Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning

Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon and Ioannis Havoutis

***Abstract*— We present a control architecture for real-time adaptation and tracking of trajectories generated using a terrain-aware trajectory optimization solver. This approach enables us to circumvent the computationally exhaustive task of online trajectory optimization, and further introduces a control solution robust to systems modeled with approximated dynamics. We train a policy using deep reinforcement learning (RL) to introduce additive deviations to a reference trajectory in order to generate a feedback-based trajectory tracking system for a quadrupedal robot. We train this policy across a multitude of simulated terrains and ensure its generality by introducing training methods that avoid overfitting and convergence towards local optima. Additionally, in order to capture terrain information, we include a latent representation of the height maps in the observation space of the RL environment as a form of exteroceptive feedback. We test the performance of our trained policy by tracking the corrected *set points* using a model-based whole-body controller and compare it with the tracking behavior obtained without the corrective feedback in several simulation environments. We also show successful transfer of our training approach to the real physical system and further present cogent arguments in support of our framework.**

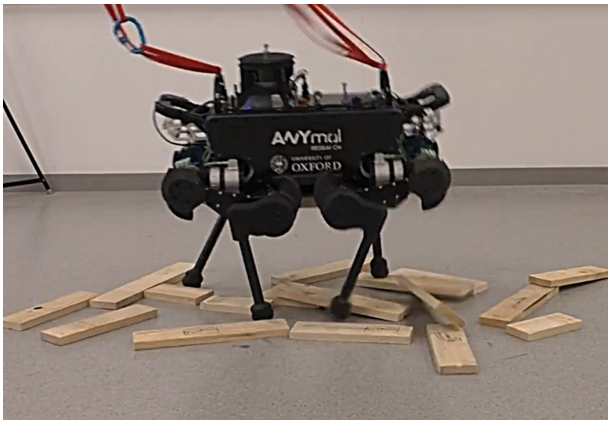

Fig. 1. The ANYmal quadruped we used for evaluating the performance of our control architecture. Among a series of tasks, we tested the response of the WBC by performing blind locomotion over randomly placed wooden tiles using corrective trajectory feedback and compared it against the baseline comprising of non-adaptive trajectory tracking. Accompanying video can be found at: https://youtu.be/GieUI6WLv7U

I. Introduction

Legged locomotion has largely been approached using model-based control methods such as short-horizon motion planning (*e.g.,* one gait cycle ahead) which can be performed online for predefined gaits to optimize target footholds and center of mass (CoM) trajectories in a model predictive control (MPC) manner [1], [2]. Such a method enables humanoid and quadrupedal robot systems to instantly recover from unexpected disturbances or to blindly adapt to rough terrains [3]. In contrast, long-horizon nonlinear trajectory optimization can be performed efficiently to optimize for base trajectories (*i.e.,* base pose and twist), target footholds, feet trajectories, contact forces and timings [4]. Despite its minimal parameterization, the computational performance of such methods are not yet suitable for fast online motion planning. In this regard, much of the research focuses on either improving the convergence rate of this formulation by exploiting learnt initial guess [5], [6] or by introducing feasibility constraints in order to account for the dynamic limits of the actual hardware [7]. In both these cases, the successful execution of the pre-determined motion plan relies on a whole-body controller (WBC) used to compute the feed-forward joint torques, target joint velocities and target joint positions in order to track the generated reference trajectories online.

While long-horizon approaches have shown impressive results in controlled environments [4], their usage outdoors has been very limited mainly due to the slow response time associated with these methods making them unreliable in presence of unexpected disturbances, perturbations and inaccurate whole-body tracking. Recent work on RL for quadrupedal locomotion [8], [9] has shown great promise for development of robust and dynamic model-free data-driven control techniques which directly map sensory information into desired robot states enabling extremely fast control response. However, in order to be used in safety-critical environments, the RL training setup necessitates constrained exploration and control behaviour [10] which requires significant platform-specific engineering thereby increasing the complexity of the RL training environment.

In this regard, we propose a new control architecture which utilizes long horizon motion plans generated using trajectory optimization solvers, corrected online using an RL policy then tracked by a model-based WBC. The WBC ensures that the necessary safety-critical constraints such as joint kinematic limits and peak joint torques are enforced during operation. The corrective RL policy generates additive deviations to the reference trajectories with the objective of

This work was supported by the UKRI/EPSRC RAIN Hub [EP/R026084/1] and the EU H2020 Projects MEMMO and THING, the EPSRC grant 'Robust Legged Locomotion' [EP/S002383/1] and a Royal Society University Research Fellowship (Fallon). This work was conducted as part of ANYmal Research, a community to advance legged robotics. The authors are with Dynamic Robots Systems (DRS) group, Oxford Robotics Institute, University of Oxford, UK. Email: {siddhant, mathieu, rorsolino, mfallon, ioannis}@robots.ox.ac.uk

prioritizing stability over tracking extremely dynamic (and possibly infeasible) trajectories in addition to directing the robot towards the desired state. We evaluate the performance of our control architecture and show that introducing the corrective feedback in the tracking loop results in more stable and feasible locomotion plans. We use the ANYmal [11] quadruped to perform these tests, both, in simulation and on the physical robot system as shown in Fig. 1.

A. Related Work

A significant amount of research in RL and optimal control (OC) domains has focused on tackling shared objectives using a unified approach [12], [13] including contributions in the form of model-based RL [14], guided policy search (GPS) [15], [16] and even a reformulation of the stochastic optimal control problem in terms of KL divergence minimization [12].

Further contributions which address certain issues associated with RL algorithms such as sample inefficiency and reward engineering, have been presented. Such methods rely on using a baseline controller so as to accelerate model-free learning as a solution [17], [18].

There has also been work which directly builds on the baseline controller in order to learn a corrective control behavior. These corrective controllers have been employed for manipulation tasks [19] and also for legged locomotion [20]. In the case of [20], a model-based controller is implemented to efficiently solve the rigid body dynamics while the model-free data-driven controller is used to capture properties such as contacts and friction which are difficult to model.

B. Contributions

Our work extends upon the above research to incorporate an RL-based trajectory correction module that performs online adaptation of the given reference trajectories. In this regard, our contributions are listed below.

- For terrain-aware locomotion, we introduce latent representations of height maps as a form of exteroceptive feedback in order to capture terrain information. We approximate this latent space representation using an unsupervised auto-encoder training strategy for dimensionality reduction and feature extraction.
- To perform successful transfer of the policy trained in simulations to the physical system, we model the actuation dynamics of the robot using an actuator network. We extend the approach detailed in [8], to generate effective joint torques for measured and commanded joint position, velocity and feed-forward torques, and also for different sets of PD gains.
- To generalize over a wide range of observation domains, we introduce a method to generate a large set of procedurally generated terrains. We introduce these terrain in our simulated RL environment setup.
- In order to perform sample-efficient reinforcement learning, we introduce a dense reward function that incorporates a robot stability metric. This is important to account for the delayed rewards which may occur when the robot becomes unstable before certain termination criteria are reached.

Finally, we show that this corrective control architecture can be transferred on to a real robot to improve its robustness to perturbation while tracking highly dynamic trajectories.

II. Overview

We use a trajectory optimization technique to generate long-horizon base and end-effector task space motion plans. Then, during the online execution of such trajectories, we use the feedback generated using the corrective RL policy for a history of tracking errors to modify the base and end-effector motion plans which are tracked using a WBC. The WBC finally generates the desired joint position, velocity and feedforward torque commands which are then tracked by the actuators using a PD controller. This motion generation scheme is illustrated in Fig. 2.

While the approach presented in this paper could be applied to continuous motions generated by re-running the trajectory optimization online, this paper limits its study to motions that are optimized offline and then tracked by the whole-body controller.

A. Trajectory Optimization

In this work, we utilize the trajectory optimizaztion solver TOWR [4] together with the smoothing costs described in [6] to generate base and end-effector (feet) motion plans. The TOWR framework formulates the locomotion problem of legged robots as a non-linear optimization problem which can generate dynamic trajectories for locomotion over complex 3D terrains. The problem is discretized into a numerically-solvable formulation using a direct-collocation transcription method which is then solved using an interior point approach [21].

To stabilize the behavior of the optimizer on uneven terrains, we consider the friction cone axis aligns with the gravity axis. This limits TOWR, which uses a local optimization approach, from considerable divergence during optimization around regions with high deviations in terrain slope. Moreover, we introduce a cost term which penalizes the gradient of the terrain at each contact in order to generate motion plans which avoid stepping on inclined surfaces and edges.

B. Whole Body Control

In order to track the motion plan *set points*, we use the whole-body controller based on the work of [3] which employs a hierarchical optimization approach to compute the feed-forward joint torques. It is important to note that, while the trajectory optimization computes contact forces for generating the trajectories, in our setup, these forces are only used to enforce a feasible motion at the trajectory generation stage and are not used by the whole-body tracking controller. Instead, the WBC takes the feet and base trajectories as inputs and recomputes the contact forces using the full dynamics model of the robot. Moreover, as in the case of

trajectory optimization, we consider the friction cone axis aligns with gravity so as to avoid unstable behaviors.

To improve the robustness during online motion plan tracking, we use the approach detailed in [22] to adapt the PD gains of the actuators and the friction coefficient used in the whole-body controller for a limb whose end-effector slips while in contact with the ground.

The WBC forwards the desired joint states and the feed-forward joint torque in addition to the PD gains (dependent on the foot contact states: support, swing or slip) to the actuator for low-level joint state tracking.

C. Online Trajectory Correction

Despite the introduction of cost terms and contact force constraints in the TOWR problem, the assumptions relating to robot dynamics considered in the formulation to ease on its computational complexity potentially results in generation of highly dynamic trajectories which cannot be tracked by the whole-body controller. Moreover, inaccurate tracking due to system delays in low-level actuation controllers and possible perturbations while interacting with the environment, could result in tracking errors which cannot be corrected by the WBC eventually resulting in a failure. As a solution, we introduce an online trajectory correction aimed at improving the feasibility of the reference trajectory, so as to perform stable whole-body motion tracking. We represent the online trajectory adaptation problem in the framework of a discrete time stochastic Markov decision process (MDP) defined as a tuple (S, A, R, P, μ), where S represents a set of states, A a set of actions, $R : S \times A \times A \to \mathbb{R}$ the reward function, $P : S \times A \times S \to [0,1]$ the state transition probability, and μ the initial state distribution. We define a stationary policy $\pi : S \to \mathcal{P}(A)$ which, in our work, is approximated using a multi-layer perceptron (MLP), as a function mapping states to probability distributions over actions such that $\pi(a|s)$ denotes the probability of selecting action a in state s. We represent the expected cumulative discounted return,

$$J(\pi) \doteq \mathop{\mathrm{E}}_{\mathcal{T}\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right], \tag{1}$$

where $\gamma \in [0,1)$ is the discount factor and $\mathcal{T}$ denotes a trajectory dependent on π. We employ a policy gradient based method, proximal policy optimization (PPO) [23] along with generalized advantage estimate (GAE) [24] so as to obtain a policy π which maximizes $J(\pi)$.

As detailed in Section III, we use the policy π to map a set of robotic system state parameters to actions representing the desired corrective deviations to the reference trajectory at each time step. The modified set point is then tracked by the model-based whole-body controller. Both, the corrective feedback and the whole body control output are computed sequentially at 400 Hz.

III. Training

This section details upon the RL environment setup for training the corrective-tracking policy.

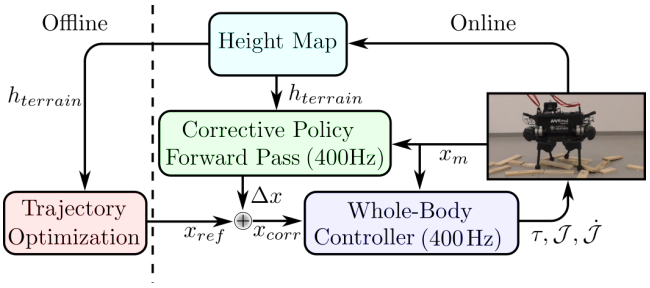

Fig. 2. Block diagram of the trajectory generation and tracking framework.

A. Action Space

The corrective policy outputs a 36-dimensional action vector comprising of $\{\delta P_{base}, \delta \mathcal{O}_{base}, \delta v_{base}, \delta \omega_{base}, \delta P_{feet}, \delta v_{feet}\}$ where the $\{\delta P_{base}\} \in \mathbb{R}^3$ vector represents the deviation in base position, $\{\delta \mathcal{O}_{base}\} \in \mathbb{R}^3$ is the deviation in base orientation, $\{\delta v_{base}\} \in \mathbb{R}^3$ is the deviation in linear base velocity, $\{\delta \omega_{base}\} \in \mathbb{R}^3$ is the deviation in angular base velocity, $\{\delta P_{feet}\} \in \mathbb{R}^{12}$ represents the deviation in the 4 end-effector positions and $\{\delta v_{feet}\} \in \mathbb{R}^{12}$ represents the deviation in the 4 end-effector velocities. At each control step, we introduce these additive deviations to the desired set point obtained from the reference trajectory generated offline.

During training, the policy outputs are sampled from a normal distribution clipped in the range $[-1.0, 1.0]$ with the mean representing the output of the policy network and the varying standard deviation representing policy exploration. We further perform action scaling for these deviations and obtain the scaling factors empirically. It is important to note that, during evaluation, we utilize the policy in a deterministic setting by only considering the output mean.

B. Observation Space

The observation space in our training environment comprises of the proprioceptive sensory information which can be computed by sensors and state-estimators on the physical system. For the task of terrain-aware corrective trajectory tracking, we further utilize an exteroceptive feedback in the form of an encoded representation of the terrain elevation relative to the robot's base.

1) Proprioceptive Sensory Information: The proprioceptive state information consists of a 486-dimensional vector defined as

$$\{\mathcal{C}^{m}_{t,\,t-4,\,t-8}, \mathcal{V}^{m}_{t,\,t-4,\,t-8}, \mathcal{C}^{traj}_{t+8,\,t+4,\,t+1,\,t,\,t-4,\,t-8},$$
$$\mathcal{V}^{traj}_{t+8,\,t+4,\,t+1,\,t,\,t-4,\,t-8}, \mathcal{C}^{corr}_{t-1,\,t-4,\,t-8}, \mathcal{V}^{corr}_{t-1,\,t-4,\,t-8}, \mathcal{C}^{traj}_{t+200}\}$$

where $\mathcal{C}_t$ refers to the generalized coordinates at time t, $\mathcal{V}_t$ refers to the generalized velocities at time t and the superscripts m, $traj$ and $corr$ refer to the measured robot state, reference trajectory set point and the corrected trajectory set point respectively. The delay between $t+1$ and t corresponds to 2.5 ms. All of the state parameters are represented in the robot's base frame. Moreover, the robot base orientation is represented in the observation as a 6-dimensional vector. For

this, we consider the first and the third row of the robot's rotation matrix. This helps avoid the problems associated with discontinuities in the 3-dimensional and 4-dimensional representations of orientation. We introduce the short-horizon goal C_{t+200}^{traj} in order for the corrective policy to generate outputs which prioritize stable tracking of long-term trajectory plans as opposed to aggressive tracking of the reference trajectory.

2) Exteroceptive State Representation: We introduce latent encoding of the height maps in the state vector s. These height maps represent the terrain elevation local to the robot's base. We use 80×80-dimensional height maps, along the robot's heading and lateral axes respectively, corresponding to an area of 1.0×1.0 m^2 centered at the robot's base position. The elevation of the height maps is clipped between $[-2.0, 2.0]$ m.

In order to perform sampling-efficient RL [25] we reduce the dimensionality of our sparse height maps using a convolutional autoencoder network [26] to learn an encoding $h = f(x)$ such that the decoder, approximated as g, is able to regenerate the input $x \approx g(h)$. We then use the encoder network f to encode our 80×80-dimensional height maps into a 60-dimensional representation.

The encoder comprises of 3 convolutional layers each with a kernel size of 3, and a depth mapping from 1 to 16 in the first layer, 16 to 24 in the next and 24 to 32 in the last layer. We append a 60-dimensional dense layer to output the encoded representation of the height maps. For training, we use a decoder network comprising of the deconvolutional layers with reversed depth mapping as that of the encoder convolutional layers to regenerate the original height map from the encoded vector. We train the auto-encoder to minimize the regeneration error using a dataset of procedurally generated height maps. The generation of height maps is detailed in Section III-E.

C. Policy Network

For the combined proprioceptive and exteroceptive state representation, our network input is 546-dimensional, and the output is 36-dimensional. We use 3 hidden layers comprising of 1024, 584 and 256 nodes in the first, second and third layer respectively. Furthermore, we use the hyperbolic tangent non-linear activation for our policy network.

D. Reward Signal

As detailed in our previous work [10], RL algorithms require significant reward function tuning to achieve a desired control behaviour. Therefore, we use some of the reward terms described in [8], [9], [10] that have been demonstrated to result in smooth quadrupedal locomotion behavior. We further add and empirically tune reward terms relevant to our task of correcting trajectories online so as to reach the trajectory horizon. These reward terms are shown in Table I (the notation is consistent with [10]).

One of the critical aspects of our task setup is to ensure the robot remains dynamically stable throughout the trajectory tracking duration. Despite significant research that has been performed to handle sparse rewards [27] in the RL setup, it is still desirable to have a dense reward curve. In this regard, we utilize a stability analysis method, detailed below, to generate a reward signal. In our RL environment setup, we introduce self-collision and collision of the base with another object as a termination criteria. Additionally, we define termination criteria based on thresholds on the joint position, velocity and acceleration and base orientation, velocity and acceleration.

TABLE I

REWARD TERMS USED IN OUR MDP FORMULATION. HERE τ REFERS TO THE JOINT TORQUE, $v_{world,t}^{foot}$ IS THE FOOT VELOCITY IN WORLD FRAME AT TIME t, F_{foot} IS THE FOOT CONTACT FORCE, $\mathcal{J}_t$ IS THE JOINT POSITION AT TIME t, $\mathcal{O}_{x,y,z}^{base}$ IS THE BASE ORIENTATION ALONG THE x,y,z AXES, $f_{h,foot}$ REPRESENTS THE FOOT HEIGHT, f_h^{des} REPRESENTS THE DESIRED FOOT CLEARANCE, P_{base}^{m} REPRESENTS THE MEASURED BASE POSITION, P_{base}^{traj} IS THE DESIRED BASE POSITION OBTAINED FROM THE REFERENCE TRAJECTORY AND $P_{base,t}^{traj,H}$ REPRESENTS THE DESIRED BASE POSITION AT THE SHORT-HORIZON (IN OUR CASE AT $t+200$) IN THE ROBOT'S BASE FRAME AT TIME t.

Term	Expression
Torque	$\|\tau\|^2$
Foot Acceleration	$\|\|v_{world,t}^{foot} - v_{world,t-1}^{foot}\|\|^2$
Foot Slip	$\|\|v_{world}^{foot}\|\|^2 \quad \forall\ F_{foot} > 0$
Smoothness	$\|\|\mathcal{J}_t - \mathcal{J}_{t-1}\|\|^2$
Orientation	$\|\|\mathcal{O}_{x,y,z}^{base} - \{0,0,\mathcal{O}_z^{base}\}\|\|^2$
Joint Velocity	$\|\dot{\mathcal{J}}_t\|^2$
Joint Acceleration	$\|\ddot{\mathcal{J}}_t\|^2$
Foot Clearance	$\sum_{foot}\left(f_{h,foot} - f_h^{des}\right)^2 \left\|v_{world}^{foot}\right\|^2$
Trajectory Tracking	$\|P_{base}^{m} - P_{base}^{traj}\|^2$
Short Horizon Goal	$\sum\{P_{base,t-1}^{traj,H} - P_{base,t}^{traj,H}\}$

In order to assess the stability of the robot we employ a criterion based on the feasible region [28] which depends on the endogenous properties of the robot, such as joint-torque limits and legs' dynamic model, on exogenous properties such as possible external wrenches applied to the base of the robot and on environmental conditions such as the friction coefficient and the orientation of the surface in the contact location.

We define the stability margin $m \in \mathbb{R}$ as the signed distance between the instantaneous capture point (ICP) ζ [29]:

$$\zeta = \mathbf{c}_{x,y} + \dot{\mathbf{c}}_{x,y}\sqrt{\frac{c_z}{g}} \tag{2}$$

and the edges of the feasible region. Here, $\mathbf{c}_{x,y} \in \mathbb{R}^2$ refers to the horizontal position component of the robot's CoM. The stability margin is positive when the ICP ζ lies within the feasible region and negative if outside. We define the robot to be dynamically stable whenever $m > 0$. We then introduce a reward term given by $\max(m, m_{max})$ so as to maximize the stability margin m. For ANYmal, we used $m_{max} = 0.13$.

E. Rough Terrain Simulation

We use the RaiSim [30] physics simulator to train our RL policy. For a detailed analysis on our choice of the simulation

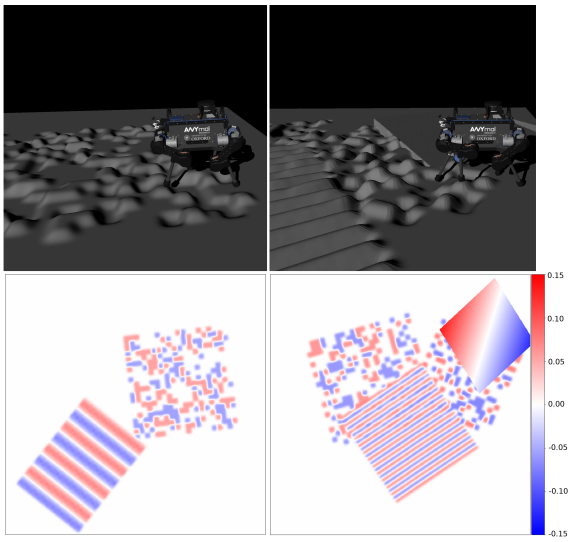

Fig. 3. *top:* Terrain visualization in the RaiSim simulator. *bottom:* The corresponding 16-bit PNG images representing the height maps. The colorbar represents the terrain elevation in m.

platform, we defer the reader to our previous work [10].

In order to obtain a robust policy for terrain-aware reference trajectory tracking we train our RL policy across a multitude of procedurally generated terrains. RaiSim lets the user import portable network graphics (PNG) images and converts them into height maps. Users also have the flexibility to scale and offset these height maps during simulation runs. We leverage this feature of the simulator to randomize our simulation environment and perform trajectory optimization over multiple terrains. We store the trajectories obtained for each of the terrains for multiple configurations of the robot's initial and final positions in a replay buffer and utilize them during training. We randomly sample the stored trajectories and reset the simulation environment for corresponding terrain and robot pose, and perform multiple training runs over a single trajectory. This enables us to ensure that the trained RL policy generalizes over multiple terrains for different reference trajectories. For training our RL policy we used a total of 1000 different simulated terrains.

Each height map contains at least 1 and at most 4 objects randomly positioned and rotated within the image area. These objects include staircase, planks, bricks, unstructured terrain and wave terrain. Examples of these images and the corresponding height maps are represented in Fig. 3.

F. Policy Transfer to the Physical System

In order to make the trained policy robust and generalizable to unaccountable factors we introduce several domain randomization methods during training. We adapt some of the strategies we used in our previous work [10]. Moreover, as detailed in [10], an accurate actuation model significantly effects the behavior of the RL policy and also helps ease the policy sim-to-real transfer process. In this regard. we employ the following schemes during training.

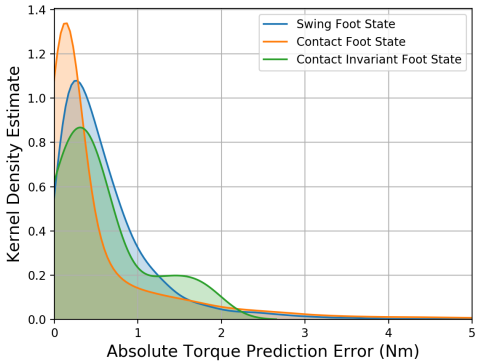

Fig. 4. The kernel density estimate plot representing the torque prediction error for different foot contact states.

1) Actuator Modelling: As introduced in [8], we train an actuator network to model the actuation dynamics of the physical system. However, unlike in the case of [8], [10] where the actuators were modelled only for joint position targets with a predefined set of PD gains, in this work, we trained the actuator network using supervised learning to take as inputs the joint position and velocity targets along with the desired feed-forward torques. Since the whole-body controller utilizes different set of PD gains for joints of legs with foot in contact, swing and contact-invariant (slip) states, we further include the foot contact state information as an input to the approximated actuator model. The actuator network consists of an 18-dimensional input given by the vector

$$\{\delta\mathcal{J}^i_{t,t-4,t-8}, \delta\dot{\mathcal{J}}^i_{t-1,t-5,t-9}, \delta\tau^i_{t-1,t-5,t-9}, \dot{\mathcal{J}}^{des,i}_{t,t-4,t-8}, \tau^{des,i}_{t,t-4,t-8}, F^{state,f}_{t,t-4,t-8}\}$$

where $\delta\mathcal{J}^i_t$ represents the error between the measured and desired joint position for joint i at time t, $\delta\tau^i_t$ represents the error between the measured and desired feed-forward joint torque for joint i at time t, $\dot{\mathcal{J}}^{des,i}_t$ represents the desired joint velocity for joint i at time t, $\tau^{des,i}_t$ represents the desired feed-forward joint torque for joint i at time t and F^{state}_t is the contact state of foot f at time t. The duration between $t+1$ and t corresponds to 2.5 ms. The network contains 2 hidden layers with 48 nodes in each of the layers and a 1-dimensional output layer representing the estimate of the joint torques observed on the physical system. We forward pass through the network at each time step for each of the 12 joints of the quadrupedal robot.

To train the actuator network, we collected data from the real system using the *dynamic gaits* controller [31]. We used dynamic gaits to generate the desired joint positions, velocities and feed-forward torques in order to track desired reference base velocity commands. We executed the controller at 400 Hz and recorded the actuation commands, the measured joint states and the measured torque for each of the 12 actuators at each control step. As a result, we generated a dataset comprising of 4.32M samples in 15 minutes. This

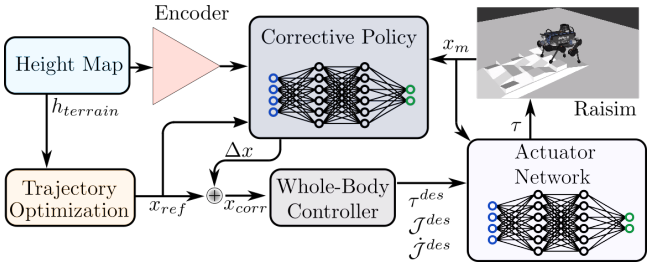

Fig. 5. Block diagram representing the modules that constitute the RL training environment of the corrective policy.

was sufficient to approximate the actuation dynamics using a neural network. As represented in Fig. 4, the absolute mean error for torque prediction using an actuator network is less than 0.8 N·m for each of the foot contact states.

2) Shifting Initial Position: Upon environment resets, we randomly shift the base position along the horizontal axes with a mean of 0 m and a standard deviation of 0.03 m.

3) Changing Gravity: We uniformly sample acceleration due to gravity between $[0.95g, 1.05g]$, where $g = 9.81\,\text{m}\cdot\text{s}^{-2}$ to emulate inertial scaling.

4) Actuator Torque Scaling: We randomly scale the output torque of our actuator network with the scaling coefficient $s_t \in [0.95, 1.05]$ to account for differences between the real actuators and the approximated model.

5) Adding Actuator Damping: We emulate actuator damping using a complementary filter given as $\mathcal{J}_t^{des'} = K_{damp}\mathcal{J}_t^{des} + (1 - K_{damp})\mathcal{J}_{t-1}^{des'}$ where the gain K_{damp} is randomized between $[0.85, 1]$.

6) Perturbing the Robot Base: We apply external forces to the robot's base along its heading and lateral axes during trajectory tracking. This enables the policy to learn a recovery behavior so as to maximize the robot stability margin.

We sample the external forces from a normal distribution with a mean 0 N and a standard deviation of 15 N. We apply these forces for duration sampled randomly between $[1, 3]$ s.

IV. Results

Training the terrain-aware corrective policy required 3.5B simulation steps with computation of 29.82M steps per hour on a desktop housing an Intel i7-8700K and an Nvidia RTX 2080Ti with 12 parallel environment executions. Additionally, we trained a blind corrective policy for which we do not include the exteroceptive feedback in the observation space in less than 250M simulation steps for locomotion over flat ground. We tested its performance in simulation and on the physical system.

A. Simulation results

For evaluation of the corrective policy we compare the locomotion behaviors obtained by the baseline motion generation strategy (offline trajectory optimization with WBC) with the same control architecture enhanced by our proposed online corrective RL policy. Table II represents the success rate, defined as the percentage of experiments that lead the robot to perform the whole trajectory without eventually falling down, observed over 1000 runs each for tracking pre-computed reference trajectories over 3 different terrains - flat, modular pallet and stacked pallet. In this experiment, we randomly shifted the initial base position of the robot along the heading and lateral axes by a mean of 0 m and a standard deviation of 0.08 m. The run was considered a success if the robot managed to reach the trajectory horizon with a base tracking error of less than 0.05 m along the horizontal axes and remained at the goal for 3 seconds without resulting in termination. We considered the run a failure if any contact-pairs apart from the feet and ground were detected. In case of flat terrain, we observed that most of the failures occurred during high base-position tracking error phases where the WBC aggressively attempted to correct the base position tracking error without recomputing plans for feet motions. For the case of the corrective policy, we observed trajectory tracking with extended feet position directly corresponding to a higher stability margin. This behavior is evident from Fig. 7 which represents the 2nd order regressive approximation of the modified trajectory and measured feet position along the x and y axis of the world frame for 1000 runs of the robot tracking a 1.5 m long pre-computed trajectory with actuation torque output randomly scaled between $[0.975, 1.02]$.

TABLE II

Success rates observed for tracking reference trajectories with and without feedback for 1000 runs each over 3 different terrains. The initial base position along the heading and lateral axes is randomly shifted by a mean of 0 m and a standard deviation of 0.085 m

	No Feedback	Blind Feedback	Perceptive Feedback
Flat	72.7	**94.6**	92.4
Modular Pallet	47.5	59.3	**80.3**
Stacked Pallet	38.8	46.2	**67.1**

For the case of locomotion over modular pallet, we observed failures mainly due to foot slip. The addition of the corrective policy resulted in a recovery behavior which we also observed during tests on the physical robot as shown in Fig. 8. In the case of the stacked pallet problem, we observed that the physical dimension of the robot was a limiting factor. The failure occurred mostly due to the base colliding with the limbs. The extended stance introduced by the corrective policy resulted in a higher success rate.

We further conducted experiments to test the ability of the corrective policy to perform state recovery. We used an offline pre-computed trajectory 1000 times over different terrains. The terrains were generated as an undulated environment where each bump and gap (see Fig. 3) has a different height sampled from a random uniform distribution in the range $[-h_{max}, h_{max}]$ where h_{max} represents the maximum terrain height. In all cases the WBC is required to track a reference base and feet trajectory corresponding to 16 steps, for a total distance of 1.5 m and a duration of 3.5 s. This motion plan is optimized offline for flat ground and is therefore not aware of the undulated terrain. Fig. 6 (left)

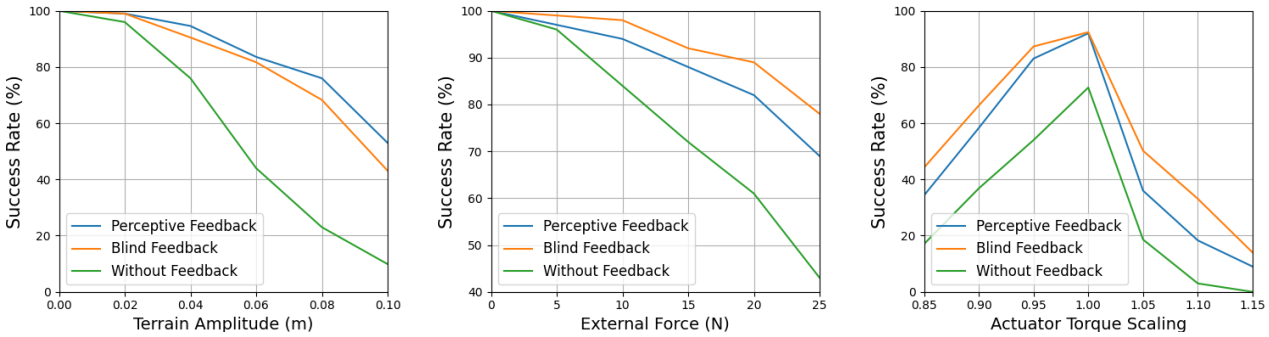

Fig. 6. ***left:*** Success-rate observed for tracking pre-computed trajectory on undulated terrains with varying height for 1000 runs each. ***middle:*** Success rate obtained for locomotion on flat ground in presence of unexpected external forces. ***right:*** Success rate observed for tracking reference trajectory on flat ground for different actuation torque scaling.

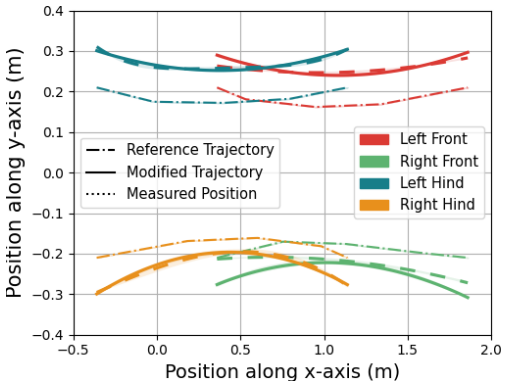

Fig. 7. Comparison of the feet position computed using a trajectory optimization solver with the 2nd order regressive approximation of the trajectory modified online by a corrective RL policy.

represents the success rate observed for h_{max} ranging between $[0.0, 0.05]$ m.

Figure 6 (middle) represents the success-rate obtained during the execution of a predefined trajectory of 3.5 s on flat terrain while perturbing the robot with a horizontal external force for a duration of sampled in the range $[1.5, 2.5]$ s. The intensity of this force is gradually increased with a step of 5 N in the range of $[0, 25]$ N. Also the start time of application of the external disturbance since the beginning of the run was randomly sampled in the range $[1.75, 2.25]$ s. Each success rate was computed using 1000 runs. We observed that the blind corrective policy performed better than the perceptive policy. This was largely due to the blind policy having significantly more experience to disturbances over flat terrain during training.

Figure 6 (right) shows the success rate obtained when we scaled the actuation torque generated by the actuator network by a scaling factor between $[0.85, 1.15]$ while tracking a pre-computed reference trajectory on flat-terrain for 1000 runs each. The effective joint torque was limited between $[-40, 40]$N·m and the initial robot base position was randomly shifted along the heading and lateral axes by a mean of 0 m and a standard deviation of 0.08 m. We observed that reducing the joint torque results in WBC failing to effectively track the desired set points whereas increasing the torque results in aggressive tracking and overshooting eventually resulting in failure.

B. Hardware results

We transferred the blind policy to the ANYmal robot and tested its performance for tracking a pre-computed reference trajectory of 3.5 s on flat ground. The reference trajectory used for this first test is highly dynamic wherein the robot travels 2.0 m in 3.5 s and achieves a maximum velocity of 0.7 m·s^{-1} with significant acceleration and deceleration along the motion plan. For the case without feedback tracking, we observed a success rate of 33.33% over 15 trials. With the introduction of blind feedback, we observed a success rate of 60.0%. We further tested the performance of our policy to recover from an undesirable state by introducing perturbations. Figure 1 shows the ANYmal robot walking over wooden tiles using the corrective policy. In this experiment, we observed that without feedback, the WBC failed to track the reference trajectory for each of the 5 trials. In comparison, for corrective tracking, we observed a success rate of 40% with most failures occurring due to foot slippage. We further introduced an unexpected wooden block along the robot's path and observed no success without feedback. We observed a success rate of 66.67% using the RL policy with most of the failures occurring due leg blockage. Furthermore, Fig. 8 presents an example of a trial that almost fails due to foot slippage but eventually succeeds to recover.

V. Conclusion and Future Work

In this work we presented a control architecture to perform online corrective trajectory tracking. We observed a higher success rate for the case of tracking trajectories with feedback in both simulation and on the real robot. However, we observed that executing the policy on the physical system necessitates a WBC which is robust to physical parameters such as contact and actuation friction. Therefore, our future work includes training an RL policy which outputs actuator tracking gains in addition to the deviation in set points.

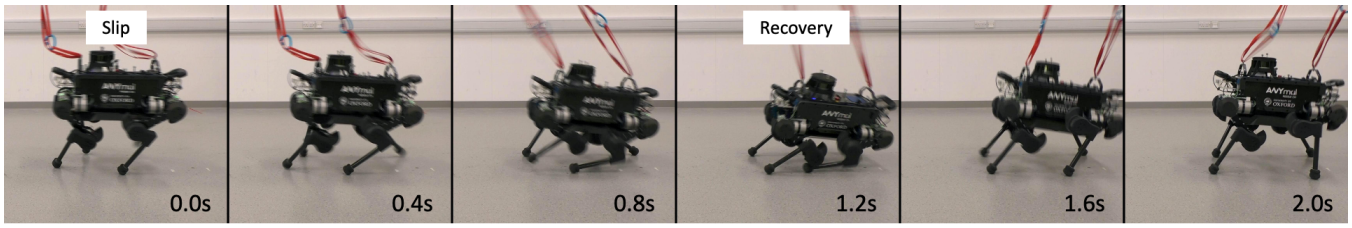

Fig. 8. Time frame of the robot recovering with the corrective controller.

Despite having been tested with motion plans generated offline, our control architecture can be extended for use with motion planners which optimize trajectories online in an MPC manner. As part of future work, we aim to obtain a perceptive policy which functions with asynchronous sensory feedback for execution on the physical system.

REFERENCES

[1] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 6316–6323.

[2] O. Villarreal, V. Barasuol, P. Wensing, and C. Semini, "MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion," in *Proc. IEEE International Conference on Robotics Automation (ICRA)*, Paris, France, May 2020.

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

[4] 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, July 2018.

[5] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2986–2993.

[6] O. Melon, M. Geisert, D. A. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," 2019.

[7] A. Bratta, R. Orsolino, M. Focchi, , V. Barasuol, G. G. Muscolo, and C. Semini, "On the Hardware Feasibility of Nonlinear Trajectory Optimization for Legged Locomotion based on a Simplified Dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[8] 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, p. eaau5872, 2019.

[9] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[10] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *arXiv preprint arXiv:2002.09676*, 2020.

[11] 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.

[12] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[13] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 76–105, 2012.

[14] K. Doya, K. Samejima, K.-i. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural computation*, vol. 14, no. 6, pp. 1347–1369, 2002.

[15] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1–9. [Online]. Available: http://proceedings.mlr.press/v28/levine13.html

[16] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 528–535.

[17] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 7559–7566.

[18] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2986–2993.

[19] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6023–6029.

[20] Z. Xie, G. Berseth, P. Clary, J. W. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," *CoRR*, vol. abs/1803.05580, 2018. [Online]. Available: http://arxiv.org/abs/1803.05580

[21] A. Wächter and L. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, 01 2004.

[22] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4170–4176, Oct 2019.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[24] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015.

[25] S. S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudík, and J. Langford, "Provably efficient rl with rich observations via latent state decoding," *arXiv preprint arXiv:1901.09018*, 2019.

[26] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[27] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *arXiv preprint arXiv:1610.04286*, 2016.

[28] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, "Feasible Region: an Actuation-Aware Extension of the Support Region," *IEEE Transactions on Robotics (TRO)*, 2020.

[29] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, Dec 2006, pp. 200–207.

[30] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[31] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.