Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Rapid Locomotion via Reinforcement Learning

Gabriel B. Margolis*[1], Ge Yang*[1 2], Kartik Paigwar[1], Tao Chen[1], and Pulkit Agrawal[1 2]
[1]MIT Improbable AI Lab [2]NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Massachusetts Institute of Technology, Cambridge, MA 02139

***Abstract*—Agile maneuvers such as sprinting and high-speed turning in the wild are challenging for legged robots. We present an end-to-end learned controller that achieves record agility for the MIT Mini Cheetah, sustaining speeds up to $3.9\,\mathrm{m/s}$. This system runs and turns fast on natural terrains like grass, ice, and gravel and responds robustly to disturbances. Our controller is a neural network trained in simulation via reinforcement learning and transferred to the real world. The two key components are (i) an adaptive curriculum on velocity commands and (ii) an online system identification strategy for sim-to-real transfer leveraged from prior work. Videos of the robot's behaviors are available at https://agility.csail.mit.edu/.**

I. Introduction

Running fast on natural terrain is challenging. Different terrains exhibit different characteristics, ranging from variable friction and softness to sloped and uneven geometry. As a robot attempts to move at faster speeds, the impact of terrain variation on controller performance increases [6, 12]. Some physical considerations only begin to influence the robot's dynamics at high speeds, including the enforcement of actuator limits [9, 10, 15], the regulation of large contact forces [21], and body control during flight phases [10, 21]. One possibility is to resolve these issues by making targeted improvements to the hand-designed models used in model-based control. Impressive progress has been made in this direction [5, 6, 9, 10, 11, 12, 15, 21]. However, in model-based control, the robot's behavior and robustness are dependent on the creativity and investment of the human designer, who must invent simplified reduced-order models that allow the robot to infer the appropriate actions under the constraint of real-time computation.

How can we perform real-time control in complex environments where efficient reduced-order models may not exist or are currently unknown? One possibility is to optimize the robot's actions with respect to a full physics model. The problem is that trajectory optimization with a full model is not possible in real-time for a complex task such as fast running on natural terrains. An alternative is to amortize the cost of trajectory optimization by learning a direct mapping from sensory observations to actions (a *policy*) using high-reward trajectories sampled from the full model. Reinforcement learning (RL) provides a way to learn such a policy. In this approach, the human designs a set of training environments and reward functions to specify a set of tasks. RL algorithms automatically discover the policy that maximizes reward across these environments and tasks. Because the RL framework does not require a human engineer to design accurate and efficient reduced-order models, it is less reliant on human effort. Consequently, RL offers a scalable controller synthesis scheme for complex tasks in challenging environments. Recent works have successfully employed RL to learn locomotion controllers [23, 24, 27, 29, 35, 37].

Our goal is to construct a system that can traverse terrains at a large range of linear and angular velocities. This corresponds to a multi-task RL setup where running with each combination of linear and angular velocity constitutes a separate task. Akin to prior work, we found that when the robot is trained to walk with a narrow range of commanded velocities, a multi-

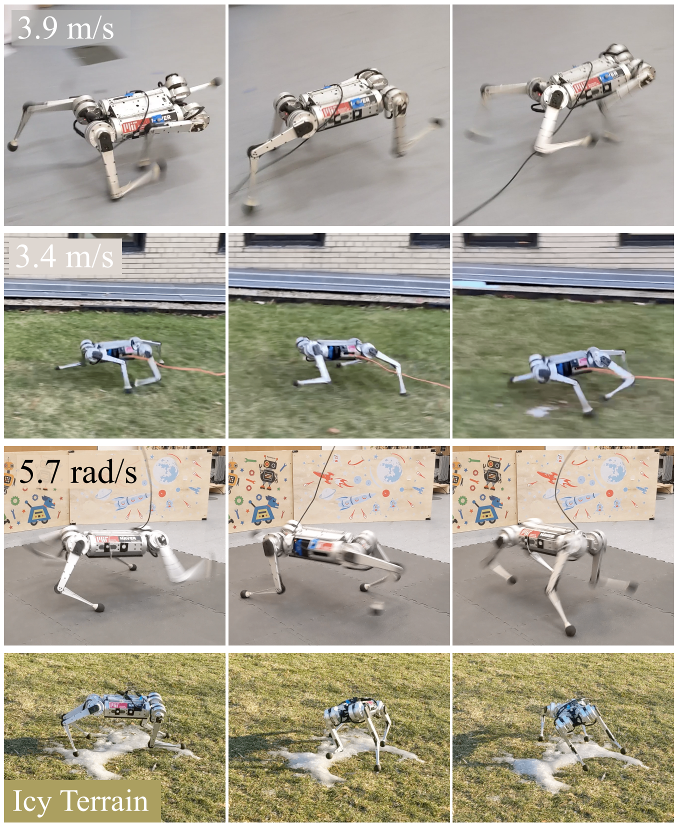


Fig. 1: An end-to-end learned controller enables the MIT Mini Cheetah to execute: (a) fast sprinting at $3.9\,\mathrm{m/s}$ (top); (b) a rough terrain 10-meter sprint at $3.4\,\mathrm{m/s}$; (c) high-speed spinning indoors; and (d) robust spinning on an icy patch. All behaviors are realized by a single neural network that is trained in simulation and deployed zero-shot in the real world.

* Equal contribution. Authors are also affiliated with Computer Science and Artificial Laboratory (CSAIL), the Laboratory for Information and Decision Systems (LIDS), and the MIT-IBM Watson AI Lab at MIT. Correspondence to {gmargo, geyang}@csail.mit.edu

task policy can be successfully learned [23, 24]. However, increasing the range of commanded velocities to include high speeds results in training failure. This issue is reminiscent of difficulty in learning multi-task policies via RL on a broad set of tasks [16]. To understand the reason for failure, consider the naive approach of training a multi-task RL policy by uniformly sampling from all tasks. If most of the tasks are challenging or infeasible, the agent will not gather significant reward, and the training will fail. This is the case in high-speed locomotion: learning to run at rapid velocities from scratch is difficult because physical considerations such as centrifugal force constrain the combinations of linear and angular speed that are realizable.

Training can be made easier by initially providing simple tasks to the agent and then slowly increasing their complexity using a curriculum [4]. Curriculum learning has been leveraged for training robotic systems in the past [25, 28, 35, 41]. Manual curriculum design can fail when the difficulty or feasibility of tasks is not known in advance. For omnidirectional running, manual curriculum design involves finding feasible linear and angular velocity combinations that satisfy physical constraints and ranking velocity commands based on their difficulty. Task difficulty is a function of both the system dynamics and the optimization algorithm, making manual curriculum design tedious and problem-dependent. Instead, we implement an automatic curriculum strategy that expands the set of tasks while respecting the physical constraints of locomotion. The proposed strategy yields significant performance improvements in learning omnidirectional high-speed locomotion.

When deployed in the real world on flat ground, our learned policy sustained a top speed of $3.9\,\mathrm{m/s}$, the highest reported speed for this robot (first row of Figure 1). On uneven outdoor terrain covered with grass, our robot achieved an average speed of $3.4\,\mathrm{m/s}$ for a $10\,\mathrm{m}$ dash (second row of Figure 1). The same policy can spin the robot at $5.7\,\mathrm{rad/s}$ on flat ground and also enables the robot to spin on the more challenging icy terrain (bottom row of Figure 1). We observed additional emergent behaviors during our experiments, including recovery from tripping and compensation for a malfunctioning motor. These results are reported qualitatively, and corresponding videos highlight the diversity of responses that emerge from end-to-end learning.

Our policy uses a minimal sensing suite, consisting only of gyroscope and joint encoders, and is therefore suitable for any typical robot quadruped, including relatively inexpensive commercially available robots. Overall, our system performs rapid locomotion both indoors and outdoors and successfully negotiates challenging terrains and disturbances. Our work fills a gap in the literature. We show that reinforcement learning can be used to learn locomotion controllers that simultaneously achieve linear and angular high-speed behaviors and operate on diverse natural terrains.

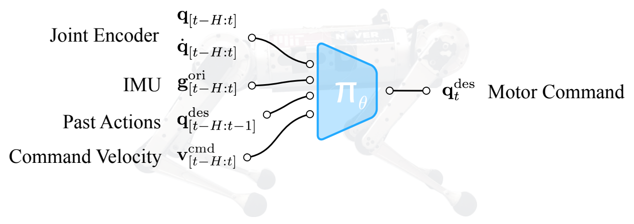

Joint Encoder
$\mathbf{q}_{[t-H:t]}$
$\dot{\mathbf{q}}_{[t-H:t]}$
IMU
$\mathbf{g}^{\text{ori}}_{[t-H:t]}$
Past Actions
$\mathbf{q}^{\text{des}}_{[t-H:t-1]}$
Command Velocity
$\mathbf{v}^{\text{cmd}}_{[t-H:t]}$
π_θ
$\mathbf{q}^{\text{des}}_t$
Motor Command

Fig. 2: Our controller is a learned mapping from sensory inputs to desired joint positions. We parameterize it as 5-layer neural network π_θ with parameters θ optimized in simulation.

II. Experimental Setup

Hardware: We use the MIT Mini Cheetah [20] as our experimental platform. The robot stands $30\,\mathrm{cm}$ tall and weighs $9\,\mathrm{kg}$. It is equipped with 12 quasi-direct-drive actuators capable of maximum output torque of $17\,\mathrm{N\,m}$. The robot's sensor suite consists of joint position encoders and an inertial measurement unit (IMU). Our neural network controller runs at $50\,\mathrm{Hz}$ on an onboard NVIDIA Jetson TX2 NX computer.

Simulation: We use the IsaacGym simulator [26] and code adapted from the open-source repository in [35]. We collect 400 million simulated timesteps using 4000 parallel agents for policy training. This is roughly equivalent to 92 real-time days, which we can simulate in under three hours of wall-clock time using a single NVIDIA RTX 3090 GPU.

III. Method

Our goal is to learn a policy $\pi_\theta(.)$ with parameters θ that takes as input sensory data and velocity commands and gives as output joint position commands (see Figure 2), which are converted into joint torques by a PD controller. The command ($\mathbf{v}^{\text{cmd}}_t$) includes the longitudinal ($\mathbf{v}^{\text{cmd}}_x$) and lateral ($\mathbf{v}^{\text{cmd}}_y$) linear velocities and the yaw rate ($\boldsymbol{\omega}^{\text{cmd}}_z$).

A. Control Architecture

Observation Space: The robot's sensors provide joint angles $\mathbf{q}_t \in \mathbb{R}^{12}$ and joint velocities $\dot{\mathbf{q}}_t \in \mathbb{R}^{12}$, measured using motor encoders, and $\mathbf{g}^{\text{ori}}_t \in \mathbb{R}^3$, which denotes the orientation of the gravity vector in the robot's body frame and is measured using the IMU. As detailed in Section III-C, the policy $\pi_\theta(\cdot)$ takes as input a history of previous observations and actions denoted by $\mathbf{o}_{t-H:t}$ where $\mathbf{o}_t = [\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{g}^{\text{ori}}_t, \mathbf{a}_{t-1}]$. Because we are learning a command-conditioned policy, the input to the policy is $\mathbf{x}_{t-H:t}$ where $\mathbf{x}_t = \mathbf{o}_t \bigoplus \mathbf{v}^{\text{cmd}}_t$. During deployment, the body velocity command $\mathbf{v}^{\text{cmd}}_t$ is specified by a human operator via remote control.

Action Space: The action, $\mathbf{a}_t \in \mathbb{R}^{12}$, assigns joint position commands for a PD controller. The proportional gain is 20, and the derivative gain is 0.5. We chose these low gains to promote smooth motions and did not tune them during our experiments.

Reward Function closely follows [35] with task reward terms for linear and angular velocity tracking, as well as a set of auxiliary terms for stability (velocity penalties on body

Term	Min	Max	Unit
Ground Friction	0.05	4.00	-
Ground Restitution	0.00	1.00	-
Payload Mass	-1.0	3.0	kg
Body Center of Mass	-0.10	0.10	m
Motor Strength	90	110	%
Forward Velocity Command ($\mathbf{v}_x^{\text{cmd}}$)	var.	var.	$\mathrm{m/s}$
Lateral Velocity Command ($\mathbf{v}_y^{\text{cmd}}$)	var.	var.	$\mathrm{m/s}$
Angular Velocity Command ($\boldsymbol{\omega}_z^{\text{cmd}}$)	var.	var.	$\mathrm{rad/s}$

TABLE I: The first set of rows report the ranges of the domain parameters we randomize. The policy is tasked to follow a range of velocity commands that are generated via curriculum strategy described in Section III-D.

Module	Inputs	Hidden Layers	Outputs
Enc. (g_{θ_d})	$\mathbf{d}_t$ (12)	[256, 128]	$\mathbf{z}_t$ (8)
Adapt. (h_{θ_a})	$\mathbf{x}_{[t-h:t-1]}$ (42×15)	[256, 32]	$\mathbf{z}_t$ (8)
Body (π_{θ_b})	$\mathbf{x}_t$ (42), $\mathbf{z}_t$ (8)	[512, 256, 128]	$\mathbf{a}_t$ (12)

TABLE II: Network architecture for encoder g_{θ_d}, adaptation module h_{θ_a}, and policy body π_{θ_b}. The teacher policy is $\pi_T(\mathbf{x}_t, \mathbf{d}_t) = \pi_{\theta_b}(\mathbf{x}_t, g_{\theta_d}(\mathbf{d}_t))$, with parameters θ_b, θ_d optimized using PPO. The student policy, $\pi_S(\mathbf{x}_t, \mathbf{x}_{[t-h:t-1]}) = \pi_{\theta_b}(\mathbf{x}_t, h_{\theta_a}(\mathbf{x}_{[t-h:t-1]}))$ reuses θ_b from the teacher and θ_a is optimized using supervised learning.

roll, pitch, height), smoothness (joint torque and acceleration penalties, action change penalty, footswing duration bonus), and safety (penalty on self-collision, penalty on joint limit violations). We found that the robot tends to sink its body at high speeds and lean into its heading. This motivated us to introduce penalties on the robot's body height and orientation. The details of the reward function are in Table VI (Appendix).

B. Teacher-Student Training

We train a locomotion policy in simulation and transfer it to the real world without fine-tuning. Because the real-world terrain and some of the robot's parameters are not precisely known, it is common practice to train $\pi_\theta(.)$ by randomizing simulation parameters denoted here as $\mathbf{d}_t$. We randomize the body mass, the center of mass, motor strength, ground friction, and ground restitution in the ranges reported in Table I.

One possibility is for the policy to learn a single behavior that works across all the randomized parameters. This learning procedure is commonly referred to as *domain randomization* [38, 39]. Let the resulting policy be $\pi_{DR}(\mathbf{x}_t)$. While $\pi_{DR}(\mathbf{x}_t)$ can cross the sim-to-real gap [38, 39], the learned behavior is conservative [38, 42] because there is no mechanism for the policy to adapt to different domain parameters. For instance, from the same starting state, it makes sense to run on ice in a manner different from running on grass. However, $\pi_{DR}(\mathbf{x}_t)$ has no mechanism for doing so.

To prevent the policy from being conservative, one approach is to include the domain parameters $\mathbf{d}_t$ as part of the policy input [7]. The policy $\pi_T(\mathbf{x}_t, \mathbf{d}_t)$, commonly referred to as a *teacher policy*, is trained using an RL algorithm to maximize the expected sum of rewards. Direct knowledge of the domain parameters provides $\pi_T(\mathbf{x}_t, \mathbf{d}_t)$ with the ability to adapt to different domains. However, this policy cannot be deployed on a real robot since $\mathbf{d}_t$ cannot be directly measured using onboard sensors. To overcome this limitation, one can deploy a *student policy*, $\pi_S(\mathbf{x}_t, \mathbf{x}_{[t-h:t-1]})$ that is trained to mimic the teacher's action via behavior cloning [34]. The main idea is that accurately matching the teacher's actions forces the student to implicitly infer domain parameters ($\mathbf{d}_t$) from a state history of h time steps, $\mathbf{x}_{[t-h:t-1]}$. Therefore, the student policy is said to perform online system identification.

Teacher-student training enables the agent to specialize its behavior to the current dynamics $\mathbf{d}_t$, instead of learning a single behavior that works across different $\mathbf{d}_t$. This so-called implicit system identification approach has been previously developed in a number of works involving object re-orientation with a multi-finger hand [8], self-driving cars [7] and locomotion [23, 24, 27, 29]. Like work applying student-teacher learning to blind walking [23, 24], our teacher policy observes $\mathbf{d}_t$, the dynamic properties of the robot and terrain. The student learns to infer them from $\mathbf{x}_{[t-h:t-1]}$, the history of joint angles, and IMU readings.

C. Policy Optimization

1) Teacher Policy: We construct the teacher policy, $\pi_T(\mathbf{x}_t, \mathbf{d}_t)$, as a composition of two modules g_{θ_d} and π_{θ_b}, such that $\pi_T(\mathbf{x}_t, \mathbf{d}_t) = \pi_{\theta_b}(\mathbf{x}_t, g_{\theta_d}(\mathbf{d}_t))$. The first module g_{θ_d} is the encoder,

$$\mathbf{z}_t = g_{\theta_d}(\mathbf{d}_t)\,, \tag{1}$$

which compresses $\mathbf{d}_t$ into an intermediate latent vector $\mathbf{z}_t$. The second module π_{θ_b} is the policy body,

$$\mathbf{a}_t = \pi_{\theta_b}(\mathbf{x}_t, \mathbf{z}_t)\,, \tag{2}$$

which predicts an action from the latent $\mathbf{z}_t$ and observation $\mathbf{x}_t$. Each module is parameterized as a neural network with ELU activations and architecture described in Table II. We optimize the teacher's parameters θ_d, θ_b together using PPO [36] to maximize the future discounted reward,

$$\max_{\theta_b, \theta_e} \mathbb{E}_{\pi_{\theta_b, \theta_e}}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big]\,. \tag{3}$$

2) Student Policy: The student policy $\pi_S(\mathbf{x}_t, \mathbf{x}_{[t-h:t-1]}) = \pi_{\theta_b}(\mathbf{x}_t, h_{\theta_a}(\mathbf{x}_{[t-h:t-1]}))$ imitates the teacher's behavior during deployment without access to $\mathbf{d}_t$. The student policy is constructed by replacing encoder $g_{\theta_d}(\mathbf{d}_t)$ with an online identification module [23, 24],

$$\hat{\mathbf{z}}_t = h_{\theta_a}(\mathbf{x}_{[t-h:t-1]})\,, \tag{4}$$

which estimates the latent $\hat{\mathbf{z}}_t$ from state history $\mathbf{x}_{[t-h:t-1]}$. We train the identification module so that its predictions $\hat{\mathbf{z}}_t$ match the encoder's output $\mathbf{z}_t = g_{\theta_d}(\mathbf{d}_t)$ as closely as possible. To this end, we optimize parameters θ_a using supervised learning on on-policy data, using the loss function

$$\mathcal{L}_{\theta_d} = \Big(h_{\theta_a}(\mathbf{x}_{[t-h:t-1]}) - g_{\theta_d}(\mathbf{d}_t)\Big)^2 = (\hat{\mathbf{z}}_t - \mathbf{z}_t)^2\,. \quad (5)$$

When this loss is low, the latent representation $\mathbf{z}_t$ is shared between the teacher and the student, so the student can reuse the teacher's policy body module as $\mathbf{a}_t = \pi_{\theta_b}(\hat{\mathbf{z}}_t, \mathbf{x}_t)$ to select actions without further training.

The optimization procedure closely follows that of [23, 24] with a few minor differences: (1) We use a shorter history of $h = 15$ observations, small enough for the adaptation module to run in real-time synchronously with the policy body; (2) We train the adaptation module simultaneously with the teacher using on-policy data. We found that the robot's ability to run at high speed was not sensitive to these design choices, which made training and deployment easier. Table II gives the architecture of each component of the system. PPO hyperparameters are listed in Table V (Appendix).

D. Curriculum Strategy

The agent learns a velocity-conditioned policy by attempting to track different velocity commands during training. To this end, the longitudinal and yaw velocity commands $\mathbf{v}_x^{\text{cmd}}$, $\boldsymbol{\omega}_z^{\text{cmd}}$ during episode k are sampled from a probability distribution $p^k_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot)$. The lateral velocity command $\mathbf{v}_y^{\text{cmd}}$ is sampled separately from a small uniform probability distribution because longitudinal and yaw speed are sufficient for omnidirectional locomotion. Without a curriculum, there is no change in the sampling procedure from episode to episode:

$$p^{k+1}_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot) \leftarrow p^{k}_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot)\,. \quad (6)$$

When velocity commands are sampled uniformly from a small range ($\mathbf{v}_x^{\text{cmd}} \in [-1.0, 1.0]$, $\boldsymbol{\omega}_z^{\text{cmd}} \in [-1.0, 1.0]$) at the start of training, the agent can learn to track them [17, 23]. However, when commanded velocities are sampled uniformly from a large distribution ($\mathbf{v}_x^{\text{cmd}} \in [-4.0, 4.0]$, $\boldsymbol{\omega}_z^{\text{cmd}} \in [-5.0, 5.0]$), we found that learning fails (Figure 3).

The reason for failure is that locomotion at high speeds is challenging, and if most of the commands are high-velocity, the agent fails to gather enough reward. This problem may be mitigated if we first expose the agent to low-velocity commands and gradually increase the desired speed via a curriculum [4]). Some works use a curriculum where the commands are updated on a fixed schedule, as a function of the timing variable k. This update rule f takes the form:

$$p^{k+1}_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot) \leftarrow f\big(p^{k}_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot), k\big)\,. \quad (7)$$

A fixed schedule requires manual tuning. Moreover, if the system designer modifies the environment or learning algorithm, the agent's learning speed will be different, which would necessitate re-tuning the curriculum schedule. Rather than advancing the command curriculum on a fixed schedule, we automatically update the curriculum using a reward-based rule [2, 25, 28, 41]. One possibility is to maintain independent distributions over command dimensions $p_{\mathbf{v}_x}(\cdot), p_{\boldsymbol{\omega}_z}(\cdot)$ such that $p_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot) = p_{\mathbf{v}_x}(\cdot)p_{\boldsymbol{\omega}_z}(\cdot)$, and to specify the update rules f_v, f_ω for each component separately:

$$p^{k+1}_{\mathbf{v}_x}(\cdot) \leftarrow f_v\big(p^{k}_{\mathbf{v}_x}(\cdot), r_{v_x}\big)\,, \quad (8a)$$

$$p^{k+1}_{\boldsymbol{\omega}_z}(\cdot) \leftarrow f_\omega\big(p^{k}_{\boldsymbol{\omega}_z}(\cdot), r_{\omega_z}\big)\,, \quad (8b)$$

where r_{v_x} and r_{ω_z} are the velocity tracking rewards as detailed in Table VI. We refer to this approach as the Box Adaptive curriculum because the probability density function in the $\mathbf{v}_x$-$\boldsymbol{\omega}_z$ plane is shaped like a box.

If $\mathbf{v}_x$ and $\boldsymbol{\omega}_z$ are chosen independently, then commands with both high linear and angular velocity will be sampled equally as often as commands with just one large velocity component. However, due to the effects of centrifugal force at high speeds, simultaneous running and turning are much more demanding than fast straight-line running or spinning in place. A curriculum that independently increases the linear and angular velocities might fail to discover some behaviors because most high-speed commands are infeasible. This motivates us to use a curriculum strategy that models the joint distribution over linear and angular velocity commands:

$$p^{k+1}_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot) \leftarrow f\big(p^{k}_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot), r_{v_x}, r_{\omega_z}\big)\,. \quad (9)$$

We refer to this as the Grid Adaptive curriculum.

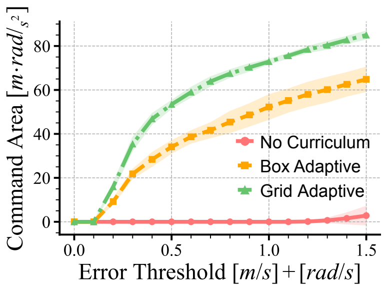
Having described the form for the two curriculum strategies, we will now provide the detailed update rules. For both strategies we initialize $p^k_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot)$ as a uniform probability distribution over ($\mathbf{v}_x^{\text{cmd}} \in [-1.0, 1.0]$, $\boldsymbol{\omega}_z^{\text{cmd}} \in [-1.0, 1.0]$). We represent distribution $p^k_{\mathbf{v}_x,\boldsymbol{\omega}_z}(\cdot,\cdot)$ as a discrete grid with resolution $[0.5\,\text{m/s}, 0.5\,\text{rad/s}]$ centered at $[0\,\text{m/s}, 0\,\text{rad/s}]$. To control the growth of the sampling distribution, we define a success threshold, γ, with constant value between 0 and 1.

1) Box Adaptive Curriculum Update Rule: At episode k, the linear and angular velocity commands for the agent are sampled independently: $\mathbf{v}_x^{\text{cmd}} \sim p^k_{\mathbf{v}}(\cdot)$, $\boldsymbol{\omega}_z^{\text{cmd}} \sim p^k_{\boldsymbol{\omega}_z}(\cdot)$. If the agent succeeds in this region of command space, we would like to add neighboring regions to the sampling distribution, potentially increasing the difficulty of future commands. Suppose the agent receives rewards $r_{v_x^{\text{cmd}}}, r_{\omega_z^{\text{cmd}}}$ in its attempt to follow $\mathbf{v}_x^{\text{cmd}}$, $\boldsymbol{\omega}_z^{\text{cmd}}$. Then we apply the update rule

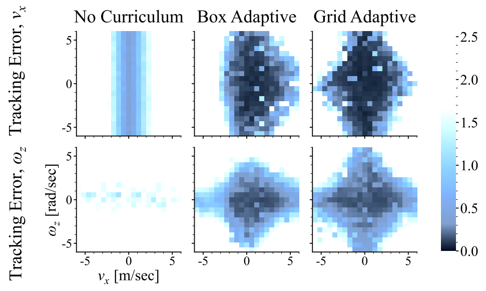
$$p^{k+1}_{\mathbf{v}_x}(\mathbf{v}_x^{\text{n}}) \leftarrow \begin{cases} p^k_{\mathbf{v}_x}(\mathbf{v}_x^{\text{n}}) & r_{v_x^{\text{cmd}}} < \gamma\,, \\ 1 & \text{otherwise.} \end{cases} \quad (10a)$$

$$p^{k+1}_{\boldsymbol{\omega}_z}(\boldsymbol{\omega}_z^{\text{n}}) \leftarrow \begin{cases} p^k_{\boldsymbol{\omega}_z}(\boldsymbol{\omega}_z^{\text{n}}) & r_{\omega_z^{\text{cmd}}} < \gamma\,, \\ 1 & \text{otherwise.} \end{cases} \quad (10b)$$

which increases the probability density on neighbors $\mathbf{v}_x^{\text{n}}$ of $\mathbf{v}_x^{\text{cmd}}$ and $\boldsymbol{\omega}_z^{\text{n}}$ of $\boldsymbol{\omega}_z^{\text{cmd}}$. Here, neighboring commands are defined as the adjacent elements in the (discretized) domain of each marginal distribution: $\mathbf{v}_x^{\text{n}} \in \{\mathbf{v}_x^{\text{cmd}} - 0.5, \mathbf{v}_x^{\text{cmd}} + 0.5\}$ and $\boldsymbol{\omega}_z^{\text{n}} \in \{\boldsymbol{\omega}_z^{\text{cmd}} - 0.5, \boldsymbol{\omega}_z^{\text{cmd}} + 0.5\}$. Suppose $\mathbf{v}_x^{\text{cmd}}$ or $\boldsymbol{\omega}_z^{\text{cmd}}$ is among the most challenging commands in one of the distributions,



(a) Command area vs. error threshold



(b) Heatmap of converged tracking error for curricular strategies.

Fig. 3: (a) Forward and angular velocity tracking performance. The Grid Adaptive curriculum tracks a larger range of velocities than the Box Adaptive curriculum for all error thresholds. (b) Velocity tracking error in the forward axis (top) and yaw axis (bottom); darker is better. In each heatmap, the x-axis varies the forward velocity command between $[-6, 6\,\mathrm{m/s}]$ and the y-axis varies the yaw rate between $[-6, 6\,\mathrm{rad/s}]$. From left to right: *No Curriculum* fails to learn meaningful velocity control; its heatmaps correspond to a robot jittering in place, as its tracking error is equal to the command. *Box Adaptive* curriculum learns to control the robot but excludes extremes of the command space. *Grid Adaptive* curriculum covers a larger command area by accounting for the combined impact of running and turning speed on task difficulty.

and the reward threshold is met. In that case, this update will result in that distribution expanding.

2) Grid Adaptive Curriculum Update Rule: At episode k, the linear and angular velocity commands for the agent are sampled from the joint distribution: $\mathbf{v}_x^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}} \sim p^k_{\mathbf{v}_x, \boldsymbol{\omega}_z}(\cdot, \cdot)$. As before, if the agent succeeds in this region of command space, we would like to increase the difficulty by adding neighboring regions to the sampling distribution. However, the distributions of $\mathbf{v}_x^{\mathrm{cmd}}$ and $\mathbf{v}_y^{\mathrm{cmd}}$ are no longer constrained to be independent. This enables us to revise our update with a new definition of the neighboring commands. Upon termination of an episode with command $[\mathbf{v}_x^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}}]$ where the agent received rewards $r_{v_x^{\mathrm{cmd}}}, r_{\omega_z^{\mathrm{cmd}}}$, we use the following update:

$$p^{k+1}_{\mathbf{v}_x, \boldsymbol{\omega}_z}(\mathbf{v}_x^{\mathrm{n}}, \boldsymbol{\omega}_z^{\mathrm{n}}) \leftarrow \begin{cases} p^k_{\mathbf{v}_x, \boldsymbol{\omega}_z}(\mathbf{v}_x^{\mathrm{n}}, \boldsymbol{\omega}_z^{\mathrm{n}}) & r_{v_x^{\mathrm{cmd}}} < \gamma \text{ or } r_{\omega_z^{\mathrm{cmd}}} < \gamma, \\ 1 & \text{otherwise.} \end{cases} \tag{11}$$

This update adds probability density to the neighboring velocity commands $[\mathbf{v}_x^{\mathrm{n}}, \boldsymbol{\omega}_z^{\mathrm{n}}]$ of $[\mathbf{v}_x^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}}]$, if those commands have not already been added. Here, neighboring commands are defined as neighbors in the 4-connected grid domain of $p^k_{\mathbf{v}_x, \boldsymbol{\omega}_z}(\cdot, \cdot)$, which is a discrete grid with resolution $[0.5\,\mathrm{m/s}, 0.5\,\mathrm{rad/s}]$. If $[\mathbf{v}_x^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}}]$ is among the most challenging commands in the joint distribution, and the reward threshold is met, this update will result in the distribution expanding locally.

E. Evaluation Metrics

The controller is tasked to track body velocity commands. Consider a command: $(\mathbf{v}_x^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}})$ corresponding to a point in the $\mathbf{v}_x^{\mathrm{cmd}}$-$\boldsymbol{\omega}_z^{\mathrm{cmd}}$ plane. We discretize this plane into a grid with resolution $[0.5\,\mathrm{m/s},\ 0.5\,\mathrm{rad/s}]$ with grid cell indices denoted as i, j. Then, for each grid cell, we define the tracking error ϵ_{ij} as the root mean square deviation, averaged over trials in that grid cell:

$$\epsilon_{ij}[\mathbf{v}_x^{\mathrm{cmd}}] = \mathbb{E}_{\mathbf{v}_x^{\mathrm{cmd}} \sim [i-1, i], \boldsymbol{\omega}_z^{\mathrm{cmd}} \sim [j-1, j]} \sqrt{\mathbb{E}_t(\mathbf{v}_x^{\mathrm{cmd}} - \mathbf{v}_x^t)^2}, \tag{12a}$$

$$\epsilon_{ij}[\boldsymbol{\omega}_z^{\mathrm{cmd}}] = \mathbb{E}_{\mathbf{v}_x^{\mathrm{cmd}} \sim [i-1, i], \boldsymbol{\omega}_z^{\mathrm{cmd}} \sim [j-1, j]} \sqrt{\mathbb{E}_t(\boldsymbol{\omega}_z^{\mathrm{cmd}} - \boldsymbol{\omega}_z^t)^2}, \tag{12b}$$

where $\mathbf{v}_x^t, \boldsymbol{\omega}_z^t$ are the forward and yaw velocity of the robot measured at time t. In our experiments, we compute tracking error from 5 trials per grid cell.

Measuring either the longitudinal or yaw velocity in isolation does not provide a complete picture of controller performance. Instead, we want a metric that captures the combinations of longitudinal and yaw velocity that the robot is able to track. To this end, we constructed an aggregate metric that captures the diversity of commands the controller can actuate given some maximum error tolerance. For a certain error threshold ϵ_0, we define the *command area* as the area of the region in the $\mathbf{v}_x^{\mathrm{cmd}}$-$\boldsymbol{\omega}_z^{\mathrm{cmd}}$ plane for which the tracking errors satisfy

$$\epsilon_{ij}[\mathbf{v}_x^{\mathrm{cmd}}] + \epsilon_{ij}[\boldsymbol{\omega}_z^{\mathrm{cmd}}] < \epsilon_0. \tag{13}$$

The dimension of the command area is $\mathrm{m/s} \cdot \mathrm{rad/s}$. Intuitively, if one controller has a larger command area than another, the former can achieve a greater range of speeds while remaining below the same error threshold ϵ_0. When we report the command area, we evaluate policies trained with five random seeds and indicate their standard deviation using an error bar.

IV. Results

A. Curriculum Learning Enables High-Speed Locomotion

Figure 3(a) visualizes the tracking error (see Section III-E) of the policies learned from the three command sampling strategies as heatmaps in the $\mathbf{v}_x^{\mathrm{cmd}}$-$\boldsymbol{\omega}_z^{\mathrm{cmd}}$ plane. The shading on each heatmap corresponds to tracking error, with darker

	Robot	RL?	Froude (-)	Speed m/s	Leg L cm
Park et al.	C2		7.1	6.4	59
Ours, Ji et al.	MC	**Y**	**5.1**	**3.9**	30
Kim et al.	MC		4.6	3.7	30
Unitree	A1		2.8	3.3	40
Kumar et al.	A1	Y	0.8	1.8	40
Hwangbo et al.	ANYmal	Y	0.5	1.5	50

TABLE III: Measure of Agility: Comparison between the Froude numbers of various prior works. MC = Mini Cheetah; C2 = Cheetah 2. We are the first to demonstrate that reinforcement learning (RL) achieves agile locomotion with Froude number ≥ 1 (along with concurrent work Ji et al.).

	$\mathbf{v}_x^{\text{cmd}}$	$\mathbf{v}_x$ (Sim)	$\mathbf{v}_x$ (Real)
With System ID ($\pi_{\theta_{ST}}$)	$6.0\,\mathrm{m/s}$	5.46	3.81 ± 0.09 (3)
W/o System ID ($\pi_{\theta_{DR}}$)	$6.0\,\mathrm{m/s}$	5.07	2.49 ± 0.07 (2)

TABLE IV: Quantifying the sim-to-real gap in the maximum-velocity regime. Velocity on the real robot is measured using a precise motion capture setup.

shades indicating lower error. We observe that the policy trained without any curriculum fails to learn. This is because the robot's random exploration at the start of training rarely results in fast body motion. Hence, the reward almost always remains small, providing minimal learning signal.

The performance of the system is improved substantially by implementing the Box Curriculum. The agent first learns to track well in the small initial command distribution, then gradually increases its capability as the commands become larger.

Using the Grid Curriculum, the performance of the policy further improves, as evidenced by the larger command area. It achieves this by maintaining a full joint distribution over linear and angular velocity, thereby modeling their interaction. When high linear and angular velocities are combined, a body experiences a centrifugal force which must be countered by frictional force to remain on the desired path. This force balance induces a constraint on maximum combinations of linear and angular velocity such that the two vary inversely $[\boldsymbol{\omega}_z \sim 1/\mathbf{v}_x]$ when the constraint is active. This phenomenon is in agreement with the apparent inverse shape of the command area boundary shown in Figure 3b, which suggests that the robot has reached a physical limit on its ability to turn at high speed. The Grid Curriculum can limit itself to sampling combinations of linear and angular speed that are jointly feasible. In contrast, because the Box Curriculum samples linear and angular velocity independently, it will frequently generate infeasible high-speed tasks that hinder learning.

B. Real-world Testing

Video of all experiments described in this section is viewable on the project website: https://agility.csail.mit.edu/.

Indoor Running To evaluate how fast our robot can run in the real world, we ramped the velocity command to $6.0\,\mathrm{m/s}$. We conducted this experiment in a motion capture arena to accurately estimate the robot's running speed (Figure 1, top). We found that policies trained with a system identification module and grid curriculum sustained an average speed of $3.8\,\mathrm{m/s}$ across multiple seeds (Table IV), with the highest sustained speed of $3.9\,\mathrm{m/s}$ among the three seeds. This is higher than the previous record of $3.7\,\mathrm{m/s}$ reported for a model-predictive control algorithm on the same robot [21]. Together with concurrent work [18], this is substantially faster than previous applications of RL to legged locomotion.

The maximum attainable speed is intimately tied to the robot's hardware properties, such as its weight, motor strength, and leg length. Although there is no perfect way to compare agility across different robot designs, *Froude number* [3] normalizes a robot's speed by its leg length and has been used to measure agility across robot platforms in the past [30]. Table III compares the Froude numbers across different quadrupeds and controllers. Along with the concurrent work of Ji et al. [18], our method is substantially more agile than previous applications of reinforcement learning.

While our robot successfully performs rapid locomotion in the real world, there exists a sim-to-real gap as reported in Table IV. The results reveal that online system identification leads to better tracking of the velocity command of $6.0\,\mathrm{m/s}$ in simulation (speed of $5.46\,\mathrm{m/s}$ with and $5.07\,\mathrm{m/s}$ without system identification), and also reduces the sim-to-real gap (average speeds of $3.81\,\mathrm{m/s}$ with and $2.49\,\mathrm{m/s}$ without system identification). While prior work demonstrated that sim-to-real gap can be mitigated at low velocities [23, 24], our results show that these findings also hold true at high speeds.

Some of the remaining sim-to-real performance gap may result from an inaccurate selection of simulated terrain and robot parameters for evaluation. For example, the configuration used for simulated evaluation might overestimate the robot's effective motor strength or the ground friction, resulting in a different maximum speed. On the other hand, some aspects of the real-world dynamics are probably not captured under any configuration of the simulator. This type of sim-to-real gap could result in suboptimal real-world top speed. The relative contributions of these factors to the sim-to-real performance gap remains uncertain.

Outdoor Running Outdoor terrain presents multiple challenges not present in indoor running, among which are changes in ground height, friction, and terrain deformation. Under these variations, the robot must actuate its joints differently to reach high speed than it would on flat, rigid terrain with high friction, such as a treadmill or paved road. To test if our system can run on outdoor terrains, we conducted an outdoor dash across an uneven grassy patch as shown in Figure 1 (second row). We record an outdoor 10-meter dash time of 2.94 seconds, corresponding to an average speed of $3.4\,\mathrm{m/s}$.

Yaw Control We evaluate our controller's yaw velocity control in the lab setting as shown in Figure 1 (third row). The robot accelerates to a maximum yaw rate of $5.7\,\mathrm{rad/s}$, then

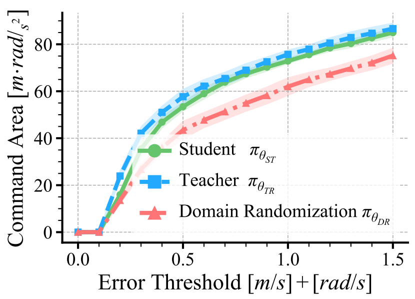
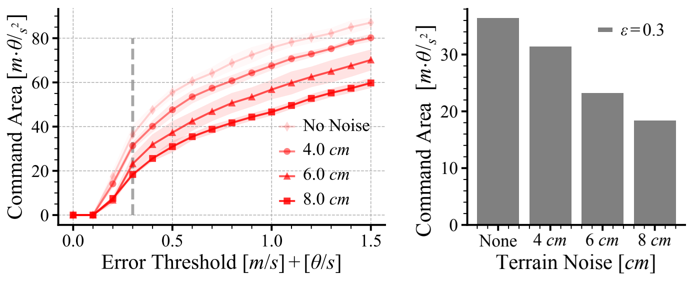


Fig. 4: Online system identification reduces tracking error, particularly at high speeds. The command area increases as the error threshold is relaxed for teacher, student, and domain-randomized policies.



(a) Command area vs. error threshold (b) Slice along $\epsilon = 0.3$

Fig. 5: (a) Increasing the magnitude of terrain roughness during training shrinks the range of commands the robot can successfully track – the command area – on flat ground. This reflects a trade-off between robustness to rough terrains and top speed on flat ground. (b) Histogram comparing the command area at error threshold $\epsilon = 0.3$, which corresponds to the **gray** vertical line on the left.

stops safely. This is 90% of the fastest yaw rate recorded on the Mini Cheetah using a model-based controller, at 6.28 rad/s [5]. However, the model-based records were achieved using two different controllers for linear [21] and angular [5] velocity. In contrast, a single policy achieved all indoor and outdoor running and spinning results in our work. To challenge the controller's spinning skills, we brought the robot outside after a snowstorm and piloted it onto an icy patch, illustrated in Figure 1 (bottom). The robot maintained stability while spinning as its feet frequently slipped on ice.

Response to Terrain Changes and Hardware Failures We tested our system in a diverse set of challenging real-world scenarios: (1) ascending a steep incline made of small pebbles. (2) maintaining balance despite a mechanical blockage to one motor. (3) tripping at high speed, flying upside down, and landing on its feet. (4) recovering via a change in gait after tripping over a small barrier. We present these qualitative results in the accompanying video.

We also deployed the model-predictive controller from [21] in scenarios (1) and (4), which were the most convenient to replicate. Unlike our learned controller, the baseline did not recover from (1) slipping down the gravelly incline and (4) tripping over the barrier. While these results highlight the robustness of policies, we want to emphasize that we are not claiming that such (or even more) robustness cannot be achieved with model-predictive control. Our claim is simply that by freeing the human from the tedious task of refining the robot's model or behavior, the RL paradigm offers a scalable alternative to obtain robust behavior in diverse conditions.

C. Ablation Studies

1) Impact of Online System Identification: System identification can become both more critical and more challenging as locomotion speed increases; this has been previously suggested by studies of model-based control systems [6, 12], but has not been explored in the context of reinforcement learning. We evaluate this hypothesis in the teacher-student setting by quantifying (1) the benefit of access to privileged information when learning to run at high speeds, and (2) the ability of the student policy to retain the performance gains using only available sensor data. We compare teacher, student, and domain randomized policies as described in Section III-B in the high-speed regime. All policies are trained under the same randomization of the privileged state (Table I).

We find that access to privileged information yields increased performance across all speeds, with the greatest benefit at high speeds. Figure 4 plots the command area (Section III-E) for the three policies as the threshold for error increases. The privileged teacher π_T trained with access to environment parameters attains a strictly larger command area than the policy π_{DR} trained with only the robot state. Using the online system identification module, we show that the student policy π_S can nearly match the teacher's performance. The student's ability to imitate the teacher is consistent across all speeds.

2) Impact of Rough-Terrain Training: One might hypothesize that for a system to operate on rough terrains, it must also be trained on rough terrains. The strategy of training on rough terrains has been applied successfully in prior works [23, 24, 37] to enable robust locomotion on diverse terrains. We find that despite training only on flat ground, our policy is sufficiently robust to deploy on various outdoor terrains. Moreover, for a blind policy, there is a trade-off between speed on flat ground and robustness on uneven terrain. Figure 5 reports the decrease in performance brought on by introducing terrain roughness when training a high-velocity locomotion policy.

V. Related Work

Model-based Control for Locomotion Seminal work in the field used simplified models and hand-specified gaits to make legged robots balance and move dynamically [14, 19, 32]. Subsequent works introduced expanded models with layered control architectures capable of operation on subsets of rough, soft, and slippery terrains [6, 12, 22, 30, 31, 33].

Recent innovations have addressed specific limitations of simple models with respect to high-speed running. Whole-body control [10] enables simultaneous modeling of robot dynamics and kinematic constraints in real-time. By framing the whole-body control task as one of regulating ground reaction forces, [21] formulated a controller capable of running at speeds up to 3.7m/s on the Mini Cheetah. Regularized Predictive Control [5] additionally applied learned heuristics within a model-based framework, which expanded the robot's ability to spin at high speeds and make tight cornering turns.

Reinforcement Learning for Locomotion [38] combined model-free reinforcement learning with dynamics randomization to learn fast trotting and bounding controllers for the Minitaur robot to move at a fixed speed and direction on flat ground. Extending this approach, [17] trained a velocity-tracking controller for the ANYmal robot for speeds up to $1.5\,\mathrm{m/s}$, and [40] applied sim-to-real reinforcement learning for agile locomotion on the Cassie biped. Followup works expanded ANYmal's robustness by training on diverse terrains using the teacher-student learning paradigm [24, 29, 35]. The mechanical design of the ANYmal robot is thought to limit it from running at higher speeds. [13, 23] investigated the capability of model-free controllers to efficiently traverse diverse terrains on the Unitree A1, a small robot with similar size, actuation, and cost to the Mini Cheetah. Although the A1's built-in MPC controller has a maximum running speed of $3.3\,\mathrm{m/s}$, these learning-based works only demonstrated the robot running up to maximum speed $1.8\,\mathrm{m/s}$.

Concurrently with our work, Ji et al. [18] also trained agile running policies for the Mini Cheetah robot using reinforcement learning. Unlike our work, [18] used a fixed-schedule curriculum on forward linear velocity only. In agreement with our results, [18] concluded that online system identification improves sim-to-real transfer in the high-speed locomotion setting. Instead of learning as we did to implicitly adapt to different environments, [18] learned to explicitly estimate robot state components such as body velocity and contact probability.

Curricula for On-Policy Reinforcement Learning Prior works have shown that a curriculum on environments can enable the discovery of behaviors that are challenging to learn directly using reinforcement learning [4]. [2] demonstrated an Automatic Domain Randomization strategy in which domain randomization scales are increased based on agent performance. Curricula on environments have also been demonstrated in locomotion context; [24, 29, 35] applied a curriculum on terrains to learn highly robust walking controllers on non-flat ground. [41] notably evaluated terrain curriculum strategies, including adaptive curricula, in the setting of stepping stone traversal with a physically simulated biped.

Student-Teacher Training Learning with a privileged teacher has been leveraged for robotics in a number of previous works. [23, 24] applied this approach to the task of blind walking. The teacher policy observed $\mathbf{d}_t$, the dynamic properties of the robot and terrain, and the student learned to infer them from $\mathbf{x}_{[t-h:t-1]}$, the history of joint angles and IMU readings. [27, 29] used the same approach to incorporate terrain geometric information into a locomotion policy. In these works, $\mathbf{d}_t$ was a ground-truth geometric heightmap of the terrain. In [29], the student policy observed a noisy heightmap. In [27], the student policy observed a forward-facing depth image. [8] applied the teacher-student training approach to the task of object reorientation using a dexterous five-fingered hand. In this work, $\mathbf{d}_t$ included the true position of the object as well as the ground-truth state of the hand's fingers. The student policy learned to imitate the teacher using point cloud observations and noisy joint angle readings that could be obtained in the real world.

VI. Discussion

This work has shown that a neural network controller trained fully end-to-end in simulation can push a small quadruped to the limits of its agility, achieving omnidirectional mobility competitive with well-engineered model-predictive controllers in the regime of high speed. Because our controller uses minimal sensing, we can implement it on a low-cost robot [20] with commercially available analogues [1]. Therefore, our method can be readily tested and built upon by others using relatively accessible materials.

Our controller achieves a Froude number of 5.1, which is the highest reported on the Mini Cheetah but lower than the fastest published system, the Cheetah 2 [30]. The Froude number does not capture the impact of any mechanical differences between robots except leg length. We can only speculate whether the agility of alternative platforms comes from improved analytical controllers or hardware differences such as motor strength and weight distribution.

Instrumentation and repeatability limited our ability to characterize the robot's outdoor performance fully. We cannot use motion capture to record the robot's state outdoors as we do in the lab. Also, it is unsafe and impractical to record a large number of high-speed trips or flips on a real robot. This constrained our analysis of the robot's outdoor behavior to be more qualitative while we performed our quantitative analysis in the laboratory setting.

The behaviors we demonstrate in this work are diverse but still limited relative to the full space of possible locomotion tasks. The system we demonstrate has only been trained to control the robot's body velocity in the ground plane. Other categories of behavior such as jumping, crouching, choreographed dance, and loco-manipulation were outside the scope of this work and would potentially require a very different task specification. Our system also does not use vision, so in general, it cannot perform tasks that require planning ahead of time, like efficiently ascending stairs or avoiding pitfalls.

Finally, we emphasize that while our system demonstrates high speed, its distinctive locomotion gait should not be interpreted as generally "better" than the many possible alternatives. On the contrary, many users of legged robots wish to optimize for objectives beyond speed, such as energy efficiency or minimization of wear on the robot. Body speed alone is an underspecified objective, meaning that there may be many equally preferable motions that attain the same speed. Combining learned agile locomotion with additional specifications such as auxiliary objectives or human preferences remains a promising direction for future work.

Acknowledgment

The authors thank the members of the Improbable AI Lab and the Biomimetic Robotics Laboratory for providing valuable feedback on the project direction and the manuscript. We are grateful to MIT Supercloud and the Lincoln Laboratory Supercomputing Center for providing HPC resources. The Mini Cheetah robot used in this work was donated by the MIT Biomimetic Robotics Laboratory and NAVER. The Biomimetic Robotics Laboratory also provided hardware support for the robot. This research was supported by the DARPA Machine Common Sense Program, the MIT-IBM Watson AI Lab, and the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://iaifi.org/). This research was also sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

[1] Unitree Robotics, A1, 2022, https://www.unitree.com/products/a1, [Online; accessed Apr. 2022].

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's cube with a robot hand. *arXiv preprint*, 2019. doi: 10.48550/arXiv.1910.07113.

[3] R McN Alexander. The gaits of bipedal and quadrupedal animals. *Int. J. Robot. Res. (IJRR)*, 3(2):49–59, June 1984. doi: 10.1177/027836498400300205.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 41–48, Montreal, Canada, June 2009. doi: 10.1145/1553374.1553380.

[5] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot. In *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 2245–2252, Madrid, Spain, October 2018. doi: 10.1109/IROS.2018.8593885.

[6] Will Bosworth, Jonas Whitney, Sangbae Kim, and Neville Hogan. Robot locomotion on hard and soft ground: Measuring stability and ground properties in-situ. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 3582–3589, Stockholm, Sweden, May 2016. doi: 10.1109/ICRA.2016.7487541.

[7] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Proc. Conf. Robot Learn. (CoRL)*, pages 66–75, Virtual, November 2020. doi: 10.48550/arXiv.1912.12294.

[8] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Proc. Conf. Robot Learn. (CoRL)*, pages 297–307, London, UK, November 2021. doi: 10.48550/arXiv.2111.03043.

[9] Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors. In *Proc. IEEE/RAS Int. Conf. Humanoid Robot. (Humanoids)*, pages 1–8, Munich, Germany, July 2021. doi: 10.1109/HUMANOIDS47582.2021.9555782.

[10] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *Proc. IEEE/RAS Int. Conf. Humanoid Robot. (Humanoids)*, pages 295–302, Madrid, Spain, November 2014. doi: 10.1109/HUMANOIDS.2014.7041375.

[11] Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 8484–8490, Montreal, Canada, May 2019. doi: 10.1109/ICRA.2019.8793669.

[12] Shamel Fahmi, Michele Focchi, Andreea Radulescu, Geoff Fink, Victor Barasuol, and Claudio Semini. STANCE: Locomotion adaptation over soft terrain. *IEEE Trans. Robot. (T-RO)*, 36(2):443–457, April 2020. doi: 10.1109/TRO.2019.2954670.

[13] Zipeng Fu, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. In *Proc. Conf. Robot Learn. (CoRL)*, pages 928–937, London, UK, November 2021. doi: 10.48550/arXiv.2111.01674.

[14] Andrei Herdt, Nicolas Perrin, and Pierre-Brice Wieber. Walking without thinking about it. In *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 190–195, Taipei, Taiwan, October 2010. doi: 10.1109/IROS.2010.5654429.

[15] Alexander Herzog, Stefan Schaal, and Ludovic Righetti. Structured contact force optimization for kino-dynamic motion generation. In *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 2703–2710, Daejeon, Korea, October 2016. doi: 10.1109/IROS.2016.7759420.

[16] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

[17] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.*, 4(26):aau5872, January 2019. doi: 10.1126/scirobotics.aau5872.

[18] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged lo-

comotion. *IEEE Robot. Automat. Lett. (RA-L)*, 7(2):4630 – 4637, April 2022. doi: 10.1109/LRA.2022.3151396.

[19] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, volume 2, pages 1620–1626, Taipei, Taiwan, September 2003. doi: 10.1109/ROBOT.2003.1241826.

[20] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini Cheetah: A platform for pushing the limits of dynamic quadruped control. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 6295–6301, Montreal, QC, Canada, May 2019. doi: 10.1109/ICRA.2019.8793865.

[21] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint*, 2019. doi: 10.48550/arXiv.1909.06586.

[22] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Auton. Robot.*, 40(3): 429–455, July 2015. doi: 10.1007/s10514-015-9479-3.

[23] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Proc. Robot.: Sci. and Syst. (RSS)*, Virtual, July 2021. doi: 10.48550/arXiv.2107.04034.

[24] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.*, 5 (47):eabc5986, October 2020. doi: 10.1126/scirobotics.abc5986.

[25] Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 4051–4058, Virtual, May 2020. doi: 10.1109/ICRA40945.2020.9197468.

[26] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint*, 2021. doi: 10.48550/arXiv.2108.10470.

[27] Gabriel B Margolis, Tao Chen, Kartik Paigwar, Xiang Fu, Donghyun Kim, Sangbae Kim, and Pulkit Agrawal. Learning to jump from pixels. In *Proc. Conf. Robot Learn. (CoRL)*, pages 1025–1034, London, UK, November 2021. doi: 10.48550/arXiv.2110.15344.

[28] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE Trans. Neural Net. Lrn. Sys.*, 31(9):3732 – 3740, September 2019. doi: 10.1109/TNNLS.2019.2934906.

[29] Takahiro Miki, Joonho Lee, Jemin Hwanbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.*, 7(62):abk2822, January 2022. doi: 10.1126/scirobotics.abk2822.

[30] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *Int. J. Robot. Res. (IJRR)*, 36(2): 167–192, March 2017. doi: 10.1177/0278364917694244.

[31] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825, March 2008. doi: 10.3182/20080706-5-KR-1001.01833.

[32] Marc H Raibert. *Legged Robots That Balance*. MIT press, 1986.

[33] Ludovic Righetti and Stefan Schaal. Quadratic programming for inverse dynamics with optimal distribution of contact forces. In *Proc. IEEE/RAS Int. Conf. Humanoid Robot. (Humanoids)*, pages 538–543, Osaka, Japan, November 2012. doi: 10.1109/HUMANOIDS.2012.6651572.

[34] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. Conf. Artif. Intel. Stat.*, pages 627–635, Ft. Lauderdale, FL, USA, April 2011. doi: 10.48550/arXiv.1011.0686.

[35] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proc. Conf. Robot Learn. (CoRL)*, pages 91–100, London, UK, November 2021. doi: 10.48550/arXiv.2109.11978.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. doi: 10.48550/arXiv.1707.06347.

[37] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. In *Proc. Robot.: Sci. and Syst. (RSS)*, Virtual, July 2021. doi: 10.48550/arXiv.2105.08328.

[38] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proc. Robot.: Sci. and Syst. (RSS)*, pages 1–9, Pittsburgh, Pennsylvania, USA, June 2018. doi: 10.15607/RSS.2018.XIV.010.

[39] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 23–30, Vancouver, BC, Canada, September 2017. doi: 10.1109/IROS.2017.8202133.

[40] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel van de Panne. Learning locomotion skills for Cassie: Iterative design and sim-to-real. In *Proc. Conf. Robot Learn. (CoRL)*, pages 1–

13, Osaka, Japan, November 2020. doi: 10.48550/arXiv. 1903.09537.

[41] Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. ALLSTEPS: Curriculum-driven learning of stepping stone skills. *Computer Graphics Forum*, 39(8):213–224, November 2020. doi: 10.1111/cgf.14115.

[42] Zhaoming Xie, Xingye Da, Michiel van de Panne, Buck Babich, and Animesh Garg. Dynamics randomization revisited: A case study for quadrupedal locomotion. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 4955–4961, Virtual, May 2021. doi: 10.1109/ICRA48506.2021.9560837.

TABLE V: Training Parameters

Hyperparameter	Value
discount factor	0.99
GAE parameter	0.95
# timesteps per rollout	21
# epochs per rollout	5
# minibatches per epoch	4
entropy bonus (α_2)	0.01
value loss coefficient (α_1)	1.0
clip range	0.2
reward normalization	yes
learning rate	1e-3
# workers	1
# environments per worker	4096
# total timesteps	400M
optimizer	Adam

Appendix

A. Training Parameters

The PPO training parameters used for all experiments are provided in Table V.

B. Reward Function

The reward terms are provided in VI.

C. Measures of Agility

Benchmarking the agility of legged robots cannot be accomplished by comparing speed alone due to differences in hardware. [3] proposed to characterize legged agility by the nondimensional Froude Number, defined as $Fr = \frac{v^2}{gl}$ where v is the body velocity, g is gravity, and l is the nominal leg length. This was motivated by the *dynamic similarity hypothesis*, which argues that animals move in a dynamically similar fashion when they have speeds proportional to the square root of their leg lengths [3]. We compile the estimated Froude numbers of quadruped systems contemporary to this work in Table III.

TABLE VI: Reward terms for task, stability, and smoothness. Reward from [35] is adapted to our robot with minor changes.

Term	Symbol	Equation
$r_{v_x^{\text{cmd}}}$: xy velocity tracking	$\exp\{-\lvert\mathbf{v}_{xy} - \mathbf{v}_{xy}^{\text{cmd}}\rvert^2/\sigma_{vxy}\}$	0.02
$r_{\omega_z^{\text{cmd}}}$: yaw velocity tracking	$\exp\{-(\boldsymbol{\omega}_z - \boldsymbol{\omega}_z^{\text{cmd}})^2/\sigma_{\omega z}\}$	0.01
z velocity	$\mathbf{v}_z^2$	-0.04
roll-pitch velocity	$\lvert\boldsymbol{\omega}_{xy}\rvert^2$	-0.001
base height	$(h - h^0)^2$	-0.6
base orientation	$\lvert\mathbf{g}_{xy}^{\text{ori}}\rvert^2$	-0.002
self-collision	$\mathbb{1}_{\text{selfcollision}}$	-0.02
joint limit violation	$\mathbb{1}_{q_i > q_{max} \| q_i < q_{min}}$	-0.2
joint torques	$\lvert\boldsymbol{\tau}\rvert^2$	-2e-7
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$	-5e-9
action rate	$\lvert\mathbf{a}_{t-1} - \mathbf{a}_t\rvert^2$	-2e-4
foot airtime	$\sum t_{air} * \mathbb{1}_{\text{new contact}}$	0.02