothold positions (xy coordinates), desired joint positions at touch-down time, desired contact state, and time left in the current phase. The latter two are per-leg quantities that fully describe the gait pattern. Footholds only contain planner coordinates since the height can be extracted from the height scan.

The height scan, which is an additional part of the observation space, enables the network to anticipate a collision-free swing leg trajectory. In contrast to similar works, we do not construct a sparse elevation map around the base [11, 27] or the feet [13]. Instead, we sample along a line connecting the current foot position with the desired foothold (Fig. 8 A). This approach has several advantages: First, the samples can be denser by only scanning terrain patches that are most relevant for the swing leg. Second, it prevents the network from extracting other information from the map, which is typically exposed to most uncertainty (for instance, occlusion, reflection, odometry drift, discretization error, etc.). Third, it allows us to conveniently model elevation map drift as a per-foot quantity, which means that each leg can have its own drift value.

We use analytical IK to compute the desired joint positions. As the motion optimizer may not provide a swing trajectory, as is the case for TAMOLS, we completely skip the swing phase. This means that the IK is computed with the desired base pose and the measured foot position for a stance leg, and the target foothold for a swing leg.

It is worth noting that we do not provide the base pose reference as observation. This was found to reduce sensitivity to mapping errors and to render the policy independent of the utilized planner. Finally, to allow the network to infer the desired walking direction, we add the reference twist (before optimization) to the observation space.

Privileged Observations

The privileged observations contain the optimized base pose, base twist, and base linear and angular acceleration, extracted one time step ahead. In addition, the critics can observe signals confined to the simulator, such as the external base wrench, external foot forces, the measured contact forces, friction coefficients, and elevation map drift.

Reward Functions

The total reward is computed as a weighted combination of several individual components, which can be categorized as follows: “tracking” of reference motions, encouraging “consistent” behavior, and other “regularization” terms necessary for successful sim-to-real transfer. The reward functions are explained below whereas weights and parameters are reported in Table 3.

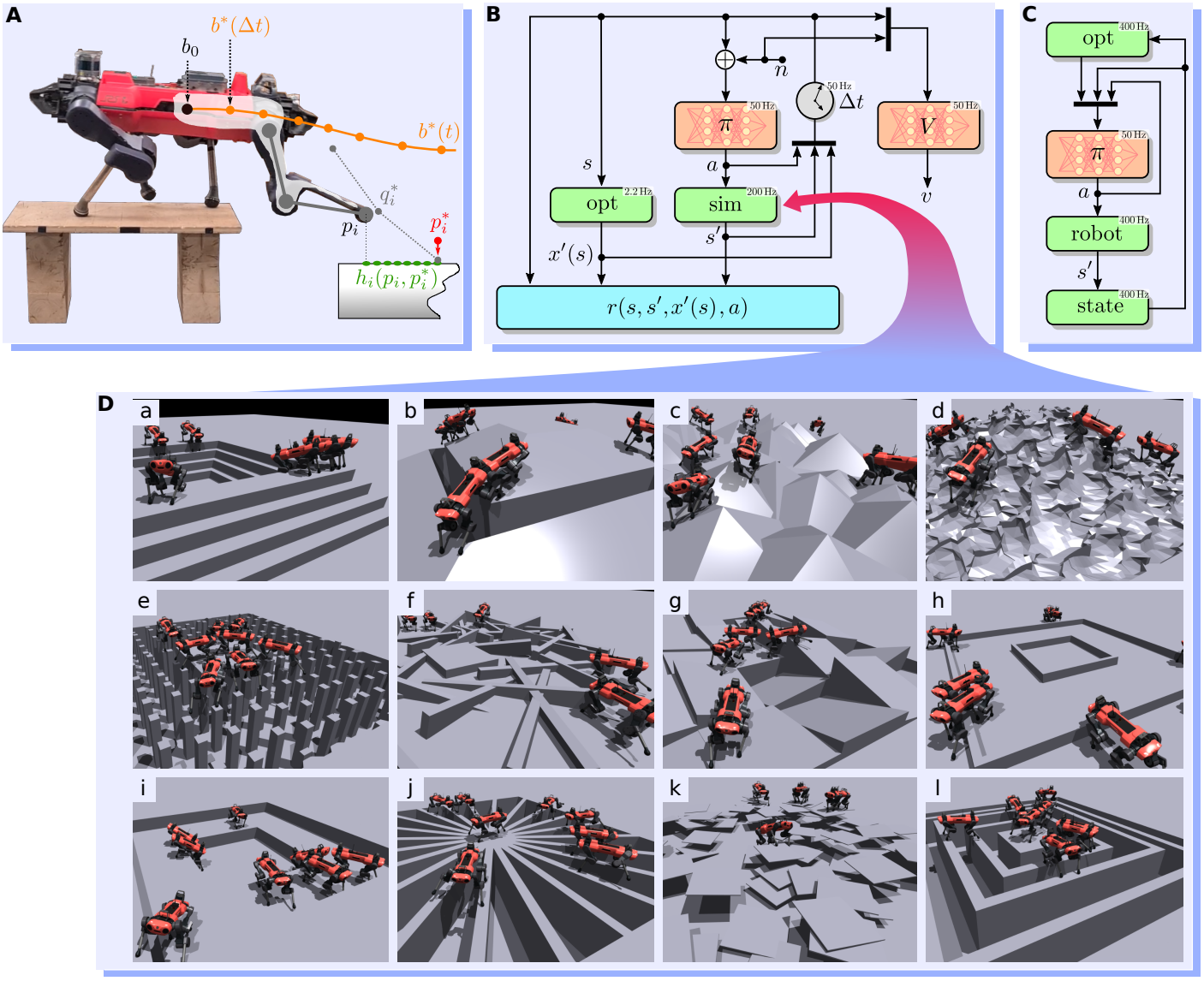

Fig. 8. Overview of the training method and deployment strategy. **(A)** The optimized solution provides footholds $\boldsymbol{p}_i^*$, desired base pose $\boldsymbol{b}^*$, twist $\dot{\boldsymbol{b}}^*$, and acceleration $\ddot{\boldsymbol{b}}^*$ (extracted one policy step Δt ahead), as well as desired joint positions $\boldsymbol{q}^*$. Additionally, a height scan h is sampled between the foot position $\boldsymbol{p}_i$ and the corresponding foothold. **(B)** Training environment: The optimizer runs in parallel to the simulation. At each leg touch-down, a new solution $\boldsymbol{x}'$ is generated. The policy π drives the system response $\boldsymbol{s}'$ toward the optimized solution $\boldsymbol{x}'(\boldsymbol{s})$, which is encouraged using the reward function r. Actor observations are perturbed with the noise vector $\boldsymbol{n}$, while critics and the TO receive ground truth data. **(C)** Deployment: Given the optimized footholds, the network computes target joint positions that are tracked using a PD control law. The state estimator (state) returns the estimated robot state, which is fed back into the policy and the optimizer. **(D)** The list of terrain types includes a) stairs, b) combinations of slopes and gaps, c) pyramids, d) slopped rough terrain, e) stepping stones, f) objects with randomized poses, g) boxes with tilted surfaces, h) rings, i) pits, j) beams, k) hovering objects with randomized poses, and l) pallets.

Base Pose Tracking

To achieve tracking of the reference base pose trajectory, we use

$$r_{Bn} = e^{-\sigma_{Bn} \cdot ||\boldsymbol{b}^*(t+\Delta t)^{(n)} \ominus \boldsymbol{b}(t)^{(n)}||^2}, \quad \textbf{(1)}$$

where $n = \{0, 1, 2\}$ is the derivative order, $\boldsymbol{b}(t)$ is the measured base pose, $\boldsymbol{b}^*(t + \Delta t)$ is the desired base pose sampled from the reference trajectory one policy step Δt ahead, and $\ominus$ denotes the quaternion difference for base orientation and the vector difference otherwise. We refer to the above reward function as a "soft" tracking task because large values can be scored even if the tracking error does not perfectly vanish.

To further analyze the reward function, we decompose the base trajectory into three segments. The "head" starts at time zero, the "tail" stops at the prediction horizon, and the "middle" connects these two segments with each other. A logarithmic reward function would prioritize the tracking of the trajectory head, while a linear penalty would focus on making progress along the whole trajectory at once. Contrary, the exponential shape of the reward function splits the tracking task into several steps. During the initial epochs, the tracking error of the trajectory middle and tail will likely be relatively large, and thus, do not contribute notably to the reward gradient. As a result, the network will minimize the tracking error of the trajectory head. Once its effect on the gradient diminishes, the errors corresponding to the trajectory middle will dominate the gradient landscape. In the final training stages, tracking is mostly improved around the trajectory tail.

Foothold Tracking

We choose a logarithmic function

$$r_{pi} = -\ln(||\boldsymbol{p}_i^* - \boldsymbol{p}_i||^2 + \epsilon), \quad \textbf{(2)}$$

to learn foothold tracking, where $\boldsymbol{p}_i$ is the current foot position of leg $i \in \{0, \ldots, 3\}$, $\boldsymbol{p}_i^*$ is the corresponding desired foothold, and $0 < \epsilon \ll 1$ is small number ensuring the function is well defined. The above reward function may be termed a "hard" tracking task, as the maximum value can only be scored if the error reaches zero. As the tracking improves, the gradients will become larger, resulting in even tighter tracking toward the later training stages.

A dense reward structure typically encourages a stance foot to be dragged along the ground to further minimize the tracking error. To prevent such drag motions from emerging, the above reward is given for each foot at most once during one complete gait cycle: more specifically, if and only if the leg is intended to be in contact and the norm of the contact force indicates a contact, that is if $||\boldsymbol{f}_i|| > 1$, then the agent receives the reward.

Consistency

In RL for legged locomotion, hesitating to move over challenging terrains is a commonly observed phenomenon that prevents informative samples from being gathered and thus impedes the agent's performance. This behavior can be explained by insufficient exploration: The majority of agents fail to solve a task while a small number of agents achieve higher average rewards by refusing to act. To overcome this local optimum, we propose to encourage consistency by rewarding actions that are intended by previous actions. In our case, we measure consistency as the similarity between two consecutive motion optimizations. If the solutions are similar, the agent is considered to be "consistent". We measure similarity as the Euclidean distance between two adjacent solutions and write

$$r_c = \sum_{\delta tj + t_0 \in (T_a \cap T_b)} -\delta t ||\boldsymbol{b}_a^*(\delta tj + t_{0,a}) \ominus \boldsymbol{b}_b^*(\delta tj + t_{0,b})|| - w_p ||\boldsymbol{p}_a^* - \boldsymbol{p}_b^*||. \quad \textbf{(3)}$$

Here, $\boldsymbol{p}_t^*$ with $t = \{a, b\}$ is a vector of stacked footholds, $w_p > 0$ is a relative weight, $\delta t = 0.01\,\text{s}$ is the discretization time of the base trajectory, and t_0 is the time elapsed since the solution was retrieved. The index a refers to the most recent solution, while b refers to the previous solution. It is important to note that the two solution vectors $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$, from which we extract the base and footholds, are only defined on their respective time intervals given by the optimization horizon τ_h, i.e, $t_a \in T_a = [0, \tau_{h,a}]$ and $t_b \in T_b = [0, \tau_{h,b}]$.

Regularization

To ensure that the robot walks smoothly, we employ two different penalty terms enforcing complementary constraints. The first term, $r_{r1} = -\sum_i |\boldsymbol{v}_i^T \boldsymbol{f}_i|$, discourages foot-scuffing and end-effector collisions by penalizing power measured at the feet. The second term, $r_{r2} = -\sum_i (\dot{\boldsymbol{q}}_i^T \boldsymbol{\tau}_i)^2$, penalizes joint power to prevent arbitrary motions, especially during the swing phase. Other regularization terms are stated in the supplementary methods (section "Implementation Details").

Training Environment

To train the locomotion policy, we employ a custom version of Proximal Policy Optimization (PPO)[36] and a training environment that is mostly identical to that introduced in[11]. It is explained in more detail in the supplementary methods (section "Training Details") and Table 1. Simulation and back-propagation are performed on GPU, while the optimization problems are solved on CPU.

Termination

We use a simple termination condition where an episode is terminated if the base of the robot makes contact with the terrain.

Domain Randomization

We inject noise into all observations except for those designated as privileged. At each policy step, a noise vector $\boldsymbol{n}$ is sampled from a uniform distribution and added to the observation vector, with the only exceptions of the desired joint positions and the height scan.

For the elevation map, we add noise before extracting the height scan. The noise is sampled from an approximate Laplace distribution where large values are less common than small ones. We perturb the height scan with a constant offset, which is sampled from another approximate Laplace distribution for each foot separately. Both perturbations discourage the network to rely extensively on perceptive feedback and help to generalize to various perceptive uncertainties caused by odometry drift, occlusion, and soft ground.

All robots are artificially pushed by adding a twist offset to the measured twist at regular time instances. Friction coefficients are randomized per leg once at initialization time. To render the motion robust against disturbances, we perturb the base with an external wrench and the feet with external forces. The latter slightly stiffens up the swing motion but improves tracking performance in the presence of unmodeled joint frictions and link inertia. The reference twist is resampled in constant time intervals and then held constant.

The solutions for the TO problems are obtained using ground truth data, which include the true friction coefficients, the true external base wrench, and noise-free height map. In the presence of simulated noise, drift, and external disturbances, the policy network is therefore trained to reconstruct a base trajectory that the optimizer would produce given the ground truth data. However, there is a risk that the network learns to remove the drift from the height scan by analyzing the desired joint positions. During hardware deployment, such a reconstruction will fail because the optimizer is subject to the same height drift. To mitigate this issue, we introduce noise to the desired joint position observations, sampled from a uniform distribution with boundaries proportional to the drift value.

Terrain Curriculum

We use a terrain curriculum as introduced in [11]. Before the training process, terrain patches of varying types and difficulties are generated, and each agent is assigned a terrain patch. As an agent acquires more skills and can navigate the current terrain, its level is upgraded, which means it will be re-spawned on the same terrain type, but with a harder difficulty. We have observed that the variety of terrains encountered during training influences the sim-to-real transfer. We thus have included a total of 12 different terrain types with configurable parameters (Fig. 8 D), leading to a total of 120 distinguishable terrain patches. The terrain types classify different locomotion behaviors, s.a. climbing ("stairs", "pits", "boxes", "pyramids"), reflexing ("rough", "rings", "flying objects"), and walking with large steps ("gaps", "pallets", "stepping stones", "beams", "objects with randomized poses"). Our terrain curriculum consists of 10 levels, where one of the configurable parameters is modulated to increase or decrease its difficulty. This results in a total of 1200 terrain patches, each with a size of $8 \times 8\,\mathrm{m}^2$, summing up to a total area of $76800\,\mathrm{m}^2$, which is approximately the size of 14 football fields or 10 soccer fields.

Training

Solving the TO problems at the policy frequency during training was found to provoke poor local optima. In such a case, the optimizer adapts the solution after each policy step: If the agent is not able to follow the reference trajectory, the optimizer will adapt to the new state s.t. the tracking problem becomes feasible again. This means that the agent can exhibit "lazy" behavior and still collect some rewards. We prevent such a local optimum by updating the optimizer only at a leg touch-down (after 0.465 seconds). This also greatly reduces learning time because computational costs are reduced by a factor of 23. After a robot fell (on average, once every 18 seconds), was pushed (after 10 seconds) or its twist commands changed (three times per episode), the optimized trajectories are no longer valid. To guarantee that the locomotion policy generalizes across different update rates, we additionally recompute the solution in all those scenarios.

We trained the policy with a massive parallelization of $64^2 = 4096$ robots, for a total of 90000 epochs. Each epoch consisted of 45 learning iterations where each iteration covered a duration of 0.02 seconds. Considering the variable update rate explained previously, this resulted in a total of 8295 days (or 23 years) of optimized trajectories. The policy can be deployed after about one day of training (6000 epochs), reaches 90 % of its peak performance after three days (20000 epochs), and is fully converged after two weeks (90000 epochs).

In comparison, the baseline-rl-1 policy was trained for 4000 epochs with 1000 parallelized robots over 5 consecutive days. Each epoch lasted for 5 seconds, resulting in a throughput of 46 simulated seconds per second. Our policy was trained for 14 days, with each epoch lasting for 0.9 seconds, leading to a throughput of 27 simulated seconds per second. Thus, despite generating 1.6 years of desired motions per day, our approach has only a 1.7 times lower throughput than the baseline.

Deployment

We deploy the policy at a frequency of 50 Hz without any fine-tuning. The motion optimizer runs at the largest possible rate in a separate thread. For TAMOLS with a trotting gait, this is around 400 Hz and for baseline-to-2 around 100 Hz (both are faster than the policy frequency). At each step, the policy queries the most recent solution from the thread pool and extracts it $\Delta t = 0.02\,\mathrm{s}$ ahead of the most recent time index.

For our experiments, we used three different types of ANYmal robots [34], two version C and one version D, for which we trained different policies. ANYmal C is by default equipped with four Intel RealSense D435 depth cameras whereas ANYmal D has eight depth cameras of the same type. For the second Version C, the depth cameras were replaced with two identical Robosense Bpearl dome LiDAR sensors. For the outdoor experiments, we mostly used this robot, as the Bpearls tend to be more robust against lighting conditions. Motion optimization and the forward propagation of the network policy are done on a single Intel core-i7 8850H machine. Elevation mapping [37] runs on a dedicated onboard Nvidia Jetson.

REFERENCES

1. J. Z. Kolter, M. P. Rodgers, A. Y. Ng, A control architecture for quadruped locomotion over rough terrain, *2008 IEEE International Conference on Robotics and Automation*, 811–818 (2008).
2. M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, S. Schaal, Fast, robust quadruped locomotion over challenging terrain, *2010 IEEE International Conference on Robotics and Automation*, 2665–2670 (2010).
3. A. W. Winkler, C. D. Bellicoso, M. Hutter, J. Buchli, Gait and trajectory optimization for legged systems through phase-based end-effector parameterization, *IEEE Robotics and Automation Letters* 1560–1567 (2018).
4. C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, C. Semini, Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control, *IEEE Transactions on Robotics* 1635–1648 (2020).
5. F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, M. Hutter, Perceptive locomotion in rough terrain – online foothold optimization, *IEEE Robotics and Automation Letters* 5370–5376 (2020).
6. P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, M. Hutter, Robust rough-terrain locomotion with a quadrupedal robot, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 5761–5768 (2018).
7. R. Grandia, F. Jenelten, S. Yang, F. Farshidian, M. Hutter, Perceptive locomotion through nonlinear model-predictive control, *IEEE Transactions on Robotics* 1–20 (2023).
8. C. Mastalli, W. Merkt, G. Xin, J. Shim, M. Mistry, I. Havoutis, S. Vijayakumar, Agile maneuvers in legged robots: a predictive control approach (2022).
9. C. D. Bellicoso, F. Jenelten, C. Gehring, M. Hutter, Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots, *IEEE Robotics and Automation Letters* 2261–2268 (2018).
10. J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots, *Science Robotics* p. eaau5872 (2019).
11. N. Rudin, D. Hoeller, P. Reist, M. Hutter, Learning to walk in minutes using massively parallel deep reinforcement learning, *5th Annual Conference on Robot Learning* (2021).
12. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning quadrupedal locomotion over challenging terrain, *Science Robotics* p. eabc5986 (2020).
13. T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning robust perceptive locomotion for quadrupedal robots in the wild, *Science Robotics* p. eabk2822 (2022).
14. N. Rudin, D. Hoeller, M. Bjelonic, M. Hutter, Advanced skills by learning locomotion and local navigation end-to-end, *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2497–2503 (2022).
15. Z. Xie, H. Y. Ling, N. H. Kim, M. van de Panne, Allsteps: Curriculum-driven learning of stepping stone skills, *Computer Graphics Forum* **39** (2020).
16. H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, J. Siekmann, A. Fern, J. Hurst, Sim-to-real learning of footstep-constrained bipedal dynamic walking, *2022 International Conference on Robotics and Automation (ICRA)*, 10428–10434 (2022).
17. W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, T. Zhang, Visual-locomotion: Learning to walk on complex terrains with vision, *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, G. Neumann, eds., 1291–1302 (PMLR, 2022).
18. A. Agarwal, A. Kumar, J. Malik, D. Pathak, Legged locomotion in

challenging terrains using egocentric vision, *6th Annual Conference on Robot Learning* (2022).
19. K. Caluwaerts, A. Iscen, J. C. Kew, W. Yu, T. Zhang, D. Freeman, K.-H. Lee, L. Lee, S. Saliceti, V. Zhuang, N. Batchelor, S. Bohez, F. Casarini, J. E. Chen, O. Cortes, E. Coumans, A. Dostmohamed, G. Dulac-Arnold, A. Escontrela, E. Frey, R. Hafner, D. Jain, B. Jyenis, Y. Kuang, E. Lee, L. Luu, O. Nachum, K. Oslund, J. Powell, D. Reyes, F. Romano, F. Sadeghi, R. Sloat, B. Tabanpour, D. Zheng, M. Neunert, R. Hadsell, N. Heess, F. Nori, J. Seto, C. Parada, V. Sindhwani, V. Vanhoucke, J. Tan, Barkour: Benchmarking animal-level agility with quadruped robots (2023).
20. R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, J. Pratt, Footstep planning for autonomous walking over rough terrain, *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 9–16 (2019).
21. F. Jenelten, R. Grandia, F. Farshidian, M. Hutter, Tamols: Terrain-aware motion optimization for legged systems, *IEEE Transactions on Robotics* 3395–3413 (2022).
22. P. Brakel, S. Bohez, L. Hasenclever, N. Heess, K. Bousmalis, Learning coordinated terrain-adaptive locomotion by imitating a centroidal dynamics planner, *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10335–10342 (2022).
23. M. Bogdanovic, M. Khadiv, L. Righetti, Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization, *Frontiers in Robotics and AI* **9** (2022).
24. X. B. Peng, P. Abbeel, S. Levine, M. van de Panne, Deepmimic: Example-guided deep reinforcement learning of physics-based character skills, *ACM Transactions on Graphics* **37** (2018).
25. X. B. Peng, Z. Ma, P. Abbeel, S. Levine, A. Kanazawa, Amp: Adversarial motion priors for stylized physics-based character control, *ACM Transactions on Graphics* **40** (2021).
26. S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner, M. Wulfmeier, M. Neunert, B. Moran, N. Siegel, A. Huber, F. Romano, N. Batchelor, F. Casarini, J. Merel, R. Hadsell, N. Heess, Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors (2022).
27. V. Tsounis, M. Alge, J. Lee, F. Farshidian, M. Hutter, Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning, *IEEE Robotics and Automation Letters* 3699–3706 (2020).
28. X. B. Peng, G. Berseth, K. Yin, M. van de Panne, Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning, *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)* **36** (2017).
29. O. Melon, M. Geisert, D. Surovik, I. Havoutis, M. Fallon, Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations (2020).
30. D. Surovik, O. Melon, M. Geisert, M. Fallon, I. Havoutis, Learning an expert skill-space for replanning dynamic quadruped locomotion over obstacles, *Proceedings of the 2020 Conference on Robot Learning*, J. Kober, F. Ramos, C. Tomlin, eds., 1509–1518 (PMLR, 2021).
31. O. Melon, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, M. Fallon, Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization, *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 9805–9811 (2021).
32. S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, I. Havoutis, Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control, *IEEE Transactions on Robotics* 2908–2927 (2022).
33. Z. Xie, X. Da, B. Babich, A. Garg, M. v. de Panne, Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model, *Algorithmic Foundations of Robotics XV*, S. M. LaValle, J. M. O'Kane, M. Otte, D. Sadigh, P. Tokekar, eds., 523–539 (Springer International Publishing, Cham, 2023).
34. M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, M. Hoepflinger, Hutter2016 - a highly mobile and dynamic quadrupedal robot, *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 38–44 (2016).
35. W. Jakob, J. Rhinelander, D. Moldovan, pybind11 – seamless operability between c++11 and python (2017). Https://github.com/pybind/pybind11.
36. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *CoRR* **abs/1707.06347** (2017).
37. T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, M. Hutter, Elevation mapping for locomotion and navigation using gpu (2022).
38. F. Jenelten, J. He, F. Farshidian, M. Hutter, Evaluation of tracking performance and robustness for a hybrid locomotion controller, https://doi.org/10.5061/dryad.b5mkkwhkq.

ACKNOWLEDGMENTS

Funding: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 852044. This research was supported by the Swiss National Science Foundation (SNSF) as part of project No 188596, and by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics (NCCR Robotics). **Author Contribution:** F.J. formulated the main ideas, trained and tested the policy using baseline-to-1, and conducted most of the experiments. J.H. interfaced baseline-to-2 with the tracking policy and conducted the box-climbing experiments. F.F. contributed to the Theory and improved some of the original ideas. All authors helped to write, improve, and refine the paper. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All (other) data needed to evaluate the conclusions in the paper are present in the paper or the Supplementary Materials. Data-sets and code to generate all our figures are made publicly available [38].

SUPPLEMENTARY METHODS

Nomenclature

$(\cdot)_i$ index for leg $i \in \{0,\dots,3\}$
τ_h prediction horizon
$\boldsymbol{b}^*(t)$ optimized base pose trajectory
$\dot{\boldsymbol{b}}^*(t)$ optimized base twist trajectory
$\ddot{\boldsymbol{b}}^*(t)$ optimized base acceleration trajectory
$\boldsymbol{q}^*$ optimized joint positions obtained from IK
$\boldsymbol{p}_i^*$ optimized foothold
$\boldsymbol{p}_t$ vector of stacked footholds, computed at time t
$\boldsymbol{p}_i$ measured foot position
$\boldsymbol{f}_i$ measured contact force
$\boldsymbol{v}_i$ measured foot velocity
$\boldsymbol{q}_i$ measured joint positions
$\boldsymbol{\tau}_i$ measured joint torques
$\boldsymbol{s}$ state of the robot
$\boldsymbol{x}'(\boldsymbol{s})$ solution of optimization problem one time-step ahead, initialized with state $\boldsymbol{s}$
$\boldsymbol{x}(\boldsymbol{s}')$ state $\boldsymbol{s}'$ extracted on subset of the optimizer
$\boldsymbol{o}$ policy observations
$\tilde{\boldsymbol{o}}$ privileged observations
$\boldsymbol{a}$ actions

Training Details

We use a simulation time step of 5 ms and a policy time step of 20 ms.

The elevation map noise is sampled from a Laplace distribution, which is approximated by two uniform distributions. We first sample the boundaries of the uniform distribution $h_{n,max} \sim \mathcal{U}(0, 0.2)\,\mathrm{m}$, and then sample the noise from $h_n \sim \mathcal{U}(-h_{n,max}, h_{n,max})$. Similar holds for the height drift: We first sample the boundaries $h_{d,max} \sim \mathcal{U}(0, 0.2)\,\mathrm{m}$ and then sample five drift values $h_{d0},\dots,h_{d4} \sim \mathcal{U}(-h_{d,max}, h_{d,max})$, where h_{d0} is constant for the entire map and $h_{d1}\dots,h_{d4}$ is sampled for each foot individually. The total drift h_d is obtained by summing up the map-drift and the per-foot drift. Elevation map noise and drift are re-sampled after a constant time interval of 8 s.

The noise vector added to the desired joint positions depends on the height drift, and is sampled from the uniform distribution $q_n \sim 2 \cdot \mathcal{U}(-h_d, h_d)$.

We push the robots after 10 s by adding a twist offset to the measured twist sampled from $\Delta\dot{\boldsymbol{b}} \sim \mathcal{U}(-1,1)[m/s]$. Friction coefficients are randomized per leg and sampled from $\mu \sim \mathcal{U}(0.1, 1.2)$. For each episode, we sample an external base wrench $\boldsymbol{\tau}_B \sim \mathcal{U}(-15, 15)$ and a external foot force $\boldsymbol{f}_{ee} \sim \mathcal{U}(-2, 2)\,\mathrm{N}$.

The reference twist is re-sampled from a uniform distribution three times per episode. A third of the robots has zero lateral velocity whereas the heading velocity is sampled from $v_x \sim \mathcal{U}(-1, 1)\,\mathrm{m/s}$. Another third has zero heading velocity and its lateral velocity is sampled from $v_y \sim \mathcal{U}(-0.8, 0.8)\,\mathrm{m/s}$. The last third has mixed heading and lateral velocities sampled from their respective distributions. For all three cases, the yaw velocity is sampled from $v_\psi \sim \mathcal{U}(-0.8, 0.8)\,\mathrm{rad/s}$.

The PPO hyperparameters used for the training are stated in Table 1. All policies were trained with a seed of 1.

Implementation Details

To parameterize the policy (or actor) network, a Gaussian distribution is used, where the mean is generated by an MLP parametrized by θ. The standard deviation is added independently of the observations as an additional layer in the network. Specifically, the policy can be written as $\pi(\boldsymbol{a} \mid \boldsymbol{o}) \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{o}), \boldsymbol{\sigma}_\theta)$, where $\boldsymbol{\mu}_\theta(\boldsymbol{o})$ represents the mean and $\boldsymbol{\sigma}_\theta$ the standard deviation. The value function (or critics) is generated by another MLP as $V(\boldsymbol{o}, \tilde{\boldsymbol{o}}) \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{o}, \tilde{\boldsymbol{o}}), \boldsymbol{0})$, which is parametrized by ϕ. For both MLP's we use three hidden layers with each having 512 neurons. Observations are normalized using running means and running standard deviation.

parameter type	number
batch size	$45 \cdot 4096 = 184320$
mini batch size	$4 \cdot 4096 = 16384$
number of epochs	5
clip range	0.2
entropy coefficient	0.0035
discount factor	0.99
GAE discount factor	0.95
desired KL-divergence	0.01
learning rate	adaptive

Table 1. PPO hyperparameters. The value 4096 is the number of parallelized environments. All policies were trained with the same parameters.

Policy and privileged observations can be found in Table 2, and relative weights of the reward functions are given in Table 3. The implementation of most of the reward functions are given in the main text. In addition to those, we penalize the action rate $r_a = -||\boldsymbol{a}_t - \boldsymbol{a}_{t-1}||^2$ and joint acceleration $r_{\ddot{\boldsymbol{q}}} = -||\ddot{\boldsymbol{q}}||^2$, with $\boldsymbol{a}_t, \boldsymbol{a}_{t-1}$ being the current and previous actions, and $\ddot{\boldsymbol{q}}$ the measured joint accelerations.

Experimental Setup for Evaluation of Optimizer Rate

The experimental setup mostly coincided with the training environment as detailed in the supplementary methods (section "Training Details"), with four minor differences: The terrain curriculum was disabled, and for each terrain type, we selected the most difficult one. A push or a velocity change did not trigger a new solution to be computed. The optimizer was not informed about the true values for friction coefficients and external base wrenches. Instead, nominal values were used. All robots were walking in heading direction with a reference velocity sampled uniformly from the interval $0.8\dots0.95\,\mathrm{m/s}$. The robots were pushed after 5 s with a disturbance twist sampled from the interval $\Delta\dot{\boldsymbol{b}} \sim \mathcal{U}(-2, 2)\,\mathrm{m/s}$, which is twice as much as experienced during training. We did not add external foot forces, noise, or map drift. For each optimizer rate in $\{1, 2, 2.5, 3.3, 5, 10, 20, 50\}$ Hz, the success and failure rates were averaged over 8 different experiments conducted with varying seeds selected from the interval $\{2,\dots,9\}$. Each experiment was conducted with 4096 robots, distributed across 120 terrains, lasting for one episode ($= 20$ s).

Experimental Setup for Performance Evaluation

Baseline-rl-3 used the same observations and policy structure as baseline-rl-2. But it was trained within the same environment as ours using a comparable number of epochs.

For the evaluation on all terrains, we used the training environment as detailed in the supplementary methods (section "Training Details"), with the following simplifications: The terrain curriculum was disabled, and for each terrain type, we selected the most difficult one. All robots were walking in heading direction with a reference velocity sampled uniformly from the interval $0.8\dots0.95\,\mathrm{m/s}$. We did not add external foot forces, noise, or map drift. The friction coefficients between feet and ground were set to 1. In addition, for the evaluation on specific terrains, we did not push the robots, and we did not re-sample

type	observations	dim	noise
policy	base twist	6	
	gravity vector	3	± 0.05
	joint positions	12	± 0.01
	joint velocities	12	± 1.5
	previous actions	12	0
	planar footholds	8	± 0.05
	desired joint positions	12	Laplace
	desired contact state	4	
	time left in phase	4	
	reference twist	3	
	height scan	40	Laplace
privileged	desired base position	3	
	desired base quaternion	4	
	desired base twist	12	
	consistency reward	1	
	external base wrench	6	
	external foot force	12	
	friction coefficients	4	
	height drift	4	

Table 2. Policy and privileged observations. The column "noise" contains the upper and lower values of the uniform distribution, with units identical to that one of the observations. Entries marked with "Laplace" indicate the noise distribution is sampled from an approximate Laplace distribution as explained in the supplementary methods (section "Training Details"). A missing value indicates that the noise level is zero.

reward type	reward name	weight	parameter
tracking	base position	1	$\sigma = 1200$
	base rotation	1	$\sigma = 90$
	base linear velocity	1	$\sigma = 10$
	base angular velocity	1	$\sigma = 1$
	base linear acceleration	1	$\sigma = 0.05$
	base angular acceleration	1	$\sigma = 0.005$
	footholds	6	$\epsilon = 10^{-5}$
consistent behavior	consistency	20	
regularization	foot power	-0.02	
	joint power	-0.025	
	action rate	-0.02	
	joint acceleration	-10^{-6}	

Table 3. Reward functions.

the reference velocities. This setup relaxes the locomotion task on topologically hard terrains such as stepping stones, that otherwise might not be traversable. Results were averaged over eight different seeds.

Experimental Setup for Performance Evaluation under Drift

The experimental setup mostly coincided with the training environment as detailed in the supplementary methods (section "Training Details"), with the following differences: The terrain curriculum was disabled, and for each terrain type, we selected the most difficult one. All robots were walking in heading direction with a reference velocity sampled uniformly from the interval $0.8 \ldots 0.95\,\mathrm{m/s}$. We did not add external foot forces or noise. The friction coefficients between feet and ground were set to 1. We did not push the robots and we did not re-sample the reference velocities. In addition, for the flat ground experiment, we replaced all terrain types with a single horizontal plane. Each experiment was repeated 21 times with a linearly increasing drift value selected from $\{0.0, 0.025, \ldots, 0.5\}\,\mathrm{m}$. Results were averaged over eight different seeds.