

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Imitating and Finetuning Model Predictive Control for Robust and Symmetric Quadrupedal Locomotion

Donghoon Youm¹, Hyunyoung Jung², Hyeongjun Kim¹, Jemin Hwangbo¹, Hae-Won Park¹, and Sehoon Ha²

Abstract—Control of legged robots is a challenging problem that has been investigated by different approaches, such as model-based control and learning algorithms. This work proposes a novel Imitating and Finetuning Model Predictive Control (IFM) framework to take the strengths of both approaches. Our framework first develops a conventional model predictive controller (MPC) using Differential Dynamic Programming and Raibert heuristic, which serves as an expert policy. Then we train a clone of the MPC using imitation learning to make the controller learnable. Finally, we leverage deep reinforcement learning with limited exploration for further finetuning the policy on more challenging terrains. By conducting comprehensive simulation and hardware experiments, we demonstrate that the proposed IFM framework can significantly improve the performance of the given MPC controller on rough, slippery, and conveyor terrains that require careful coordination of footsteps. We also showcase that IFM can efficiently produce more symmetric, periodic, and energy-efficient gaits compared to Vanilla RL with a minimal burden of reward shaping.

Index Terms—Legged robots, reinforcement learning, imitation learning.

I. INTRODUCTION

DESIGNING a robust locomotion controller has been a challenging problem for roboticists due to the under-actuated, discrete, and high-dimensional dynamics of legged robots. Roboticists have tackled legged locomotion with various approaches, from model-based control methods to learning-based techniques. Over the decades, traditional model-based control has demonstrated impressive agile and robust gaits on quadrupedal robots [1]–[7]. However, it often requires extensive effort to develop proper mathematical models, which becomes more challenging for complex and unconventional situations. On the other hand, learning-based approaches, such as deep reinforcement learning (deep RL),

Manuscript received April 24, 2023; Revised July 18, 2023; Accepted September 12, 2023.

This letter was recommended for publication by Editor Clement Gosselin upon evaluation of the Associate Editor and Reviewers' comments

This work was supported by Korea Evaluation Institute of Industrial Technology (KEIT) funded by the Korea Government (MOTIE) under Grant No.20018216, Development of mobile intelligence SW for autonomous navigation of legged robots in dynamic and atypical environments for real application. The Mini Cheetah robot was provided by MIT Biomimetic Robotics Lab and NAVER LABS. Corporation. (Donghoon Youm and Hyunyoung Jung are co-first authors.) (Corresponding author: Hae-Won Park.)

¹Donghoon Youm, Hyeongjun Kim, Jemin Hwangbo, and Hae-Won Park are with the Korea Advanced Institute of Science and Technology, Yuseong-gu, Daejeon, 34141, Republic of Korea ydh0725, kaist0914, jhwangbo, haewonpark@kaist.ac.kr

²Hyunyoung Jung and Sehoon Ha are with the Georgia Institute of Technology, Atlanta, GA, 30308, USA hjung331, sehoonha@gatech.edu

Digital Object Identifier (DOI): see top of this page.



Fig. 1. Our Imitating and Finetuning Model Predictive Control (IFM) framework can learn a robust, symmetric, periodic, and efficient policy that enables the Mini-cheetah robot to overcome a 7.5cm height obstacle.

have the potential to automatically solve unintuitive locomotion problems on challenging terrains by leveraging a massive amount of data [8]–[15]. However, deep RL may need to be followed by repetitive reward shaping to obtain high-quality motions. Otherwise, the learned policy may learn jerky and asymmetric gaits that cannot be transferred to real robots.

In recent years, researchers have investigated frameworks to combine model-based control and deep RL, taking the strengths of both approaches. One popular method is a hierarchical control architecture, where a learnable policy generates high-level commands, such as the target speed or footstep locations, for a low-level model predictive controller (MPC) [16]–[18]. Also, there have been attempts to replace one control module with a neural network [19]. However, these approaches still require executing MPC at high frequency and can yield additional issues, such as significantly increased simulation costs and inference speed.

We seek inspiration from the recent success of pre-training in various machine learning areas, including computer vision and natural language processing. Particularly, we pre-train a policy by mimicking the expert's behaviors. One possible solution is to replicate the generated actions of the MPC using supervised learning by carefully designing a loss function [20], [21]. On the other hand, motion imitation allows users to learn a policy that can track the given reference motion using deep RL, which shows natural motions both in simulation [22] and on hardware [23], [24]. Bogdanovic et al. [25] further proposed to improve the performance via task-based finetuning. However, the performance of motion imitation is still bounded by the given single reference trajectory, which makes it difficult to obtain a general policy for various command tracking tasks.

This work proposes an Imitating and Finetuning Model Predictive Control (IFM) framework, which is designed to take advantage of both model-based control and learning-based algorithms. We first design an MPC using Differential Dynamic Programming (DDP) and Raibert heuristics. We then train a neural network policy to imitate the expert controller using Dataset Aggregation (Dagger) [26]. Finally, we finetune the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

policy to overcome various challenging terrains using the on-policy deep RL algorithm, Proximal Policy Optimization [27]. We further employ many sub-components, such as action space conversion, constrained exploration, terrain curriculum, and domain randomization, for seamless integration and better performance.

We demonstrate the benefit of the proposed IFM framework by conducting both simulation and hardware experiments on the Mini Cheetah robot [6]. We first show that our approach can train a new policy with improved robustness compared to the expert policy (MPC) on rough terrains, particularly when it is required to take unconventional footfall patterns, such as moving conveyor belts. It also has significantly reduced computation time compared to that of MPC. In addition, IFM finds symmetric, periodic, and energy-efficient gaits compared to Vanilla RL with a minimal burden of reward shaping, which leads to better hardware performance. It also converges much faster than Vanilla RL, which makes it more suitable for challenging locomotion problems. In our experiments, the IFM policies outperform the other methods on hardware regarding the success rate and tracking error. We summarize the contributions as follows.

- We propose a novel Imitating and Finetuning Model Predictive Control (IFM) framework to take the benefits of MPC and RL.
- We demonstrate that IFM can significantly improve the performance of MPC and reduce the computation time.
- We demonstrate that IFM can improve the performance of Vanilla RL in terms of overall tracking errors and motion styles, with less burden of reward engineering.

II. RELATED WORK

A. Model-based approaches

A model-based approach leverages a mathematical model to illustrate the dynamics of the given system. In the case of quadrupedal robot control, the robot is typically modeled as a single rigid body with small leg mass assumption [1]–[5] to make optimization feasible. Particularly, MPC [1]–[3], [5]–[7] has been widely used for several decades. It predicts the robot's future states and generates optimal control inputs based on a predefined cost function. MPC has the advantages of being explainable, robust, and capable of producing natural locomotion in legged robots. However, MPC is often formulated based on the predefined model, which requires experts' prior knowledge to develop specific models to overcome rough and dynamic terrains. Some recent research has focused on using terrain information and model-based optimization for foothold selection [28]. Still, many researchers adopt Raibert's heuristics [29], which cannot guarantee the robot's performance in cases where the foot slips significantly during walking or gets stuck on obstacles. In this work, we utilize the aforementioned advantages of MPC while compensating for its weakness with a learning-based approach.

B. Learning-based approaches

The recent progress in learning-based control methods has demonstrated the effectiveness of legged robot control [8]–[10], which has been proven to be effective in many scenarios.

The performance of these learning-based approaches can be further boosted by introducing additional techniques, such as privileged learning [30], online adaptation [11], [12], system identification [8], belief propagation [13], curriculum learning [14], [15], and many more. However, learning-based approaches often require careful reward shaping to obtain high-quality, symmetric, and periodic motions, which sometimes need up to ten reward terms, ranging from task terms to various regularizations. The proposed framework, IFM, intends to supplement these works by pre-training a control policy with an MPC expert. Liang et al. [31] employed a similar approach. However, their use of human demonstrations precludes the application of DAGger [26] and there are no results from the real-world test.

C. Combined approaches

There exist a few works that aim to combine model-based and learning-based control approaches. One approach is hierarchically integrating the MPC and RL algorithms, where high-level decision-making is done by the learnable policy, and low-level motion control is done by the model-based control [16]–[19]. Villarreal et al. [32] adopted this approach for combining a vision-based foothold selector and MPC. The other approach is to train the residual network using a learnable policy together with a model-based controller [33]. While demonstrating significant performance improvement, these formulations require expensive optimization for running MPC. Carius et al. [20] and Reske et al. [21] utilize the Hamiltonian function as a loss function to imitate the existing MPC. Their methods can even slightly outperform the original controller by learning from multiple experts, but still, there is no explicit notion of finetuning.

The approach of imitating MPC has been explored in other applications, such as autonomous driving [34] and drone [35], [36]. Particularly, Nagami et al. [37] adopted this imitation learning as a pre-training to solve problems with sparse rewards that are hard to solve by Vanilla RL. Inspired by these works, our work aims to significantly improve the performance of MPC-based locomotion controllers while improving the quality of the RL-generated motions without reward engineering.

III. IMITATING AND FINETUNING MPC

We present a novel Imitating and Finetuning Model Predictive Control (IFM) framework for combining the strengths of model-based and learning-based control. We first develop an expert MPC using DDP and Raibert heuristics. Then we pre-train a policy by imitating the given expert MPC and finetune the cloned policy on various challenging tasks. The framework is summarized in Fig. 2.

From the perspective of MPC, our framework provides an opportunity to improve the controller on challenging terrains that are hard to describe mathematically. From the perspective of deep RL, our framework finds periodic, symmetric, and physically plausible gaits without complex reward shaping in a sample-efficient manner.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

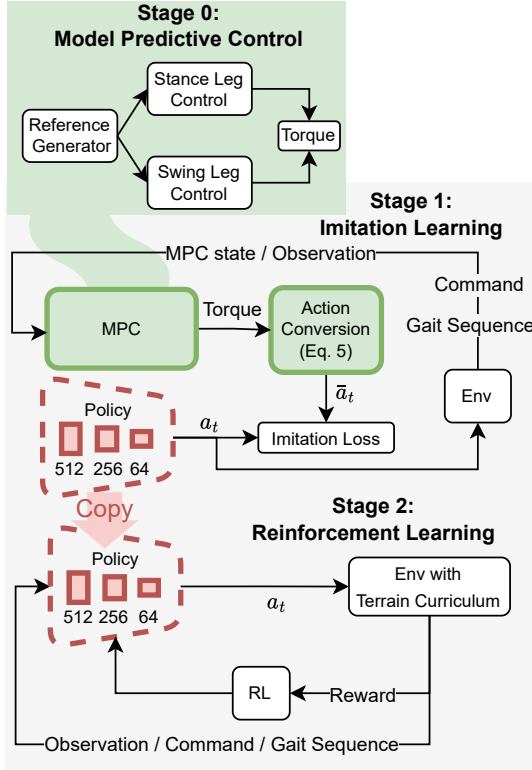


Fig. 2. Overview of the proposed Imitating and Finetuning Model Predictive Control (IFM). We first develop the expert controller π^E using MPC. Then we pre-train a neural network policy π^I by cloning the expert's behaviors on the flat terrain. Finally, we finetune the policy on more challenging terrains using RL.

A. Stage 0: Model Predictive Control

Our framework, IFM, starts by developing an expert controller π^E . In theory, our framework is agnostic to the design process of the given expert policy, which can be obtained from various approaches, including heuristic methods, model-based algorithms, or other learning-based methods. In our implementation, we develop an expert policy, which uses DDP as in Tassa et al. [38] with friction pyramid constraints for stance legs and Raibert heuristics [29] coupled with task space impedance control for swing legs.

1) *Controller Design*: We define the state and control input of the convex MPC as follows:

$$\begin{aligned} \mathbf{x}(t) &\triangleq [\Theta_t, \mathbf{p}_t, \boldsymbol{\omega}_t, \mathbf{v}_t] \\ \mathbf{u}(t) &\triangleq [\mathbf{f}_1^T, \mathbf{f}_2^T, \mathbf{f}_3^T, \mathbf{f}_4^T], \end{aligned} \quad (1)$$

where Θ_t , \mathbf{p}_t , $\boldsymbol{\omega}_t$, and \mathbf{v}_t , are the Z-Y-X Euler angles, position, angular velocity, and linear velocity of the base at time t , respectively. Each \mathbf{f}_i represents a ground reaction force of the i^{th} foot and is multiplied by the transpose of foot Jacobian to apply joint torque.

Then we define our cost function of DDP as follows:

$$\min_{\mathbf{x}, \mathbf{u}} \sum_{i=0}^{T-1} \|\mathbf{x}_{i+1} - \bar{\mathbf{x}}_{i+1}\|_{\mathbf{Q}} + \|\mathbf{u}_i\|_{\mathbf{R}}, \quad (2)$$

where $\bar{\mathbf{x}}$ is the desired state and \mathbf{Q} and \mathbf{R} are weight matrices. The reference generator outputs a reference state trajectory by incorporating the current state and velocity command in the same manner as Di et al. [3]. The velocity command comprises

the desired linear velocities in the x and y directions and a yaw velocity. In order to match the control frequency of the policy network, which will be described later, 100Hz, the control dt of the MPC was set to 0.01s, and the horizon length T was set to 26.

Note that the formulated MPC cannot be executed in real-time on hardware because it takes 12.382 ms for optimization. However, we only run this MPC controller in simulation to generate pre-training data.

In addition, we made two changes to the original framework of Tassa et al. [38] based on the work of Di et al. [3]. First, we adopt friction pyramid constraints instead of box constraints. We also implement the swing leg controller to compute joint torque for leg i using task space impedance control as:

$$\boldsymbol{\tau}_i = \mathbf{J}_i^T [K_p^t (\mathbf{p}_i^f - \bar{\mathbf{p}}_i^f) + K_d^t (\mathbf{v}_i^f - \bar{\mathbf{v}}_i^f)] \quad (3)$$

where \mathbf{J}_i is the i -th foot Jacobian, $\mathbf{p}_i^f, \mathbf{v}_i^f$ are the position and velocity of the i -th swing leg foot. We used task space PD gain $K_p^t = 150$ and $K_d^t = 5$. The upper bar denotes the reference foot state. The reference generator uses the Raibert heuristic [29] for the foothold and a cubic Bézier curve connecting the current foot to this foothold to create the reference foot trajectory. The gait sequence is a predefined swing time of 0.13 s and a stance time of 0.13 s without flying phases.

2) *Action Conversion*: Many MPC frameworks generate torques by converting the given desired contact forces using Jacobian transpose, including our expert controller. Conversely, it is more common for RL to select the target joint angles as actions because torque-based policies often exhibit significant vibration on hardware. Therefore, we convert the torque command of the MPC into the target joint angles to make it compatible with imitation and RL policies.

For this purpose, we assume the joint space proportional-derivative (PD) controller that generates torques as:

$$\boldsymbol{\tau} = K_p^j (\mathbf{q} - \bar{\mathbf{q}}) - K_d^j \dot{\mathbf{q}}, \quad (4)$$

where \mathbf{q} , $\dot{\mathbf{q}}$, and $\bar{\mathbf{q}}$ are the current joint angles, the current joint velocities, and the target joint angles, respectively. We used joint space PD gains $K_p^j = 17$ and $K_d^j = 0.4$ in our whole experiments. We rearrange this equation to compute the target angle $\bar{\mathbf{q}}$ from the given torque $\boldsymbol{\tau}$:

$$\bar{\mathbf{q}} = \frac{1}{K_p^j} (K_p^j \mathbf{q} - K_d^j \dot{\mathbf{q}} - \boldsymbol{\tau}). \quad (5)$$

B. Stage 1: Imitation Learning

1) *Imitation learning with Data Aggregation*: Despite its robustness, the expert π^E is intrinsically a fixed controller that prevents it from being trained for further improvement. In addition, it is computationally more expensive than a single feed-forward neural network because of complex numerical optimization. Therefore, we convert π^E to the neural network policy π^I whose weights can be optimized by minimizing the imitation loss:

$$L_{imi} = \sum_{\mathbf{x} \in D} \|\pi^E(\mathbf{x}) - \pi^I(\mathbf{x})\|^2, \quad (6)$$

where D is the collected dataset. However, this naive behavior cloning approach often fails when the policy encounters previously unseen data during testing, which is known as the distribution mismatch problem. We can mitigate this issue by adopting the Dataset Aggregation (Dagger) approach [26], where we roll out ten trajectories from the policy π^I , label the correct action with the expert π^E and add them to the dataset D for every iteration. This approach results in a more stable policy by collecting in-distribution data and using it for training. Note that this imitation stage is not computationally expensive compared to the RL finetuning: we only need to sample less than 400K data for Dagger training.

2) *Policy Architecture*: The policy π^I is implemented as a Multi-Layer Perceptron (MLP) with three hidden layers consisting of 512, 256, and 64 neurons and leaky ReLU as the activation function. The input to the MLP consists of body orientation, body angular velocity, current joint angle and velocity, previous action, joint angle, and velocity history, predefined contact sequence and phase, and user command. Although the predefined contact sequence is provided as part of inputs, the policy π^I does not need to follow the fixed contact schedule explicitly. This implies that our policy is not strictly limited to fixed footfall patterns defined in Sec. III-A.

The joint history is vital information for blind locomotion on rough terrain, which enables the robot to infer terrain properties. Although we only use flat terrain in the imitation learning stage, we include joint history as an input for later RL finetuning tasks. We also do not include the body linear velocity as an observation since it can be implicitly inferred from previous outputs, joint history, and current joint information. In our experience, explicitly providing the body linear velocity as an input can potentially degrade the performance of the policy on hardware because the robot needs to rely on a noisy state estimator.

The policy's output is defined as the joint angle difference from its initial position, scaled by a factor σ , such that $\bar{\mathbf{q}} = \mathbf{q}_{init} + \sigma \mathbf{a}$, where \mathbf{q}_{init} is the predefined initial pose, \mathbf{a} is the policy's output, and σ is the scaling factor.

3) *Training Details*: We optimize the policy parameters with respect to the given loss function (Eq. 6) using adaptive momentum estimation (Adam) [39]. We terminate the sampling if the non-feet body part touches the ground or the robot flips over 70° to avoid including samples where the MPC has failed to converge. The batch size of data increases linearly by aggregating 4000 data at every iteration (10 environments \times 400 transitions), and each batch is trained for 10 epochs.

C. Stage 2: Reinforcement Learning

So far, we have obtained the initial expert controller π^E and the imitation policy π^I . However, neither is optimal due to the model assumptions of the original expert. In this section, we aim to further boost the performance of the policy by finetuning it on challenging terrains using RL. Compared to the MPC, the RL policy π^R can flexibly adapt the robot's gait to various terrains without explicit modeling. Compared

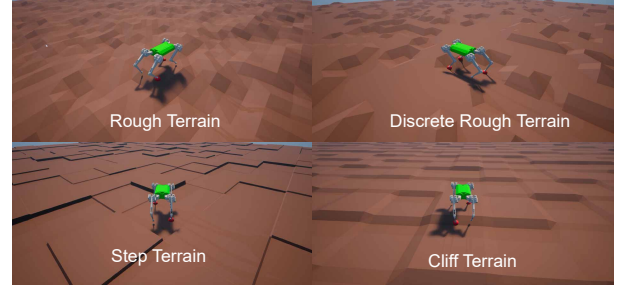


Fig. 3. Illustration of the challenging terrains for RL finetuning.

to IL, RL explicitly optimizes the performance of the policy on downstream tasks.

1) *Curriculum over Downstream Tasks*: We select several challenging terrains as downstream tasks, including rough terrain, discrete rough terrain, step terrain, and cliff terrain, as shown in Fig. 3. These terrains are generated by varying the *terrain factor*, which is the scaled maximum height of the terrain. For example, we used Perlin noise with 0.5 fractal octaves, 1 to 5 fractal lacunarity and 0.45 fractal gain while varying the z-scale using this *terrain factor* to generate rough terrain. We also introduce curriculum learning to avoid the loss of the style on harsh terrains. Specifically, we start with a terrain factor of 0.01 and increase it exponentially by a factor of 0.96 every 5 iterations. We also sample the flat terrain with about a 5% probability to prevent the robot from losing its pre-trained motion.

2) *Reward Formulation*: One challenge in applying deep RL to high-dimensional control problems is that the agent often produces unintended and noisy motions that cannot be transferred to a real robot, depending on the formulation. Therefore, researchers often have approached this problem with manual reward engineering, which uses up to ten reward terms to obtain the desired style of locomotion [14], [15]. On the other hand, we unburden this process by leveraging imitation learning which reduces the search space of policy to an action space that is feasible for the real robot.

We finetune the policy using deep RL in two different settings. In a simple reward (SR) setting, the following minimally constructed rewards terms are used: (1) tracking the given velocity command $[\bar{v}_x, \bar{v}_y, \bar{\omega}_z]$, (2) minimizing the unnecessary movements in the other directions (v_z, ω_x, ω_y), and (3) regularizing the torque usage. In a contact reward (CR) setting, we introduce an additional contact biasing reward on top of the existing terms to encourage the policy to follow a predefined contact sequence.

For each episode, we also randomly sample the initial state, and the target command from $U(-1, 1)$.

3) *Domain Randomization*: The model trained on simulation might not work if we deploy it directly to the real world. Therefore, We perform domain randomization for successful zero-shot sim-to-real transfer. We adopt a similar randomization technique as Ji et al. [9]. To be more specific, we randomize the observation noise, motor friction, PD controller gains, ground friction coefficient, and the delay in motor command transmission time. For the range of ground friction coefficient, we uniformly sample from $[0.3, 1.0]$, which is a wider range than Ji et al. Since there is about 10 ms latency

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

between the actual controller and the robot, we model this latency by setting the PD target after uniformly sampled time between $[0, 10]$ ms.

4) *Training Details*: We use Proximal Policy Optimization (PPO) [27] coupled with asymmetric actor-critic algorithm [40] for training. We use the same actor network as in Sec. III-B but modify the policy to have a Gaussian distribution to effectively explore possible solutions during training. To be more specific, the MLP structure outputs the mean of the policy, and the learnable parameters act as the standard deviation of the Gaussian distribution. For the critic network, we use the same MLP structure as in the actor network, which has three hidden layers consisting of 512, 256, and 64 neurons and leaky ReLU as the activation function. In addition, unlike actor observation, the noise was not included in the value observation, but the linear velocity and body height were included, and joint history was excluded.

5) *Constrained Exploration*: In IFM, it is important for the policy to maintain its pre-trained motion during finetuning. Therefore, we set the initial standard deviation, learning rate, and PPO clip parameters to small values to allow the policy to explore the environment while retaining its meaningful pre-trained motion. Specifically, we set the initial standard deviation to 1.0 (or 2.0 in CR), the policy network learning rate to 10^{-5} , and the PPO clip parameter to 0.05 (or 0.1 in CR). We use a learning rate of 10^{-3} for the critic network.

IV. EXPERIMENTAL RESULTS

We design experiments to investigate the following research questions: (1) can IFM improve the robustness of model-based controller and reduce the computation time? (2) Can IFM improve the performance of vanilla RL in terms of tracking errors and motion quality? (3) is IFM more sample efficient than Vanilla RL?

A. Experimental Details

1) *Baselines*: We select three controllers for our baselines which are given as follows:

- **MPC**: We use two different MPCs for simulation and hardware experiments. In the simulation, MPC is the model-based control policy described in Sec. III-A. In hardware experiments, the controller is convex MPC with the Whole Body Impulse Control (WBIC) [7] that is tuned for the hardware.
- **Dagger**: Dagger is the policy trained up to the imitation stage in Sec. III-B.
- **Vanilla RL(SR)**: The Vanilla RL(SR) baseline is trained using the same reward function as the simple reward setting in Sec. III-C but without any imitation pre-training. We adopt the IFM’s training strategy, including a terrain curriculum. However, we set the standard deviation to 3.0 and the PPO clip parameter to 0.2 and do not limit the exploration.
- **Vanilla RL(CR)**: The Vanilla RL(CR) baseline is trained similarly but with an additional contact biasing reward as in III-C2.
- **Concurrent RL**: The Concurrent RL baseline is the exact work from Ji *et al.* [9].

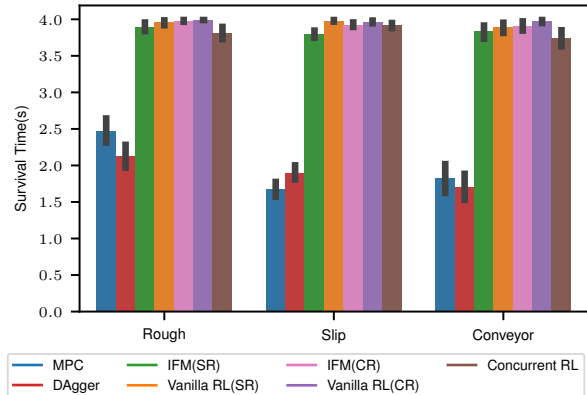


Fig. 4. Comparison of the robustness in simulated environments. IFM (ours) and Vanilla RL perform better than MPC and DAGger.

2) *Simulation*: We use RaiSim [41] as our physics simulator. The simulation time step is given as 0.001s, and the control frequency is set to 100 Hz. We conduct the experiments on AMD Ryzen Threadripper 3970X 32-Core Processor CPU and GeForce RTX 3080 Ti GPU.

3) *Hardware*: We select the Mini Cheetah robot [6] as the hardware platform, which is lightweight and capable of demonstrating dynamic locomotion.

B. Simulation

This section discusses the improved robustness and tracking performance of our method, particularly compared to the model-based expert controller.

1) *Robustness*: We evaluated the robustness of all the control policies on three different terrains in the simulation: rough, slippery, and conveyor belts. Here, conveyor terrain is constructed such that several blocks relatively move in the x -direction. Please note that rough and slippery terrains are part of training environments, while conveyor belt terrain is unseen during training. We select the survival time until fall as a performance criterion.

The results are illustrated in Fig. 4. Overall, IFM significantly improves the robustness of MPC and DAGger. This trend is not only for in-distribution tasks (rough and slip terrains) seen during training but also for a novel conveyor terrain. These results imply that learning-based policies can effectively adapt the robot’s gait, particularly contact sequences, to challenging terrains, while MPC’s performance is limited due to its predefined model, such as a fixed contact sequence and reference swing leg trajectory.

On the other hand, IFM(SR)’s performance is comparable to that of Vanilla RL(SR) and even slightly worse for some out-of-distribution tasks. This is because Vanilla RL(SR) found an asymmetric, energy-inefficient gait that is similar to pronking. This discrepancy decreases as contact biasing reward is introduced and more extensive exploration is employed. We will further compare the Vanilla RL and IFM policies in Sec. IV-D.

2) *Tracking Performance*: We also compare the policies by measuring their capabilities to follow the velocity command on the flat terrain. The command vector consists of a 3-dimensional vector, $[v_x, v_y, \omega_z]$, which is sampled uniformly

TABLE I
SIMULATION AND HARDWARE PERFORMANCE ON THE FLAT TERRAIN

Method	Simulation		Hardware Experiments			
	Track err (↓) (linear)	Track err (↓) (angular)	Track err (↓) (linear)	Track err (↓) (angular)	CoT (↓) (1m/s)	PPI (↓) (1m/s)
MPC/WBIC	0.1520 (±0.0169)	0.5335 (±0.0519)	0.2334 (±0.0062)	0.7901 (±0.0108)	0.4374 (±0.0113)	1.715 (±0.3547)
DAGger	0.1838 (±0.0217)	0.5214 (±0.0449)	0.2694 (±0.0065)	1.153 (±0.0075)	0.4489 (±0.0020)	1.880 (±0.3067)
Vanilla RL(SR)	0.2017 (±0.0125)	0.2757 (±0.0114)	0.3814 (±0.0711)	1.339 (±0.1195)	1.287 (±0.0981)	10.13 (±2.007)
IFM(SR)	0.1335 (±0.0163)	0.2509 (±0.0153)	0.1620 (±0.0103)	0.7320 (±0.0100)	0.4237 (±0.0144)	1.798 (±0.3975)
Vanilla RL(CR)	0.1766 (±0.0175)	0.2926 (±0.0123)	0.2653 (±0.0057)	0.9680 (±0.0146)	0.9154 (±0.0317)	5.001 (±0.6006)
IFM(CR)	0.1879 (±0.0175)	0.2624 (±0.0102)	0.2826 (±0.0102)	0.6978 (±0.0321)	0.3250 (±0.0049)	2.936 (±0.1979)
Concurrent RL	0.1793 (±0.0153)	0.2185 (±0.0089)	0.1722 (±0.0155)	0.4099 (±0.0040)	0.2206 (±0.0065)	2.385 (±0.1602)

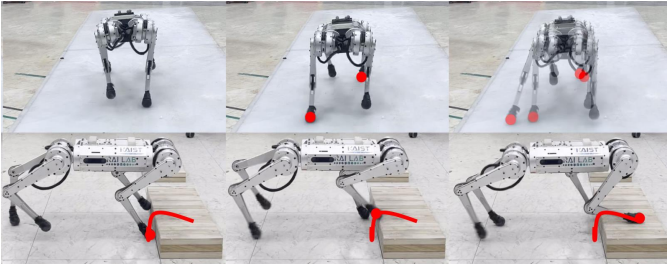


Fig. 5. IFM policy can traverse slippery terrain with the friction of 0.22 (Top) and also overcome a 7.5cm step using foot-trapping reflexes (Bottom).

from $v_x \in [-0.3, 1]$, $v_y \in [-0.5, 0.5]$, and $\omega_x \in [-0.6, 0.6]$. We changed this command four times in one trajectory.

The results are summarized in Table I where we highlighted the top 3 performances with gray backgrounds. IFM(SR) and Concurrent RL show the best performance in linear and angular velocity tracking, respectively. MPC and DAGger failed to follow the command and fell off 12% of the tests, which never happened for both IFM and Vanilla RL. They also showed worse linear and angular velocity tracking performance than IFM(SR). Interestingly, Vanilla RL(SR) demonstrated poor linear tracking performance, even worse than MPC or DAGger. This is mainly because it has excessive z -direction movements produced by its pronking motion. Despite being trained with the same reward formulation as Vanilla RL(SR), IFM(SR) did not show that problem because of pre-training. Vanilla RL(CR) showed better tracking than Vanilla RL(SR) by following specific contact sequences, but still worse than IFM(SR).

3) *Computation Time*: IFM reduces MPC’s computation time from 0.0154s to 0.0010s, thanks to its simple network architecture. This showcases the benefits of IFM in scalable learning and real-time control.

C. Hardware Experiments

We further conducted hardware experiments.

1) *Robustness*: First, we measured the success rate of stepping over the three individual steps whose heights were 2.5cm, 5cm, and 7.5cm, respectively (Table II). The success rate is measured over 10 trials. WBIC could not step over the 7.5 cm height, which is beyond its foot clearance. The DAGger policy showed even worse performance and failed to overcome the 5 cm height. IFM(CR) showed the best performance so far at a 90 % success rate (Fig. 1 and Fig. 5).

TABLE II
HARDWARE EXPERIMENTAL RESULTS ON CHALLENGING TERRAINS

Method	Step terrain			Slippery terrain(1m/s)	
	Success rate(%) (†)			Tracking error (↓)	Pass time(s) (↓)
	2.5cm	5cm	7.5cm		
WBIC	100	100	0	Fail	Fail
DAGger	100	0	0	Fail	Fail
Vanilla RL(SR)	100	100	60	0.6550	2.63
IFM(SR)	100	100	60	0.1893	2.00
Vanilla RL(CR)	100	100	80	Fail	Fail
IFM(CR)	100	100	90	0.2396	2.550
Concurrent RL	100	0	0	0.2081	2.335

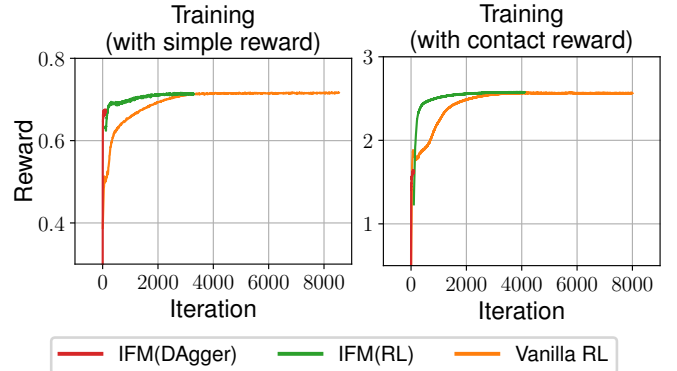


Fig. 6. Learning curves of Vanilla RL and IFM, which illustrate faster convergence of IFM over Vanilla RL.

Further, we gave the robot a 1m/s command on the slippery terrain. We spread out the boric acid powder on a 6m whiteboard to create a terrain with a friction coefficient of 0.22 and measured the tracking error and the time taken to cross the central 2m area of the whiteboard. In the case of MPC and DAGger, Raibert’s heuristic foothold could not maintain stability when the foot slipped significantly, resulting in a fall. Both IFM(SR) and Vanilla RL(SR) can pass the slippery surface, but IFM(SR) showed significant improvements over the tracking error and pass-time. Similarly, IFM(CR) succeeded to pass this terrain in a reasonable time while Vanilla RL(CR) failed.

2) *Tracking Performance*: We measured the tracking performance on hardware, where each controller is tasked to follow the 6-second command (Fig. 7) for five trials. The averaged errors are also summarized in Table I. IFM(SR) and Concurrent RL are showing the lowest tracking errors in linear and angular velocity, respectively. This is notable in that WBIC is specifically designed for the hardware and Concurrent RL has complex rewards (nine terms) and an additional state estimation for better sim to real transferability. On the other hand, both Vanilla RL(SR) and Vanilla RL(CR) show worse tracking performance.

D. Comparison between IFM and Vanilla RL

This section further highlights the strengths of IFM over Vanilla RL: (1) learning efficiency, (2) tracking performance, and (3) motion quality.

1) *Learning Efficiency*: We compare Vanilla RL’s and IFM’s learning efficiency by plotting the reward in Fig. 6. The learning curves represent the mean of three trials, with

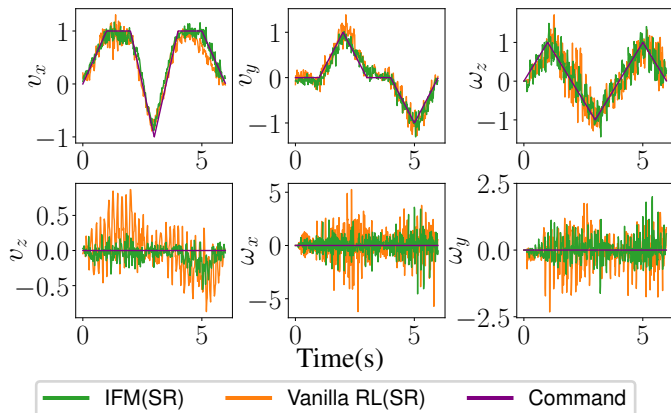


Fig. 7. Hardware command tracking performance. Overall, IFM(SR) shows better tracking than Vanilla RL(SR), particularly in v_z , ω_x , and ω_y .

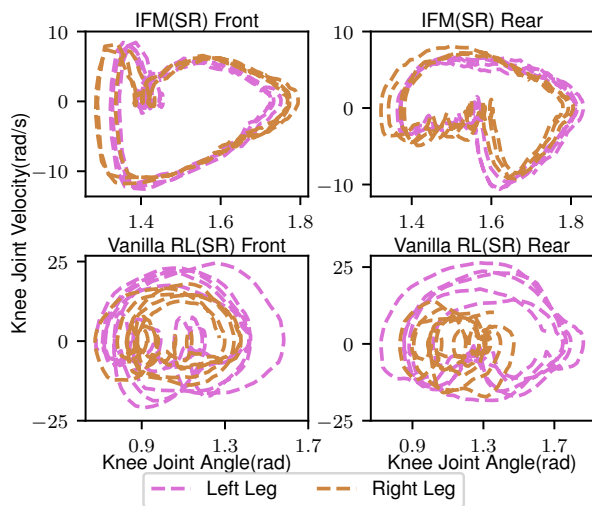


Fig. 8. Hardware knee joint phase-portrait. IFM(SR)’s left leg and right leg have mostly overlapping phase-portrait, meaning that it has a periodic and symmetric motion. On the other hand, Vanilla RL(SR) shows highly asymmetric motions with rarely overlapping phase-portrait.

the exclusion of confidence intervals due to their narrow range. Note that we divide the IFM learning curve into DAgger and RL stages. The plot clearly shows that IFM (~ 2000 iters) converges faster than Vanilla RL (~ 4000 iters).

2) *Tracking Performance*: In our experiments, Vanilla RL tends to show great robustness, as shown in Fig. 4 and Tab. II. However, it is not well suited for velocity tracking and shows consistently lower tracking performances in both simulation and real-world environments. We plot the illustrative scenario for simple reward policies in Fig. 7, which shows worse tracking of Vanilla RL(SR) in v_z , ω_x , and ω_y directions due to its unnecessary movements. It will be possible to correct this behavior by comprehensive manual reward shaping, such as Vanilla RL(CR): however, IFM offers a more structured approach to obtain better gaits.

3) *Motion Quality*: We conduct a more in-depth analysis of the motion quality of IFM and Vanilla RL by measuring the Cost of Transportation (CoT) and Phase-Portrait Index (PPI) [42] on hardware. As shown in Table I, IFM methods’ CoT and PPI are comparable to those of MPC/WBIC. This is not surprising because IFM can be viewed as the finetuned

TABLE III
QUALITATIVE COMPARISON OF MPC+RL METHODS

	IFM (ours)	Contact Adaptive [19]	GLiDE [16]	MPC-net [20]
MPC+RL approach	Imitation	Hierarchical	Hierarchical	Imitation
Network Action Space	Target joint angle	Contact sequences	Body acceleration	Joint velocities and foot contact forces
Computation Time	Low ($<1\text{ms}$)	High	High	Low ($<1\text{ms}$)
Contact Adaptation	Yes	Yes	No	Yes
Improvement of MPC	Yes	Yes	Yes	No
Demo terrain ¹	F,R,Sl,C,St	F,Sl,C	F	F

version of a model-based expert. On the other hand, Vanilla RL policies exhibit very poor CoT and PPI, significantly larger than the others. The phase portrait in Fig. 8 shows clear differences in PPI in the simple reward setting. In addition, the left and right phase portraits of IFM(SR) are nearly overlapped, which indicates near-perfect symmetry.

Our experimental results do not indicate that vanilla RL cannot discover symmetric and efficient gaits. However, it often requires task-specific reward shaping, as a successful reward function in one scenario can exhibit hugely different motions in other situations. IFM provides a more explicit method for regulating the style of motion and significantly alleviates the challenge of reward engineering.

E. Qualitative comparisons of MPC+RL methods

To provide additional contextualization of our work, we present qualitative comparison in Table III. In comparison to other studies, our proposed method exhibits faster computation time and more robust foot adaptation. This advantage stems from the absence of online optimization and the lack of predefined swing leg controls in IFM. Furthermore, we have demonstrated the improved performance of IFM through finetuning its expert MPC, as evidenced by our extensive evaluation in simulation and on hardware.

V. CONCLUSIONS

We propose a novel Imitating and Finetuning Model Predictive Control framework to obtain an effective locomotion controller with periodic and symmetric gaits. We first develop a model-based controller using DDP and the Raibert heuristic. Then we train an imitation policy using behavior cloning and further finetune it on challenging terrains via deep RL. By finetuning with RL, IFM can have significantly improved robustness compared to its expert policy (MPC) on challenging terrains and also reduces the computation time. By pre-training with MPC, IFM can find better gaits with lower CoT and PPI in a sample-efficient manner compared to Vanilla RL. We conduct comprehensive experiments in both simulation and hardware environments to highlight the benefits of the proposed method.

¹Demonstrated terrains in each paper. F: flat terrain, R: rough terrain, Sl: slippery terrain (or banana peel), C: conveyor belt (or treadmill), St: step terrain.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

One future research direction will be to extend the proposed method for vision-based locomotion so that the policy can actively negotiate the obstacles. We expect that our approach can greatly contribute to more sample-efficient and effective learning for visual locomotion as well. However, we may need to carefully design the imitation learning stage to support more complicated network architectures, such as Long short-term memory or convolutional neural network.

REFERENCES

- [1] S. Hong, J.-H. Kim, and H.-W. Park, "Real-time constrained nonlinear model predictive control on so (3) for dynamic legged locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 3982–3989.
- [2] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1154–1171, Aug. 2021.
- [3] J. D. Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018.
- [4] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 3287–3292.
- [5] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2245–2252.
- [6] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 6295–6301.
- [7] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 2019, arXiv:1909.06586v1.
- [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," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872.
- [9] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4630–4637, Apr. 2022.
- [10] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Sci. Robot.*, vol. 5, no. 49, 2020, Art. no. eabb2174.
- [11] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2950–2957, Apr. 2020.
- [12] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *Proc. Robot.: Sci. Syst.*, 2021, Art. no. 011.
- [13] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, 2022, Art. no. eabk2822.
- [14] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Proc. Robot.: Sci. Syst.*, 2022, Art. no. 022.
- [15] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. 5th Annu. Conf. Robot Learn.*, 2022, pp. 91–100.
- [16] Z. Xie, X. Da, B. Babich, A. Garg, and M. v. de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," 2021, arXiv:2104.09771.
- [17] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *Proc. 5th Annu. Conf. Robot Learn.*, 2021, pp. 773–783.
- [18] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang, "Visual-locomotion: Learning to walk on complex terrains with vision," in *Proc. 5th Annu. Conf. Robot Learn.*, 2021, pp. 1291–1392.
- [19] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a contact-adaptive controller for robust, efficient legged locomotion," 2020, arXiv:2009.10019.
- [20] J. Carius, F. Farshidian, and M. Hutter, "Mpc-net: A first principles guided policy search," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2897–2904, Apr. 2020.
- [21] A. Reske, J. Carius, Y. Ma, F. Farshidian, and M. Hutter, "Imitation learning from mpc for quadrupedal multi-gait control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 5014–5020.
- [22] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 143:1–143:14, Jul. 2018.
- [23] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Proc. Robot.: Sci. Syst.*, 2020, Art. no. 064.
- [24] Y. Fuchioka, Z. Xie, and M. van de Panne, "Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," 2022, arXiv:2210.01247.
- [25] M. Bogdanovic, M. Khadiv, and L. Righetti, "Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization," 2021, arXiv:2107.06629.
- [26] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.
- [27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, arXiv:1707.06347.
- [28] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 5761–5768.
- [29] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.
- [30] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986.
- [31] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," in *Proc. IEEE/CVF Eur. Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 584–599.
- [32] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 2436–2442.
- [33] M. Carroll, Z. Liu, M. Kasaei, and Z. Li, "Agile and versatile robot locomotion via kernel-based residual learning," 2023, arXiv:2302.07343.
- [34] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," in *Proc. Robot.: Sci. Syst.*, June. 2018.
- [35] E. Kaufmann, A. Loquercio, R. Ranfil, M. Müller, V. Koltun, and D. Scaramuzza, "Deep Drone Acrobatics," in *Proc. Robot.: Sci. Syst.*, July. 2020, Art. no. 040.
- [36] A. Tagliabue, D.-K. Kim, M. Everett, and J. P. How, "Demonstration-efficient guided policy search via imitation of robust tube mpc," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 462–468.
- [37] K. Nagami and M. Schwager, "HJB-RL: Initializing Reinforcement Learning with Optimal Control Policies Applied to Autonomous Drone Racing," in *Proc. Robot.: Sci. Syst.*, July. 2021, Art. no. 062.
- [38] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1168–1175.
- [39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, arXiv:1412.6980.
- [40] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," 2017, arXiv:1710.06542.
- [41] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 895–902, Apr. 2018.
- [42] E. T. Hsiao-Wecksler, J. D. Polk, K. S. Rosengren, J. J. Sosnoff, and S. Hong, "A review of new analytic techniques for quantifying symmetry in locomotion," *Sym.*, vol. 2, no. 2, pp. 1135–1155, 2010.