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, and show that introducing the corrective feedback results in increase of the success rate from 72.7% to 92.4% for tracking precomputed dynamic long horizon trajectories on flat terrain and from 47.5% to 80.3% on a complex modular uneven terrain. We also show successful transfer of our training approach to the real physical system and further present cogent arguments in support of our framework.**

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 guesses [5], [6], [7] or by introducing feasibility constraints in order to account for the dynamic limits of the actual hardware [8]. 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.

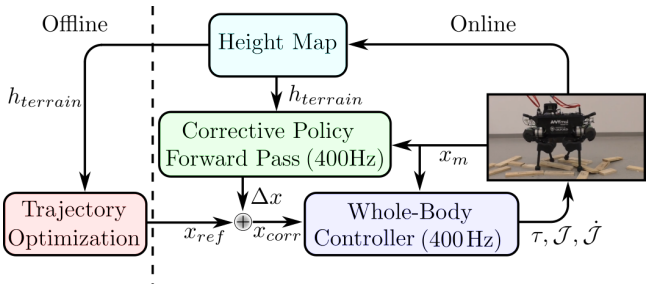

Fig. 1. Block diagram of the trajectory generation and tracking framework. Experiments were performed using the ANYmal B quadruped. The accompanying video can be found here: https://youtu.be/Ve4SDl1wI9s.

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 learning based approaches for quadrupedal locomotion [9], [10], [11] 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. RL, in particular, has been further employed for locomotion over uneven terrain [12], [13] demonstrating impressive locomotion behaviour. However, in order to be used in safety-critical environments, the RL training setup necessitates constrained exploration and control behaviour [14] which requires significant platform-specific engineering thereby increasing the complexity of the RL training environment.

In this work, we consider the problem of tracking dynamic long horizon motion plans generated using a trajectory optimization solver, TOWR [4]. Unlike MPC-based short horizon motion plans which can be recomputed online to recover from perturbations, long horizon motion plans are computationally expensive for online replanning and require precise whole-body tracking. As a result, the performance of controllers employing such long horizon motion plans is often lower than that of MPC-based controllers.

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

In this regard we present an approach to improve the performance of tracking long horizon motion plans by introducing a new control architecture which utilizes long horizon motion plans generated using TOWR, corrected online using an RL policy and 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 online with the objective of prioritizing stability over tracking extremely dynamic (and possibly infeasible) trajectories in addition to directing the robot towards the desired long horizon goal. We evaluate the performance of our control architecture using the ANYmal [15] quadruped, both, in simulation and on the physical robot.

II. Related Work

A significant amount of research in RL and optimal control (OC) domains has focused on tackling shared objectives using a unified approach [16], [17] including contributions in the form of model-based RL [18], guided policy search (GPS) [19], [20] and even a reformulation of the stochastic optimal control problem in terms of KL divergence minimization [16].

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 [21] and also for legged locomotion [22]. In the case of [22], 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.

Additionally, quasi-static control approaches such as model-based FreeGait [23] have been proposed for terrain-aware locomotion. In contrast, the trajectory optimization solver, TOWR, used in our work generates dynamic whole-body motion plans. Furthermore, as opposed to recent work on RL for dynamic locomotion over challenging [13] and uneven terrain [24] which focus on short horizon planning and control, in this work, we address the issues with tracking dynamic long horizon plans by performing online trajectory adaptation while tracking reference trajectories generated offline.

III. 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. 1.

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 [25].

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.

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.

To improve the robustness during online motion plan tracking, we use the approach detailed in [26] 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

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) [27] along with generalized advantage estimate (GAE) [28] so as to obtain a policy π which maximizes $J(\pi)$.

As detailed in Section IV, 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.

IV. Training

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

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.

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 terrain-aware corrective trajectory tracking, we also utilize an exteroceptive feedback in the form of an encoded representation of the terrain elevation.

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. We introduce the short-horizon goal $\mathcal{C}^{traj}_{t+200}$ 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.

TABLE I

Reward terms used in our MDP formulation. Here τ refers to the joint torque, $v^{foot}_{world,t}$ 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, $\mathcal{O}^{base}_{x,y,z}$ is the base orientation along the x,y,z axes, $f_{h,foot}$ represents the foot height, f^{des}_h represents the desired foot clearance, P^{m}_{base} represents the measured base position, P^{traj}_{base} is the desired base position obtained from the reference trajectory and $P^{traj,H}_{base,t}$ 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	$\|\|v^{foot}_{world,t} - v^{foot}_{world,t-1}\|\|^2$
Foot Slip	$\|\|v^{foot}_{world}\|\|^2 \quad \forall\ F_{foot} > 0$
Smoothness	$\|\|\mathcal{J}_t - \mathcal{J}_{t-1}\|\|^2$
Orientation	$\|\|\mathcal{O}^{base}_{x,y,z} - \{0,0,\mathcal{O}^{base}_{z}\}\|\|^2$
Joint Velocity	$\|\dot{\mathcal{J}}_t\|^2$
Joint Acceleration	$\|\ddot{\mathcal{J}}_t\|^2$
Foot Clearance	$\sum_{foot} \left(f_{h,foot} - f^{des}_h\right)^2 \left\|v^{foot}_{world}\right\|^2$
Trajectory Tracking	$\|P^{m}_{base} - P^{traj}_{base}\|^2$
Short Horizon Goal	$\sum\{P^{traj,H}_{base,t-1} - P^{traj,H}_{base,t}\}$

In order to perform sample-efficient RL [29] we reduce the dimensionality of our sparse height maps using a convolutional autoencoder network [30] 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.

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

The reward terms used for training the corrective policy are shown in Table I (the notation is consistent with [14]).

One of the critical aspects of our task setup is to ensure the robot remains dynamically stable throughout the trajectory

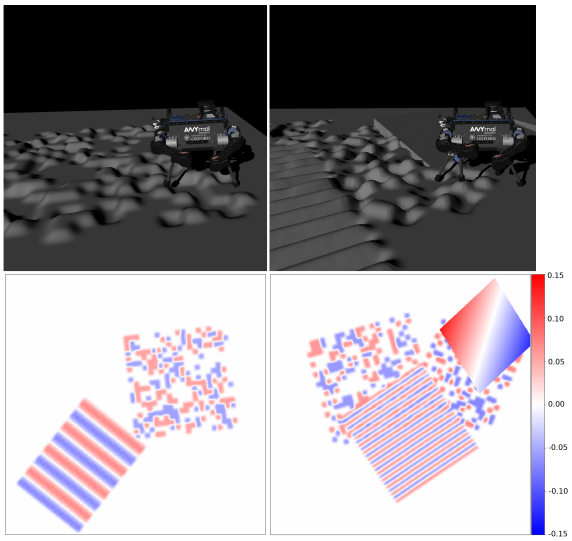

Fig. 2. *top:* Terrain visualization in the RaiSim simulator. *bottom:* The corresponding 16-bit PNG images representing the height maps. The colorbar represents the terrain elevation in m.

tracking duration. Despite significant research that has been performed to handle sparse rewards [31] 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. To assess the stability of the robot we employ a criterion based on the feasible region [32] which depends on the endogenous and exogenous properties of the robot, and also on environmental conditions such as the friction coefficient and the surface normals.

We define the stability margin $m \in \mathbb{R}$ as the signed distance between the instantaneous capture point (ICP) [33] and the edges of the feasible region. The stability margin m is positive when the ICP lies within the feasible region and negative if outside. We define the robot to be dynamically stable whenever $m > 0$. We thus introduce a reward term given by $\max(m, m_{max})$ so as to maximize the stability margin. For ANYmal, we used $m_{max} = 0.13$.

E. Rough Terrain Simulation

We use the RaiSim [34] physics simulator to train our RL policy. 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. We randomize our simulation environment and perform trajectory optimization over multiple terrains. 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. We generate these terrains using 2-dimensional height maps as detailed in [24]. Examples of these height maps and the corresponding simulated terrains are represented in Fig. 2.

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. These are described below.

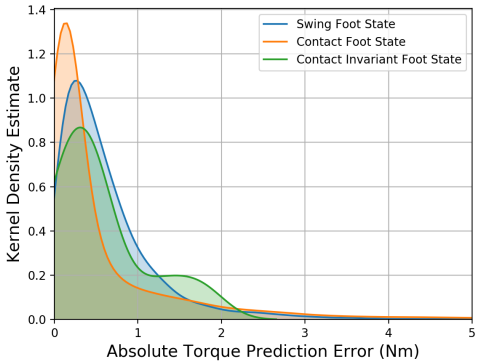

Fig. 3. The kernel density estimate plot representing the torque prediction error for different foot contact states.

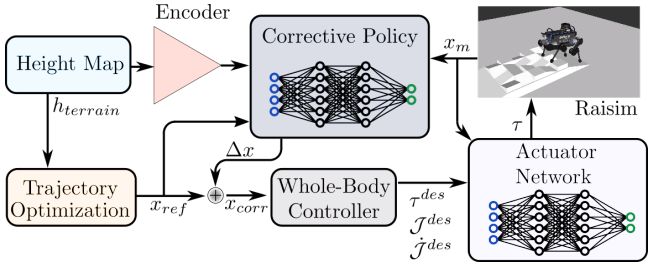

Fig. 4. Block diagram representing the modules that constitute the RL training environment of the corrective policy.

1) Actuator Modelling: As introduced in [9], we train an actuator network to model the actuation dynamics of the physical system. However, unlike in the case of [9], [14] 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.

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**

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) 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.

V. RESULTS

Training the terrain-aware corrective policy required 3.5B simulation steps with computation of 29.82M steps per hour on a standard desktop computer. 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 behaviour 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 $[0.975, 1.02]$.

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 behaviour which we also observed during tests on the physical robot as shown in Fig. 5. In the case of the stacked pallet problem, we observed that the physical dimension of the robot was a limiting factor.

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. 2) 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) 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. 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.

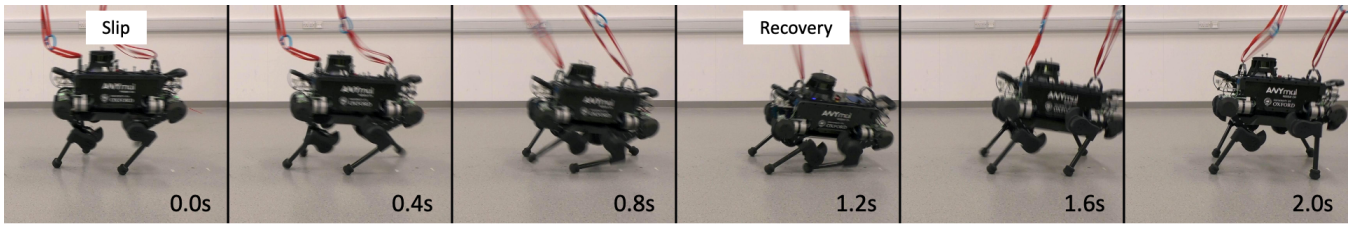

Fig. 5. Time frame of the robot recovering with the corrective controller.

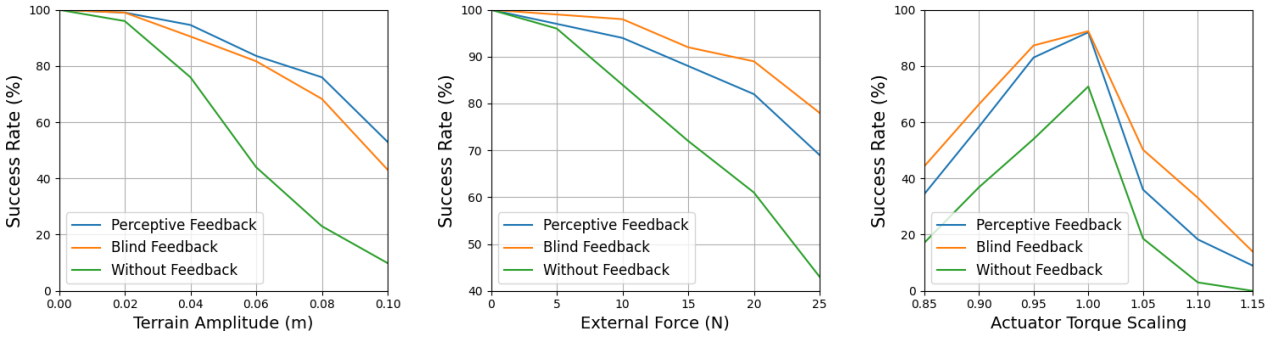

Fig. 6. ***left:*** Success-rate observed for tracking pre-computed trajectory on undulated terrains with varying height for 1000 runs each. ***middle:*** Success rate obtained for locomotion on flat ground in presence of unexpected external forces. ***right:*** Success rate observed for tracking reference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

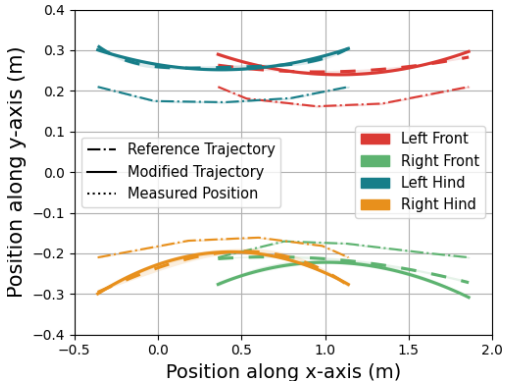

Fig. 7. Comparison of the feet position computed using a trajectory optimization solver with the 2nd order regressive approximation of the trajectory modified online by a corrective RL policy.

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 $\mathrm{m{\cdot}s^{-1}}$ 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%. Note that the poor performance associated with tracking dynamic long horizon motion plans is due to accumulation of tracking errors which the whole-body controller then aggressively attempts to correct. For the case of less dynamic trajectory tracking wherein the robot travels 1.5 m in 3.5 s, we observed a success rate of 80.0% and 100.0% for open-loop and feedback-based tracking respectively. We further tested the performance of our policy to recover from an undesirable state by introducing perturbations. We tested 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. 5 presents an example of a trial that almost fails due to foot slippage but eventually succeeds to recover.

VI. Conclusion and Future 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. 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. Additionally, we observed that executing the policy on the real robot requires a WBC which is robust to actuation dynamics. Therefore, our future work includes training an RL policy which also outputs actuator tracking gains. We also 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] D. Surovik, O. Melon, M. Geisert, M. Fallon, and I. Havoutis, "Learning an Expert Skill-Space for Replanning Dynamic Quadruped Locomotion over Obstacles," in *2020 Conference on Robot Learning (CoRL)*, Nov. 2020.

[8] 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.

[9] 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.

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

[11] A. L. Mitchell, M. Engelcke, O. P. Jones, D. Surovik, S. Gangapurwala, O. Melon, I. Havoutis, and I. Posner, "First steps: Latent-space control with semantic constraints for quadruped locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5343–5350.

[12] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[13] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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

[20] 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.

[21] 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.

[22] 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

[23] P. Fankhauser, C. Dario Bellicoso, C. Gehring, R. Dubé, A. Gawel, and M. Hutter, "Free gait — an architecture for the versatile control of legged robots," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1052–1058.

[24] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," 2020.

[25] 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.

[26] 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.

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

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

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.