Learning Locomotion Skills from MPC in Sensor Space

Majid Khadiv MKHADIV@TUEBINGEN.MPG.DE
Max-Planck Institute for Intelligent Systems, Tübingen, Germany

Avadesh Meduri AM9789@NYU.EDU
Tandon School of Engineering, New York University, USA

Huaijiang Zhu HZHU@NYU.EDU
Tandon School of Engineering, New York University, USA

Ludovic Righetti LUDOVIC.RIGHETTI@NYU.EDU
Tandon School of Engineering, New York University, USA

Bernhard Schölkopf BS@TUEBINGEN.MPG.DE
Max-Planck Institute for Intelligent Systems, Tübingen, Germany

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Nonlinear model predictive control (NMPC) is one the most powerful tools for generating control policies for legged locomotion. However, the large computation load required for solving optimal control problem at each control cycle hinders its use for embedded control of legged robots. Furthermore, the need for a high-quality state estimation module makes the application of NMPC in real world very challenging, especially for highly agile maneuvers. In this paper, we propose to use NMPC as an expert and learn control policies directly from proprioceptive sensory measurements. We perform an extensive set of simulations on the quadruped robot Solo12 and show that it is possible to learn different gaits using only proprioceptive sensory information and without any camera or lidar which are normally used to avoid drift in state estimation. Interestingly, our simulation results show that with the same structure of the function approximators, learning estimator and control policy separately outperforms end-to-end learning of dynamic gaits such as jump and bound. A summary of simulation experiments can be found here.

Keywords: Agile locomotion, Control in sensor space, learning from MPC.

1. Introduction

Nonlinear model predictive control (NMPC) has recently shown great promise in generating agile and robust locomotion skills Mastalli et al. (2022); Grandia et al. (2022); Meduri et al. (2023). In this paradigm, an optimal control problem is solved at each control cycle, using the current estimate of the states, to generate optimal control actions. Performing this online generation of optimal policies comes at the cost of three main issues; 1) the online computation burden is very large which makes it impractical for embedded systems, 2) the non-convexity of the underlying optimization problem makes it always possible that the solutions converge to a poor local minima which can be catastrophic on real hardware, 3) there is a need for a high-quality state estimation module to provide drift-free estimate of the states. To circumvent these three main issues, in this paper we aim at using NMPC as an expert and learning the optimal control policy directly from proprioceptive sensory measurements.

Thanks to the recent success in the use of reinforcement learning (RL) for controlling legged robots in the real world Hwangbo et al. (2019); Xie et al. (2020), the past few years have witnessed an explosion of the use of reinforcement learning for generating robust locomotion policies for legged robots Siekmann et al. (2021); Li et al. (2021); Bogdanovic et al. (2022); Aractingi et al. (2022). While earlier works relied on an extra state estimation module to provide the control policy with the states, the latest works have achieved impressive results with learning control policies directly from sensory measurements Lee et al. (2020); Miki et al. (2022); Agarwal et al. (2022). In these approaches, first a teacher policy is trained through RL using privileged information in the simulation. Then a student policy is trained in a supervised fashion to mimic the latent space and action of the teacher policy, relying only on a short history of the sensory measurements. However, learning the teacher policy through reinforcement learning is highly time-consuming and without making the policy output high-level abstract parameters such as stepping frequencies and foot position residuals, these approaches need heavy reward shaping for each single task. This paper aims at mainly replacing the training of the teacher policy with a more efficient approach which is imitation learning from NMPC. The other difference of the presented work compared to those teacher-student approaches is that our latent space is a set of interpretable physical states of the system.

Learning from NMPC and from sensory information has been done in Levine et al. (2016) for manipulation. The main observation space in that work is vision and it is assumed that the objects to be manipulated are fully observed, hence the system is fully observable. This is different from the locomotion problem where the base position is not directly observable and need to be inferred from noisy IMU measurements (base linear acceleration and angular velocity) and joint encoder measurements. This is especially an issue when the robot has flight phase and experience intermittent impacts with the environment. Most recent works that implemented these types of agile locomotion behaviours on real robots fused proprioceptive measurements with camera or lidar to avoid issues state estimation drift Mastalli et al. (2022); Dhédin et al. (2022). In this paper, we aim at learning these skills by relying only on proprioceptive measurements.

Recently, Carius et al. (2020); Viereck et al. (2022) learned locomotion skills using demonstrations from NMPC. Both approaches tried to keep the structure of the optimal control problem in the form of learning control Hamiltonian and value function, respectively. However, this will add a computation burden in the execution time to compute the actions from the learned components. Compared to these approaches, in this work we would like to minimize the computation at run-time, hence we investigate if it is feasible to learn a policy to output values that can be transferred to action with minimum computation at run-time. Another difference of our work with respect to Carius et al. (2020); Viereck et al. (2022) is that we learn different locomotion skills in the sensor space.

The main contributions of the paper are as follows:

- We show that it is possible to learn locomotion policies from NMPC without keeping any structure of the optimal control problem and learn a wide range of gaits for a quadruped.
- We compare different formulations of action space and show that the policy that outputs set points for a fixed PD controller outperforms other formulations including torque policy.
- We show that it is possible to learn highly dynamic gaits (jump and bound) with only proprioceptive sensory measurements, i.e. IMU and joint encoders. This is in contrast with the state-of-the-art NMPC implementations that need camera or lidar to have drift-free pose estimates of the robot base (mainly height), which is crucial for the success of the controller in dynamic locomotion tasks.

- We show that with the same structure of the neural networks, learning an estimator that explicitly infers the physical system states from sensor measurements and using those as the policy input outperforms learning an end-to-end policy mapping from the sensor space to the action space.

2. Nonlinear model predictive control

In this section, we briefly explain the NMPC formulation Meduri et al. (2023) we use in this paper as our expert. For a given fixed gait, a desired reference motion velocity, and time of each phase, a contact plan is automatically generated which specifies the location of the robot end-effectors at each point in time. Our NMPC decomposes the whole body trajectory generation problem into a kinematic and dynamic optimizer. In this setting, given a contact plan, the dynamic optimizer takes into account the centroidal momentum dynamics and constructs a finite-horizon optimal control problem to find a feasible set of contact forces and centroidal trajectories for the given contact plan

$$\min_{\mathbf{X}_t;\mathbf{F}_t} \quad \sum_{t=0}^{T-1} {}_t(X_t;F_t) + {}_T(X_T;F_T) \tag{1a}$$

$$\text{s.t.} \quad c_{t+1} = c_t + \dot{c}_t \quad t \tag{1b}$$

$$\dot{c}_{t+1} = \dot{c}_t + \sum_{j=1}^{N} n_t^j \frac{f_t^j}{m} \quad t + g \quad t \tag{1c}$$

$$k_{t+1} = k_t + \sum_{j=1}^{N} n_t^j ((p_t^j \quad c_t) \quad f_t^j) \quad t \tag{1d}$$

$$_{t;j}; \quad \sqrt{(f_{t;x}^j)^2 + (f_{t;y}^j)^2} \quad f_{t;z}^j \quad ; f_{t;z}^j \quad 0 \tag{1e}$$

$$_{t;j}; \; p_t^j \quad ; \quad _t 2 \quad ; \quad c_0; \dot{c}_0 = c_{init}; \dot{c}_{init} \tag{1f}$$

where c represents the center of mass CoM location, k is the angular momentum around the CoM, m is the robot mass, g is the gravity vector, n_j is a binary integer that describes whether the end-effector j is in contact, $f_j; p_j$ are the end-effector force and location respectively (assuming point-contact end-effectors). ${}_t(X_t;F_t)$ is the running cost, ${}_T(X_T;F_T)$ is the terminal cost, and $X_t = [c_t;\dot{c}_t;k_t;:::;g], F_t = [f f_{t;x}^j;f_{t;y}^j;f_{t;z}^j;:::;g]$. Furthermore, t is the time discretization, is the friction coefficient, is the set of all allowed stepping locations, are kinematic constraint s written as bounds on the CoM position, $c_{init}; \dot{c}_{init}$ are the initial conditions for the CoM.

While (1d) is nonlinear and makes the whole optimization problem non-convex, our solver Meduri et al. (2023) takes the bi-convex structure of the problem and solves it efficiently using alternating direction method of multipliers (ADMM). Also the second order cone constraint in (1e) is handled using a first order solver based on proximal operators Meduri et al. (2023). To track these momentum trajectories, while penalizing contact constraints, we solve a whole-body kinematic optimal control problem using DDP and solve it using the Crocoddyl Mastalli et al. (2020) to output

the desired whole-body trajectories. We formulate the problem as

$$\min_{q;v;\underline{v}} \sum_{t=0}^{T} {}^{\text{mom}}_{t}(l_t; k_t) + {}^{\text{CoM}}_{t}(c_t) + {}^{\text{eff}}_{t}(q_t; v_t) + \|\underline{v}\|$$
$$\text{s.t.} \quad q_{t+1} = q_t \quad v_t \quad t; \quad v_{t+1} = v_t + \underline{v}_t \quad t \tag{2}$$

where, ${}^{\text{mom}}_{t}(l_t; k_t)$ is a momentum cost that tracks the optimal linear and angular momentum computed by the centroidal OCP (1), ${}^{\text{CoM}}_{t}(c_t)$ is the center of mass tracking cost with the optimal CoM trajectory (c_t) obtained from the centroidal OCP, ${}^{\text{eff}}_{t}(q_t; v_t)$ is the end-effector locations and velocity cost, and $\|\underline{v}\|$ is a penalty on the control. Finally, $\sum$ stands for summation over $SE(3)$ objects.

The kinematic and dynamic optimizers re-compute trajectories at 20 Hz and the joint torques are computed at 1 kHz and are sent to the robot. The contact forces from the dynamics optimizer and whole-body joint trajectories from kinematic optimizer are then used to compute feedforward torques (^{ff}) using full dynamics of the system. Finally, a low-impedance controller is added to the feedforward torques of each joint to account for model errors and uncertainties

$$_{t} = {}^{ff}_{t} + \underbrace{k_p(q^d_t \quad q_t) + k_v(v^d_t \quad v_t)}_{{}^{fb}_{t}} \tag{3}$$

where k_p and k_v are the position and velocity tracking gains for each joint which are xed. The superscript d stands for the desired trajectory that comes from the kinematic optimizer (2).

3. Learning

In this work, we consider the simplest form of imitation learning, i.e., behavioral cloning which casts the learning problem from experts demonstrations as a supervised learning problem Pomerleau (1988). Basically, we ignore the fact that the samples are generated from a Markov decision process (MDP) and are not independently and identically distributed (i.i.d.). While this can be problematic for some domains and urged the use of interactive demonstrations Ross et al. (2011); Levine and Koltun (2013) or of ine reinforcement learning without on-policy interactions Ernst et al. (2005); Lange et al. (2012); Levine et al. (2020), for the locomotion gaits we considered in this paper and the choice of action and state space we did not nd a need for those algorithms. Hence for the learning problem, we rst generate datasets of ine for different gaits and solve a supervised learning problem to train control policies.

3.1. Policy structure

The ultimate goal in controlling a robot is to nd a policy that maps sensor measurements to joint torques. While our expert NMPC needs access to the states of the robot to compute actions (the dynamics model needs access to the states), this is not the case when we learn the control policy from it. To learn actions from sensor measurements, we consider two different approaches. In the rst approach, we learn two separate networks, one mapping measurements to states and the other mapping states to actions. The second approach directly maps measurements to actions without a need to encode measurements rst to intermediate states (see Fig. 1).

Figure 1: Two different structures to learn control policies from sensor measurements. The first approach is to learn an estimator (Estimator network) that maps measurements to states and a policy (Policy network) to map states to the actions. The second approach is to map directly measurements to actions (End-to-end network).

To compare these two approaches, we train an estimator network and use it together with a separate trained policy network, and one end-to-end network that directly maps measurements to actions (see Fig. 1). To make the comparison more meaningful, we consider the same number of layers and neurons for the separate (estimator and policy) and end-to-end networks. We also consider the same number of latent variables of the end-to-end network as the output of the estimator network (which is input to the policy network).

3.2. Action space

To minimize the run-time computation, we limit our analysis to the case where the output of the learned policy can be directly (or with a simple mapping) applied to the robot. In particular, we do not consider those approaches that learn some components of the optimal control problem and necessitate solving a smaller optimal control problem at run-time Carius et al. (2020); Viereck et al. (2022). We thus consider three different action spaces: the first and most intuitive one is the joint torques computed from (3) (called torque policy) Levine et al. (2016). The second action space is a structured one where the policy outputs q^{ff}, q^d, and v^d at each time (called structured policy). These values are then used to compute the torque applied to the joints at run-time using (3). The third action space we consider is the desired set point for a fixed gain PD controller (called PD policy)

$$\tau_t = k_p(q^d_{t+1} - q_t) - k_v v_t \tag{4}$$

Here, q^d_{t+1} sets a desired goal position for the joints to reach in the next time step. This value cannot be queried as a by-product of the NMPC problem. However, given the measured joint position and velocity and the applied torque at each time, the desired joint target is computed using

$$q^d_{t+1} = q_t + (\tau_t + k_v v_t)/k_p \tag{5}$$

It is important to mention that, as opposed to the structured policy, the PD policy does not try to track a desired planned trajectory. Instead, it sets a position goal at each cycle to be achieved by the controller, together with a damping term. This action space is also used widely in reinforcement learning for locomotion Peng and van de Panne (2017); Hwangbo et al. (2019); Bogdanovic et al. (2022) and has been shown to be beneficial for sim-to-real transfer Bogdanovic et al. (2020).

3.3. State space

The complete state space includes the base and joints position and velocity. However, as the locomotion skills are cyclic on unconstrained environment, we remove the base absolute horizontal position from the state space. Instead, we add the position of all the feet with respect to the base in horizontal directions to the state space. With this and without adding any information about contact to the state space, we let the network figure out those information on its own. We also found it crucial to add a phase variable (a scalar between zero and one) as input to the network. This phase variable, starts with zero at the start of the gait cycle and increases with time until the end of the cycle that reaches one. This phase variable encodes the cycling nature of the gaits.

3.4. Measurement space

We limit our measurement space to only the propriceptive sensors on the robot, without any camera, lidar or external motion capture system. This includes IMU measurements (linear acceleration and angular velocity) as well as the joint encoder measurements. Since this sensor space does not provide us with the position (mainly height) and orientation of the robot base, the states of the system are not directly observed from the measurements. We add realistic noise (based on the noise characteristics of the sensors) in the simulation when collecting data.

3.5. Data collection

To collect a diverse dataset for the learning problem, we need to initialize the robot from a distribution of initial conditions and then run the NMPC to collect data. We sample the initial condition around a nominal trajectory of the robot for the given gait. To make sure that the perturbed trajectory is consistent with the contact plan of the gait, we project the perturbations in the nullspace of the contact constraints as follows

$$\begin{bmatrix} q_c \\ v_c \end{bmatrix} = \left(I - A_c^{\dagger} A_c \right) \begin{bmatrix} q \\ v \end{bmatrix} \tag{6}$$

where q_c and v_c are the generalized coordinate and velocity perturbations that are consistent with the contact constraints. I stands for the identity matrix and the superscript $\dagger$ stands for pseudoinverse. The unconstrained perturbations are sampled from a Gaussian distribution, i. e., $q, v \sim \mathcal{N}(\mu, \sigma^2)$. The null-space projection matrix A_c is defined as

$$A_c = \begin{bmatrix} J_c & 0 \\ \dot{J}_c & J_c \end{bmatrix} \tag{7}$$

where J_c is the concatenation of Jacobians of the feet in contact. The constrained perturbation is added to the trajectory at the current time, i.e., $q + q_c$ and $v + v_c$. The NMPC is then rolled out from this initial condition in the simulation environment. If the rollout is successful and the robot does not fall down, we add the samples to the dataset. Otherwise, we simply ignore all the samples of the rollout. The whole procedure of training the policies are summarized in Algorithm 1.

4. Results

We present our results for three different gaits (trot, jump, and bound) in simulation for the Solo12 quadruped robot Grimminger et al. (2020). We aim to answer three main questions. Which one of

the action spaces in Section 3.2 work better for learning the controller? Can we learn a wide range of dynamic locomotion skills using only proprioceptive sensor measurements? How does end-to-end learning of locomotion skills compare with learning the estimator and the policy separately?

We collect data for learning using the procedure presented in Section 3.5 and use Pybullet for the simulation environment Coumans and Bai (2016). At each re-planning time of the NMPC, we perturb the robot around its nominal state and roll out the NMPC in simulation. This provided us with 264,450 samples for trot, 266,350 samples for jump, and 220,500 samples for bound. We note that this number of samples is at least one order of magnitude less than the number of samples acquired by a reinforcement algorithm to learn these gaits Bogdanovic et al. (2022). We used these samples for all the presented results in the paper.

For all the gaits, We use a three layer neural network with 256 neurons for the policy and a two layer network with 256 neurons for the estimator. We use Adam as the optimizer for training the network. All the layers are batch normalized and we use L_1 loss for training both the controller and the estimator. We considered a batch size of 64 samples and 500 epochs for all the training. All the simulations are done on a NVIDIA GeForce RTX 3050 Ti Laptop with Intel® Core™ i7-11800H @ 2.30GHz × 16 processor, and 31.1 GiB memory.

4.1. Choice of action space

We consider three different action spaces we presented in Section 3.2, i.e., torque, structured, and PD. To compare different action spaces, we test the three gaits giving the learned policy access to the ground truth states from the simulation. While the torque policy can complete the trot gait, for

Figure 2: Velocity tracking performance analysis for different action spaces. While both structured and PD policies are able to closely replicate the MPC behaviour, the tracking performance of the structured policy degrades drastically for jump and bound.

both the jump and bound gaits which are more dynamic, it fails to achieve the desired behaviour and the robot falls down a few moments after starting the gait (see the video). On the other hand, both the PD and structured policies are able to control the robot for jump and bound for all the gaits. However, as can be seen in Fig. 2 and the video, the structured policy has more undesired oscillations especially for bound. As, we will show in the next subsection, these oscillations cause instability when state estimation error is introduced.

To understand why the the PD policy outperforms the other two action spaces, we plot the test loss of the three policies in Fig. 3. The structure of the neural networks for all the policies are the same and they are trained with the same optimizer and loss terms. Furthermore, we have normalized the outputs of the networks to make the comparison more meaningful. Interestingly, we can see that even if the torque policy reaches lower test loss for jump, the policy fails to admit a stable gait. This rules out the hypothesis that torque has a more complex function to learn compared to the other action spaces such as PD. We believe the main reason that the PD policy outperforms the torque policy is that it is more robust to the function approximation error. In other words, an error in the set-point position of the PD controller causes less harm to the final performance of the system than the same error in joint torque. This also means that the structured policy is a valid action space and the main reason it functions slightly worst than PD action in practice is that it causes more approximation error per joint due to its larger number of policy outputs.

Figure 3: Test loss of training policies with the same structure of the neural networks.

4.2. Learning from sensor measurements

Classically, NMPC is used with a model-based filter such as extended Kalman filter (EKF) Bloesch et al. (2013); Camurri et al. (2020). However, without the use of camera or lidar, the base position and yaw angle of the robot base drift quickly. While for locomotion on flat terrain the horizontal position and yaw angle are not required for control, the base height is very important for control of dynamic motions such as jump and bound. In such cases, it becomes necessary to fuse the proprioceptive measurements with camera or lidar in standard state estimation settings Mastalli et al. (2020); Dárdin et al. (2022). In this subsection, we investigate on the question if it is possible to learn dynamic gaits such as jump and bound with only proprioceptive measurements, i.e., base IMU and joint encoders.

To learn locomotion skills from proprioceptive measurements, we learned a separate estimators for each gait that uses only IMU and encoder measurements and output the states required for the policy. Then we used the same estimator for the three considered action spaces. With the error of the torque policy fails to generate stable motions for all the gaits. Structured policy is able to produce stable trot and jump policy, but it fails to generate stable bound. However, PD policy is able to stabilize all the gaits in the presence of state estimation error (see the video). Not only are all the gaits stable, but they are also robust to different types of disturbances. Figure 4 illustrated the maximum amount of impulse in different directions that the robot could recover from.

In terms of computation time, the learned policy takes on average 0.5 ms to compute the actions from the estimated state, while the NMPC takes around 35 ms to recompute the whole body trajectory. While the effect of this computation time for NMPC on the real hardware can be mitigated by evaluating trajectories from the point in time that they are available Meduri et al. (2023), on an embedded system with very small computing budget this delay will be extremely larger which can degrade drastically the performance of the NMPC. However, with the learned policy not only the computation budget will not be a problem on the real hardware, but we also do not need to have computationally very intensive state estimators that fuse camera or lidar with the properioceptive sensors Mastalli et al. (2020); Dárdin et al. (2022) to provide drift-free estimate of the base states.

Figure 4: Push recovery performance of the learned policy together with estimator. The values are the maximum impulse (N.s) in each direction that the robot could recover from.

4.3. End-to-end learning

An alternative to learning the estimator and policy separately is to learn a single end-to-end policy that maps directly measurements to actions. To make the comparison more meaningful, we considered the structure of the network to have exactly the same as the estimator and policy in the previous section. Basically, we consider the size of the latent space of this end-to-end network the same as the size of the state space. The number of hidden layers and neurons before and after the latent layer are the same as the ones of the estimator and state policy, respectively. While with the end-to-end structure we were able to learn the trot motion, both the jump and bound motions failed (see the video). Interestingly, Ji et al. (2022) have made a similar observation for trot gaits learned through reinforcement learning. They showed that, for their problem, training an estimator that explicitly outputs some physical states of the system performs better than an end-to-end framework with inferred latent representation. While no concrete conclusion can be made based on these observations, it would be very interesting to systematically study this problem.

5. Conclusions

In this paper, we investigated the problem of learning agile locomotion skills in sensor space using NMPC as expert. We have shown that it is possible to learn a variety of agile gaits (trot, jump, and bound) using simple supervised learning without any need to interactive demonstrations from expert. Furthermore, we have shown that having the policy directly output joint torques fails to generate dynamic motions such as jump and bound. However, by adding a notion of joint position and velocity feedback in the policy, we can learn those gaits without any need to keep the structure of the optimal control problem. Furthermore, we have shown that it is possible to learn agile locomotion skills using only propriceptive measurements. Finally, we have shown that with the same neural network structure, learning the estimator and state policy separately outperform learning directly from sensor measurements.

In the future, we are planning to implement the gaits on the real robot. Furthermore, we are interested in combining the imitation learning problem from MPC with reinforcement such that we can improve the performance of the learned policy.

Acknowledgement

This work was supported by the Max Planck Institute for Intelligent Systems.

References

Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. arXiv preprint arXiv:2211.07638, 2022.

Michel Aractingi, Pierre-Alexandre Léziart, Thomas Flayols, Julien Perez, Tomi Silander, and Philippe Souères. Controlling the solo12 quadruped robot with deep reinforcement learning. 2022.

Michael Bloesch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian Gehring, C David Remy, and Roland Siegwart. State estimation for legged robots-consistent fusion of leg kinematics and imu. Robotics, 17:17–24, 2013.

Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Learning variable impedance control for contact sensitive tasks. IEEE Robotics and Automation Letters, 5(4):6129–6136, 2020.

Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization. Frontiers in Robotics and AI, 9, 2022.

Marco Camurri, Milad Ramezani, Simona Nobili, and Maurice Fallon. Pronto: A multi-sensor state estimator for legged robots in real-world scenarios. Frontiers in Robotics and AI, 7:68, 2020.

Jan Carius, Farbod Farshidian, and Marco Hutter. Mpc-net: A first principles guided policy search. IEEE Robotics and Automation Letters, 5(2):2897–2904, 2020.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. GitHub repository, 2016.

Victor Dhédin, Haolong Li, Shahram Khorshidi, Lukas Mack, Adithya Kumar Chinnakkonda Ravi, Avadesh Meduri, Paarth Shah, Felix Grimminger, Ludovic Righetti, Majid Khadiv, et al. Visual-inertial and leg odometry fusion for dynamic locomotion. arXiv preprint arXiv:2210.02127, 2022.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. Journal of Machine Learning Research, 6, 2005.

Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model predictive control. arXiv preprint arXiv:2208.08373, 2022.

Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Thomas Flayols, et al. An open torque-controlled modular robot architecture for legged locomotion research. IEEE Robotics and Automation Letters, 5(2):3650–3657, 2020.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. Science Robotics, 4(26):eaau5872, 2019.

Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. IEEE Robotics and Automation Letters, 7(2):4630–4637, 2022.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In Reinforcement learning, pages 45–73. Springer, 2012.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. Science robotics, 5(47):eabc5986, 2020.

Sergey Levine and Vladlen Koltun. Guided policy search. In International conference on machine learning, pages 1–9. PMLR, 2013.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research, 17(1):1334–1373, 2016.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Of ine reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 2811–2817. IEEE, 2021.

Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An ef cient and versatile framework for multi-contact optimal control. 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 2536–2542. IEEE, 2020.

Carlos Mastalli, Wolfgang Merkt, Guiyang Xin, Jaehyun Shim, Michael Mistry, Ioannis Havoutis, and Sethu Vijayakumar. Agile maneuvers in legged robots: a predictive control approach. arXiv preprint arXiv:2203.07554, 2022.

Avadesh Meduri, Paarth Shah, Julian Viereck, Majid Khadiv, Ioannis Havoutis, and Ludovic Righetti. Biconmp: A nonlinear model predictive control framework for whole body motion planning. IEEE Transactions on Robotics, 2023.

Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. Science Robotics, 7(62):eabk2822, 2022.

Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 1–13, 2017.