Rapid locomotion via reinforcement learning

Gabriel B. Margolis1,*, Ge Yang1,2,*, Kartik Paigwar1, Tao Chen1 and Pulkit Agrawal1,2

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 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. Videos of the robot’s behaviors are available at https://agility.csail.mit.edu/.

Keywords
Robot learning, legged locomotion, sim-to-real reinforcement learning

1. 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 (Bosworth et al., 2016; Fahmi et al., 2020). Some physical considerations only begin to influence the robot’s dynamics at high speeds, including the enforcement of actuator limits (Chignoli et al., 2021; Dai et al., 2014; Herzog et al., 2016), the regulation of large contact forces (Kim et al., 2019), and body control during flight phases (Dai et al., 2014; Kim et al., 2019). 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 (Bledt and Kim 2020; Bosworth et al., 2016; Chignoli et al., 2021; Dai et al., 2014; Ding et al., 2019; Fahmi et al., 2020; Herzog et al., 2016; Kim et al., 2019). 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 (Kumar et al., 2021; Lee et al., 2020; Margolis et al., 2021; Miki et al., 2022; Rudin et al., 2021; Siekmann et al., 2021).

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

1Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA
2NSF AI Institute for Artificial Intelligence and Fundamental Interactions, Cambridge, MA, USA
*Gabriel B. Margolis and Ge Yang are co-first authors

Corresponding author:
Gabriel B. Margolis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA.
Email: gmargo@mit.edu
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-task policy can be successfully learned (Kumar et al., 2021; Lee et al., 2020). 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 (Hessel et al., 2019). 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 (Bengio et al., 2009). Curriculum learning has been leveraged for training robotic systems in the past (Li et al., 2020; Matiisen et al., 2019; Rudin et al., 2021; Xie et al., 2020b; Lee et al., 2020). 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 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 m/s for a 10 m dash (second row of Figure 1). The same policy can spin the robot at 5.7 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.

1.1. Article history

This work was originally presented at the Robotics: Science and System XVIII Conference in New York City, NY, USA, in June 2022. The version you are currently reading includes revised and expanded method and results sections. It is also accompanied by a public code release. Finally, we have extended the discussion section with some reflections on recent work framing the sim-to-real problem as system identification or multi-task learning, indicating some refreshed future directions for study.

2. Experimental setup

2.1. Hardware

We use the MIT Mini Cheetah (Figure 2, Katz et al. (2019)) as our experimental platform. The robot stands 30 cm tall and weighs 9 kg. It is equipped with 12 quasi-direct-drive actuators, each capable of maximum output torque of 17 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 Hz on an onboard NVIDIA Jetson TX2 NX computer.

2.2. Simulation

We use the Isaac Gym simulator (Makoviychuk et al., 2021), and our training code builds on top of the open-
source repository in Rudin et al. (2021). We collect 400 million simulated timesteps in total using 4000 parallel agents for policy training. This is roughly equivalent to 92 days of robot data. This data collection takes under 3 hours of wall-clock time using a single NVIDIA RTX 3090 GPU.

3. Method

Our goal is to learn a policy \( \pi(\cdot) \) with parameters \( \theta \) that takes as input sensory data and velocity commands and gives as output joint position commands (see Figure 3), which are converted into joint torques by a PD controller. The command \( (v_r^{cmd}) \) includes the longitudinal \( (v_z^{cmd}) \) and lateral \( (v_\theta^{cmd}) \) linear velocities and the yaw rate \( (\omega_z^{cmd}) \).

3.1. Control architecture

3.1.1. Observation space. The robot’s sensors provide joint angles \( q_i \in \mathbb{R}^{12} \) and joint velocities \( \dot{q}_i \in \mathbb{R}^{12} \), measured using motor encoders, and \( g_{ori} \in \mathbb{R}^{3} \), which denotes the orientation of the gravity vector in the robot’s body frame and is measured using the IMU. The policy \( \pi(\cdot) \) takes as input a history of previous observations and actions denoted by \( o_{t-H:t} \) where \( o_t = [q_t, \dot{q}_t, g_{ori}^{cmd}, a_{t-1}] \). Because we are learning a command-conditioned policy, the input to the policy is \( x_{t-H:t} \) where \( x_t = o_t \oplus v_r^{cmd} \). During deployment, the body velocity command \( v_r^{cmd} \) is specified by a human operator via remote control.

3.1.2. Action space. The action, \( a_t \in \mathbb{R}^{12} \), 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.

3.1.3. Reward function. In the style of Rudin et al. (2021), we apply task reward terms for linear and angular velocity tracking, as well as a set of auxiliary terms for stability (velocity penalties on body roll, pitch, and height); smoothness (joint torque and acceleration penalties, action change penalty, and footswing duration bonus); and safety (penalty on self-collision and penalty on joint limit violations). We found that the robot tends to sink its body at high speeds and lean into its heading, which can cause unsafe contact between the knees and the ground. This motivated us to introduce penalties on the robot’s body height and orientation. The details of the reward function are in Table 1.

3.2. 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 \( \pi(\cdot) \) by randomizing simulation parameters denoted here as \( d \). We randomize the body mass, the center of mass, motor strength, ground friction, and ground restitution in the ranges reported in Table 2.

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 (Tan et al., 2018; Tobin et al., 2017). Let the resulting policy be \( \pi_{DR}(x) \). While \( \pi_{DR}(x) \) can cross the sim-to-real gap (Tan et al., 2018; Tobin et al., 2017), the learned behavior is conservative (Tan et al., 2018; Xie et al., 2021) 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, \( \pi_{DR}(x) \) has no mechanism for doing so.

To prevent the policy from being conservative, one approach is to include the domain parameters \( d \), as part of the policy input (Chen et al., 2020). The policy \( \pi(x, d) \), 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 \( \pi(x, d) \) with the ability to adapt to different domains. However, this policy cannot be deployed on a real robot since \( d \) cannot be directly measured using onboard sensors. To overcome this limitation, one can deploy a student policy, \( \pi_{teacher}(x_t, x_{t-H:t-1}) \) that is trained to mimic the teacher’s action via behavior cloning (Ross et al., 2011). The main idea is that accurately matching the teacher’s actions forces the
Domain parameters

Table 1. Reward terms for task, stability, and smoothness. We adapt the reward structure from Rudin et al. (2021) to our robot and task.

| Term | Equation | Weight |
|------|----------|--------|
| $r_{xy}$: xy-vel | $\exp\left(-\frac{1}{2} \frac{v_{xy}^2}{\sigma_{xy}^2}\right)$ | 0.02 |
| $r_{yaw}$: yaw-vel | $\exp\left(-\frac{1}{2} \frac{\omega_{z}^2}{\sigma_{\omega_z}^2}\right)$ | 0.01 |
| $z$ vel | $v_z$ | -0.04 |
| roll-pitch vel | $\omega_{xy}^2$ | -0.001 |
| height | $(h-h_{0})^2$ | -0.6 |
| orientation | $\omega_{z}^2$ | -0.002 |
| self collision | $1_{\text{self-collision}}$ | -0.02 |
| Joint limit | $1_{\text{joint limit}}$ | -0.2 |
| joint torques | $\tau^2$ | -2e-7 |
| joint accel | $\dot{q}^2$ | -5e-9 |
| action rate | $|a_{t-1} - a_t|^2$ | -2e-4 |
| foot airtime | $\sum_{t} 1_{\text{contact}}$ | 0.02 |

Table 2. The first set of rows report the ranges of the domain parameters we randomize and identify ($d_i$). The policy is tasked to follow a range of velocity commands that are generated via curriculum strategy described in section 3.3.3.

| Term | Min | Max | Unit |
|------|-----|-----|------|
| Domain parameters | | | |
| 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 | % |
| Command velocity $v_{cmd}$ | var. | var. | m/s |
| Forward velocity Cmd | var. | var. | m/s |
| Lateral velocity Cmd | var. | var. | m/s |
| Angular velocity Cmd | var. | var. | rad/s |

The student to implicitly infer domain parameters ($d_i$) from a state history of $h$ time steps, $x_{[t-h:t-1]}$ 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 $d_i$, instead of learning a single behavior that works across different $d_i$. 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 (Chen et al., 2021), self-driving cars (Chen et al., 2020), and locomotion (Kumar et al., 2021; Lee et al., 2020; Margolis et al., 2021; Miki et al., 2022). Like work applying student–teacher learning to blind walking (Kumar et al., 2021; Lee et al., 2020), our teacher policy observes $d_i$, the dynamic properties of the robot and terrain. The student learns to infer them from $x_{[t-h:t-1]}$, the history of joint angles, and IMU readings.

3.3. Policy optimization

3.3.1. Teacher policy. We construct the teacher policy, $\pi_T(x, d_i)$, as a composition of two modules $g_{\theta_b}$ and $\pi_{\theta_b}$, such that $\pi_T(x, d_i) = \pi_{\theta_b}(x_t, g_{\theta_b}(d_i))$. The first module $g_{\theta_b}$ is the encoder,

$$z_i = g_{\theta_b}(d_i),$$

which compresses $d_i$ into an intermediate latent vector $z_i$. The second module $\pi_{\theta_b}$ is the policy body,

$$a_t = \pi_{\theta_b}(x_t, z_i),$$

which predicts an action from the latent $z_i$ and observation $x_t$. We chose the dimension of $z_i$ as 8 since we found this is the lowest we can set the dimension without reducing performance in our setup. Each module is parameterized as a neural network with ELU activations and architecture described in Table 3. We optimize the teacher’s parameters $\theta_b, \theta_b$ together using Proximal Policy Optimization (PPO) (Schulman et al., 2017) to maximize the future discounted reward,

$$\max_{\theta_b, \theta_b} \mathbb{E}_{x_{[t-h:t-1]}} \sum_{t=0}^{\infty} \gamma^t r_t.$$  \hspace{2cm} (3)

3.3.2. Student policy. The student policy $\pi_S(x_t, x_{[t-h:t-1]}) = \pi_{\theta_b}(x_t, h_{\theta_b}(x_{[t-h:t-1]}))$ imitates the teacher’s behavior during deployment without access to $d_i$. The student policy is constructed by replacing encoder $g_{\theta_b}(d_i)$ with an adaptation module (Kumar et al., 2021),

$$\hat{z}_i = h_{\theta_b}(x_{[t-h:t-1]}),$$

which estimates the latent $\hat{z}_i$ from state history $x_{[t-h:t-1]}$. We train the identification module so that its predictions $\hat{z}_i$ match the encoder’s output $z_i = g_{\theta_b}(d_i)$ as closely as possible. To this end, we optimize parameters $\theta_s$ using supervised learning on on-policy data, using the loss function

$$\mathcal{L}_{\theta_s} = (h_{\theta_b}(x_{[t-h:t-1]}) - g_{\theta_b}(d_i))^2 = (\hat{z}_i - z_i)^2.$$  \hspace{2cm} (5)

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

The optimization procedure follows that of Kumar et al. (2021) 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
Table 3. Network architecture for encoder \(g_{\theta_1}\), adaptation module \(h_{\theta_2}\), and policy body \(\pi_{\theta_3}\). The teacher policy is \(\pi_T(x, d) = \pi_{\theta_1}(x, g_{\theta_2}(d))\), with parameters \(\theta_1, \theta_2\) optimized using PPO. The student policy, \(\pi_S(x, x_{[t-h:t-1]}) = \pi_{\theta_3}(x, h_{\theta_2}(x_{[t-h:t-1]}))\) reuses \(\theta_2\) from the teacher and \(\theta_3\) is optimized using supervised learning.

| Encoder \(g_{\theta_1}\) | | | | | |
|---|---|---|---|---|
| **Inputs** | | | | | |
| | | | | | |
| **Hidden layers** | | | | | |
| [256, 128] | | | | | |
| **Outputs** | | | | | |
| | | | | | |
| **Adaptation module \(h_{\theta_2}\)** | | | | | |
| **Inputs** | | | | | |
| | | | | | |
| **Hidden layers** | | | | | |
| | | | | | |
| **Outputs** | | | | | |
| | | | | | |
| **Policy body \(\pi_{\theta_3}\)** | | | | | |
| **Inputs** | | | | | |
| | | | | | |
| **Hidden layers** | | | | | |
| | | | | | |
| **Outputs** | | | | | |
| | | | | | |

3.3.3. 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 \(v_{x}^{\text{cmd}}\) and \(v_{y}^{\text{cmd}}\) during episode \(k\) are sampled from a probability distribution \(p_{v_{x},v_{y}}^{k}(\cdot, \cdot)\). The lateral velocity command \(v_{z}^{\text{cmd}}\) is sampled separately from a small uniform probability distribution because longitudinal and yaw speed are sufficient for omnidirectional locomotion (Figure 4).

Without a curriculum, there is no change in the sampling procedure from episode to episode:

\[
p_{v_{x},v_{y},v_{z}}^{k+1}(\cdot, \cdot) = p_{v_{x},v_{y},v_{z}}^{k}(\cdot, \cdot)
\]  

(6)

When velocity commands are sampled uniformly from a small range \((-1.0, 1.0)\) at the start of training, the agent can learn to track them (Hwangbo et al., 2019; Kumar et al., 2021). However, when commanded velocities are sampled uniformly from a large distribution \((-4.0, 4.0)\), we found that learning fails (Figure 5).

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 (Bengio et al., 2009). 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 takes the form:

\[
\begin{align*}
p_{v_{x},v_{y}}^{k+1}(\cdot, \cdot) & \leftarrow f(p_{v_{x},v_{y}}^{k}(\cdot, \cdot), k) \\
p_{v_{z}}^{k+1}(\cdot) & \leftarrow f_{v_{z}}(p_{v_{z}}^{k}(\cdot), r_{v_{z}})
\end{align*}
\]  

(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 (Akkaya et al., 2019; Li et al., 2020; Matiisen et al., 2019; Xie et al., 2020b). One possibility is to maintain independent distributions over command dimensions \(p_{v_{x}}(\cdot), p_{v_{y}}(\cdot)\) such that \(p_{v_{x},v_{y}}(\cdot, \cdot) = p_{v_{x}}(\cdot)p_{v_{y}}(\cdot)\) and to specify the update rules \(f_{v_{x}}\) and \(f_{v_{y}}\) for each component separately:

\[
\begin{align*}
p_{v_{x}}^{k+1}(\cdot) & \leftarrow f_{v_{x}}(p_{v_{x}}^{k}(\cdot), r_{v_{x}}) \\
p_{v_{y}}^{k+1}(\cdot) & \leftarrow f_{v_{y}}(p_{v_{y}}^{k}(\cdot), r_{v_{y}})
\end{align*}
\]  

(8a, 8b)

where \(r_{v_{x}}\) and \(r_{v_{y}}\) are the velocity tracking rewards as detailed in Table 1. We refer to this approach as the Box Adaptive curriculum because the probability density function in the \(v_{x},v_{y}\) plane is shaped like a box.

If \(v_{x}\) and \(v_{y}\) 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:

Table 4. Hyperparameters used during training with PPO (Schulman et al., 2017).

| Hyperparameter | Value |
|----------------|-------|
| Discount factor | 0.99 |
| GAE parameter | 0.95 |
| # timesteps per rollout | 21 |
| # epochs per rollout | 5 |
| # minibatches per epoch | 4 |
| Entropy bonus | 0.01 |
| Value loss coefficient | 1.0 |
| Clip range | 0.2 |
| Reward normalization | Yes |
| Learning rate | \(1 \times 10^{-3}\) |
| # workers | 1 |
| # environments per worker | 4096 |
| # total timesteps | 400 M |
| Optimizer | Adam |
\( pk_{v}\omega (\cdot /C_1) \leftarrow f/C_16 \)

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 \( pk_{v}\omega (\cdot /C_1) \) as a uniform probability distribution over \( (v_{cmd},\omega_{cmd} /C_1) \). We represent \( pk_{v}\omega (\cdot /C_1) \) 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, \( \gamma \), with constant value between 0 and 1.

3.3.4. Box Adaptive curriculum update rule. At episode \( k \), the linear and angular velocity commands for the agent are sampled independently:

\[
\begin{align*}
    v_{cmd} &\sim p_k^k(\cdot), \\
    \omega_{cmd} &\sim p_k^k(\cdot).
\end{align*}
\]

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 \( r_{v_{cmd}}, r_{\omega_{cmd}} \) in its attempt to follow \( v_{cmd}, \omega_{cmd} \), then we apply the update rule

\[
\begin{align*}
    p_{k+1}^{v_{cmd}}(v_{cmd}) &= \begin{cases} 1 & r_{v_{cmd}} \geq \gamma, \\ p_k^k(v_{cmd}) & \text{otherwise} \end{cases}, \\
    p_{k+1}^{\omega_{cmd}}(\omega_{cmd}) &= \begin{cases} 1 & r_{\omega_{cmd}} \geq \gamma, \\ p_k^k(\omega_{cmd}) & \text{otherwise} \end{cases}.
\end{align*}
\]

which increases the probability density on neighbors \( v_{cmd} \) of \( v_{cmd} \) and \( \omega_{cmd} \) of \( \omega_{cmd} \) if the reward threshold is met. Here, neighboring commands are defined as the adjacent elements in the (discretized) domain of each marginal distribution: \( v_{cmd} \in [v_{cmd} - 0.5, v_{cmd} + 0.5] \) and \( \omega_{cmd} \in [\omega_{cmd} - 0.5, \omega_{cmd} + 0.5] \). Suppose that \( v_{cmd} \) or \( \omega_{cmd} \) is among the most challenging commands in one of the distributions, and the reward threshold is met. In that case, this update will result in that distribution expanding.

3.3.5. Grid Adaptive curriculum update rule. At episode \( k \), the linear and angular velocity commands for the agent are sampled from the joint distribution:

\[
    (v_{cmd}, \omega_{cmd}) \sim p_k^k(\cdot, \cdot).
\]

\( v_{cmd}, \omega_{cmd} \sim p_k^k(\cdot, \cdot) \)

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 $v^\text{cmd}_x$ and $\omega^\text{cmd}_x$ 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 $[v^\text{cmd}_x, \omega^\text{cmd}_x]$ where the agent received rewards $R_{v^\text{cmd}}, R_{\omega^\text{cmd}}$, we use the following update:

$$p_{x_v, \omega_v}^{k+1}(v^\text{cmd}_x, \omega^\text{cmd}_x) = \begin{cases} p_{x_v, \omega_v}^k(v^\text{cmd}_x, \omega^\text{cmd}_x) & R_{v^\text{cmd}} < \gamma \text{ or } R_{\omega^\text{cmd}} < \gamma, \\ 1 & \text{otherwise.} \end{cases}$$

(13)

This update adds probability density to the neighboring velocity commands $[v^\text{cmd}_x, \omega^\text{cmd}_x]$ of $[v^\text{cmd}_x, \omega^\text{cmd}_x]$, if those commands have not already been added. Here, neighboring commands are defined as neighbors in the 4-connected grid domain of $p_{x_v, \omega_v}^k(\cdot, \cdot)$, which is a discrete grid with resolution $[0.5 \text{ m/s}, 0.5 \text{ rad/s}]$. If $[v^\text{cmd}_x, \omega^\text{cmd}_x]$ 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.

### 3.4. Evaluation metrics

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

$$\epsilon_{ij}[v^\text{cmd}_x] = \mathbb{E}_{v^\text{cmd}_x \sim [i-1,i], \omega^\text{cmd}_x \sim [j-1,j]} \sqrt{\mathbb{E}_t[(v^\text{cmd}_x - v^t_x)^2]},$$

(14a)

$$\epsilon_{ij}[\omega^\text{cmd}_x] = \mathbb{E}_{\omega^\text{cmd}_x \sim [i-1,i], v^\text{cmd}_x \sim [j-1,j]} \sqrt{\mathbb{E}_t[(\omega^\text{cmd}_x - \omega^t_x)^2]},$$

(14b)

where $v^t_x, \omega^t_x$ 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 $\epsilon_0$, we define the command area as the area of the region in the $v^\text{cmd}_x$-$\omega^\text{cmd}_x$ plane for which the tracking errors satisfy

$$\epsilon_{ij}[v^\text{cmd}_x] + \epsilon_{ij}[\omega^\text{cmd}_x] < \epsilon_0$$

(15)

The dimension of the command area is m/s -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 $\epsilon_0$. When we report the command area, we evaluate policies trained with five random speeds over repeated 20 s episodes and indicate their standard deviation using an error bar.

### 3.5. Measures of agility

Benchmarking the agility of legged robots cannot be accomplished by comparing speed alone due to differences in hardware. Alexander (1984) proposed to characterize legged agility by the nondimensional Froude number, defined as $Fr = v^2/gl$ 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 (Alexander 1984). We compile the estimated Froude numbers of quadruped systems contemporary to this work in Table 5.

### 4. Results

Video of hardware experiments may be viewed on the project website: https://agility.csail.mit.edu/.

#### 4.1. Curriculum learning enables high-speed locomotion

Figure 5 visualizes the tracking error (see Section 3.4) of the policies learned from the three command sampling strategies as heatmaps in the $v^\text{cmd}_x$-$\omega^\text{cmd}_x$ plane. The shading on each heatmap corresponds to tracking error, with darker 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 and then gradually increases its capability as the commands become larger. The curriculum plateaued within the training duration of 400M timesteps.

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 forces 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 $[\omega_z \sim 1/v_x]$ when the constraint is active. This phenomenon is in agreement with the apparent inverse shape of the
command area boundary shown in Figure 6, 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.

4.2. Real-world indoor running

To evaluate how fast our robot can run in the real world, we ramped the velocity command linearly from 0.0 m/s to 6.0 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 m/s across multiple speeds (Table 6), with the highest sustained speed of 3.9 m/s among the three speeds. This is higher than the previous record of 3.7 m/s reported for a model-predictive control algorithm on the same robot (Kim et al., 2019).

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 (Alexander 1984) normalizes a robot’s speed by its leg length and has been used to measure agility across robot platforms in the past (Park et al., 2017). Table 5 compares the Froude numbers across different quadrupeds and controllers. Along with the concurrent work of Ji et al. (2022) and Jin et al. (2022), our system is substantially more agile than previous applications of reinforcement learning (RL) achieves highly agile locomotion with Froude number $\geq 1$.

Table 5. Measure of agility: Comparison between the Froude numbers of various prior works. MC: Mini Cheetah; BP: Black Panther; C2: Cheetah 2. Our work (indicated in bold) and concurrent works Ji et al. and Jin et al. have recently provided the initial evidence that reinforcement learning (RL) achieves highly agile locomotion with Froude number $\geq 1$.

| Robot      | RL? | Froude (---) | Speed m/s | Leg L cm | Mass kg | Year |
|------------|-----|-------------|-----------|----------|---------|------|
| Jin et al. | BP  | Y           | 8.5       | 5.0      | 30      | 10   | 2022 |
| Park et al.| C2  | Y           | 7.1       | 6.4      | 59      | 45   | 2017 |
| Ours       | MC  | Y           | 5.1       | 3.9      | 30      | 9    | 2022 |
| Ji et al.  | MC  | Y           | 5.0       | 3.85     | 30      | 9    | 2022 |
| Kim et al. | MC  | Y           | 4.6       | 3.7      | 30      | 9    | 2019 |
| Unitree    | A1  | Y           | 2.8       | 3.3      | 40      | 12   | 2021 |
| Kumar et al.| A1 | Y           | 0.8       | 1.8      | 40      | 12   | 2021 |
| Hwangbo et al. | ANYmal | Y | 0.5 | 1.5 | 50 | 30 | 2019 |

Figure 6. Heatmap of converged tracking error for curricular strategies. 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 $[-6, 6]$ m/s and the y-axis varies the yaw rate between $[-6, 6]$ 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.

While our robot successfully performs rapid locomotion in the real world, there exists a sim-to-real gap as reported in Table 6. The results reveal that online system identification leads to better tracking of the velocity command of 6.0 m/s in simulation (speed of 5.46 m/s with and 5.07 m/s without system identification) and also reduces the sim-to-real gap (average speeds of 3.81 m/s with and 2.49 m/s without system identification). While prior work demonstrated that sim-to-real gap can be mitigated at low velocities (Kumar et al., 2021; Lee et al., 2020), 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 terrain used during the simulated evaluation might not have the same friction parameters as the real terrain, which would result 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. Jin et al. (2022) suggests that the maximum sprinting...
speed on flat ground may be highly sensitive to the lag, which we did not model. The relative contributions of these factors to the sim-to-real performance gap in our system remain uncertain.

4.3. Real-world 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. Despite training only on flat ground, our controller was exposed to domain randomization of the contact dynamics, mass distribution, and motor dynamics that incentivize similar adaptive responses necessary to traverse uneven terrain. 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 s, corresponding to an average speed of 3.4 m/s.

4.4. Real-world 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 rad/s and then 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 (Bledt and Kim 2020). However, the model-based records were achieved using two different controllers for linear (Kim et al., 2019) and angular (Bledt and Kim 2020) 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.

4.5. Response to terrain changes and hardware failures

We tested our system in a diverse set of challenging real-world scenarios: (1) recovering via a change in gait after tripping over a small barrier; (2) ascending a steep incline made of small pebbles; (3) tripping at high speed, flying upside down, and landing on its feet; and (4) maintaining balance despite a mechanical blockage to one motor. We present these qualitative results in the video on the project website, with snapshot images included in Figure 7.

We also deployed the model-predictive controller from Kim et al. (2019) in scenarios (1) and (2), which were the most convenient to replicate. Unlike our learned controller, the baseline did not recover from tripping over the barriers and was slipping down the gravelly incline. While these results highlight the robustness of policies, we want to emphasize that we are not claiming that such (or even more) robustness cannot ever be achieved with model-predictive control. However, the baseline model-predictive controller in this case is a strong and widely adopted method based on common modeling assumptions. 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 more robust behavior in diverse conditions.

4.6. Ablation: 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 (Bosworth et al., 2016; Fahmie et al., 2020) 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 in the high-speed regime. All policies are trained under the same randomization of the domain parameters $\mathbf{d}$, (Table 2).

We find that access to privileged information yields increased performance across all speeds, with the greatest benefit at high speeds. Figure 8 plots the command area for the three policies as the threshold for error increases. The privileged teacher $\pi_T$ trained with access to environment parameters attains a strictly larger command area than the policy $\pi_{DR}$ trained with only the robot state. Using the adaptation module, we show that the student policy $\pi_S$ can nearly match the teacher’s performance. The student’s ability to imitate the teacher is consistent across all speeds.

4.7. Ablation: 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 (Kumar et al., 2021; Lee et al., 2020; Siekmann et al., 2021) 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 9 reports the decrease in performance brought on by introducing terrain roughness when training a high-velocity locomotion policy. Figure 10 visualizes how the command area at a fixed performance threshold varies with terrain noising. Regardless of the training noise level, performance is evaluated on simulated flat ground.

5. Related work

5.1. Model-based control for locomotion

Seminal work in the field (Herdt et al., 2010; Kajita et al., 2003; Raibert, 1986) used simplified models and hand-
specified gaits to make legged robots balance and move dynamically. Subsequent works (Bosworth et al., 2016; Fahmi et al., 2020; Kuindersma et al., 2015; Park et al., 2017; Raibert et al., 2008; Righetti and Schaal, 2012) introduced expanded models with layered control architectures capable of operation on subsets of rough, soft, and slippery terrains.

Recent innovations have addressed specific limitations of simple models with respect to high-speed running. Whole-body control (Dai et al., 2014) 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, Kim et al. (2019) formulated a controller capable of running at speeds up to 3.7 m/s on the Mini Cheetah. Regularized Predictive Control (Bledt and Kim 2020) 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.

5.2. Reinforcement learning for locomotion

Tan et al. (2018) 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, Hwangbo et al. (2019) trained a velocity-tracking controller for the ANYmal robot for speeds up to 1.5 m/s, and Xie et al. (2020a) applied sim-to-real reinforcement learning for agile locomotion on the Cassie biped. Follow-up works expanded ANYmal’s robustness by training on diverse terrains using the teacher–student learning paradigm (Lee et al., 2020; Miki et al., 2022; Rudin et al., 2021). The mechanical design of the ANYmal robot is thought to limit it from running at higher speeds. Fu et al. (2021) and Kumar et al. (2021) 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 m/s, these learning-based works focused on robust traversal of diverse terrains only demonstrated the robot running up to maximum speed 1.8 m/s.

Concurrently with our work, Ji et al. (2022) and Jin et al. (2022) have demonstrated learning-based fast running on the MIT Mini Cheetah and a Mini Cheetah–like robot. Ji et al. (2022) trained agile running policies for the Mini Cheetah robot. They introduced a concurrent state estimation architecture which assists policy optimization by simultaneously learning a state estimator using supervised learning. Jin et al. (2022) achieved the highest recorded forward speed of 5.0 m/s using a Mini Cheetah–like robot. They accomplished this by introducing imitation-relaxation,
a reference trajectory-based reward shaping technique, and by reducing the overall system latency.

Among Ji et al. (2022), Jin et al. (2022), and our work, each has different relative strengths and weaknesses. First, Ji et al. (2022) and Jin et al. (2022) are focused exclusively on the task of maximizing forward sprinting speed. Ji et al. (2022) used a fixed-schedule curriculum on forward linear velocity only. Jin et al. (2022) designed shaped reference trajectories for different forward running speeds, but not for high-speed turning or backward running. Due to our novel adaptive curriculum strategy, we are able to learn high-speed cornering, spinning, and backward locomotion while retaining comparable forward speed to these works. A second difference among these works lies in their method for dealing with complex terrains. Our work applies a teacher–student architecture based on implicit system identification (Kumar et al., 2021) to improve performance across diverse randomized terrains. In agreement with our results, Ji et al. (2022) concluded that online system identification improves sim-to-real transfer in the high-speed locomotion setting; but instead of learning as we did to implicitly adapt to different environments, Ji et al. (2022) learned to explicitly estimate robot state components such as body velocity and contact probability. While both our work and Ji et al. (2022) used the observation history as input to a feedforward neural network to infer the system state at each timestep, Jin et al. (2022) differs by relying on the hidden state of an LSTM to capture the relevant information from past observations. This smaller architecture allows faster inference and reduces latency, enabling faster top-speed running. Jin et al. (2022) also provides a full analysis of the impact of latency on top robot speed which may be of interest to readers.

5.3. 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 (Bengio et al., 2009). Akkaya et al. (2019) 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 the locomotion context; Lee et al. (2020); Miki et al. (2022); and Rudin et al. (2021) applied a curriculum on terrains to learn highly robust walking controllers on non-flat ground. Xie et al. (2020b) notably evaluated terrain curriculum strategies, including adaptive curricula, in the setting of stepping stone traversal with a physically simulated biped.

5.4. Teacher–student learning

Learning with a privileged teacher has been leveraged for robotics in a number of previous works. Kumar et al. (2021) and Lee et al. (2020) applied this approach to the task of blind walking. The teacher policy observed , the dynamic properties of the robot and terrain, and the student learned to infer them from , the history of joint angles and IMU readings. Margolis et al. (2021) and Miki et al. (2022) used the same approach to incorporate terrain geometric information into a locomotion policy. In these works, was a ground-truth geometric heightmap of the terrain. In Miki et al. (2022), the student policy observed a noisy heightmap. In Margolis et al. (2021), the student policy observed a forward-facing depth image. Chen et al. (2021) applied the teacher–student training approach to the task of object reorientation using a dexterous five-fingered hand. In this work, 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.

6. Discussion

6.1. Contributions

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 (Katz et al., 2019) with commercially available analogues (Unitree 2022). Therefore, our method can be readily tested and built upon by others using relatively accessible materials.

6.2. Code release

We have published our training code, including implementation of the teacher–student training and adaptive curriculum strategy, at https://github.com/Improbable-AI/rapid-locomotion-rl.

6.3. Limitations

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.

6.4. Perspective on the field and future directions

Recently, a plethora of methods have been proposed for dealing with uncertainty in robot and environment dynamics during sim-to-real transfer of learned control policies (Tan et al., 2018; Hwangbo et al., 2019; Kumar et al., 2021; Ji et al., 2022). One view espoused in early works was that the utility of domain randomization is in mitigating misalignment between the simulated and real system resulting from approximations made in the simulator. Naive domain randomization carries a problem: it makes the agent conservative. One remedy is to perform better system identification to reduce the amount of domain randomization required. In this spirit, several works identified key sources of model mismatch in simulators and showed that by resolving these, the sim-to-real gap may be functionally diminished. For example, Hwangbo et al. (2019) trained an actuator network on real data to accurately simulate the motor dynamics, while Xie et al. (2021) characterized and modeled the system lag, and showed that this was sufficient for good locomotion performance on flat ground. However, this was not the end of the story. Improving the simulator accuracy benefits performance, but this alone does not address the multistate scenario, in which the robot needs to adapt to different terrains. In this case, domain randomization and history-informed estimation remain necessary. Domain randomization can result in a conservative policy, but the history-informed estimation can alleviate this problem by allowing better identification of the current domain using online data. It is prudent to view domain randomization as simply defining a multistate problem—where every randomly selected domain corresponds to a task we might encounter in the real world—and to view estimation as providing an optimization tool for resolving ambiguity in the task.

Several approaches to history-based estimation or online system identification have been explored in prior literature (Yu et al., 2017; Chen et al., 2020). One possibility is to use a recurrent neural network and rely on the optimizer to learn an effective memory representation (Jin et al., 2022). Another possibility is to learn a space of latents or true physics parameters as a direct function of the state history (Lee et al., 2020; Kumar et al., 2021; Hwangbo et al., 2019; Ji et al., 2022). In our work, we rely on a teacher–student method with an emergent latent space. Our experiments show that the importance of integrating historical information is emphasized in the instance of maximizing agility.

In this light, the path via sim-to-real learning toward expanding the motor skills of robots appears to involve pairing expansion of the task distribution together with algorithmic and architectural improvements that facilitate optimization across many tasks. Recently, some works have achieved this by expanding the locomotion task to include new dimensions of variation. Most simulators used for sim-to-real locomotion use a rigid body model, and therefore, deformable terrains like sand or mattresses are outside the training distribution. Although we can hope that a policy trained with other randomizations can generalize to these terrains, its performance is likely to be suboptimal. To address this, Choi et al. (2023) augmented a simulator with a deformable terrain model and trained a quadruped robot to better adapt to the dynamics of soft and granular surfaces. Other work has expanded the locomotion task via manipulation of the reward function rather than manipulation of the simulated environments. Margolis and Agrawal (2022) optimized a multi-task policy to realize many structured quadrupedal gaits on flat surfaces using an additional gait reward. This work showed that while a gait-free policy can achieve faster speeds on flat ground, some structured gaits can generalize better to out-of-distribution tasks such as new terrains or energy efficiency criteria.

In addition to broadening the set of target terrains, the locomotion task can be expanded to include the control of additional limbs or objects in the environment. For example, dribbling a soccer ball requires a legged robot to run while simultaneously applying forces with the feet to control the soccer ball, a body with its own distinct dynamics. Ji et al. (2023) modeled the variable drag dynamics of interaction between a soccer ball and different surfaces, which enabled an adaptive policy to dribble the ball across pavement, grass, and mud. Another task beyond locomotion is to control a quadruped with a mounted arm, which incurs a change in robot dynamics as well as additional actuated degrees of freedom. Fu et al. (2022) mounted an arm on a small quadruped and provided reward based on the end effector position target, yielding an expanded task space jointly encompassing locomotion and manipulation behaviors, and accomplished control using the same sim-to-real paradigm.

Another emerging frontier for highly multistate locomotion lies with the incorporation of additional sensors such as cameras. Such sensors can decrease the burden of estimation and make new strategies possible by directly providing additional information about the current task to the robot. Margolis et al. (2021) found success in applying teacher–student methods to learn visual locomotion across a
restricted set of terrains where foot placement is critical. Miki et al. (2022) and Agarwal et al. (2022) accomplished extension of this approach to a more complex catalogue of terrains representative of those found in natural environments. These works relied on the availability of procedural terrain generation techniques to define the distribution of training tasks. As practitioners seek to expand the capabilities of these robots, how to design the environment distribution remains a critical question. How, for example, can we procedurally generate realistic and feasible training environments for legged mobile manipulation? Capturing the real-world distribution of movable objects and their visual appearances is nontrivial. Curriculum learning is another critical aide used in these works as well as our own. But implementing a curriculum on an arbitrary complex set of environments is another challenge: How can we sort environments from easiest to hardest? How can we prune infeasible or trivial environments?

In conclusion, it remains unclear which specific design choices from sim-to-real research today will transfer well to more complex settings in the future. But our work on rapid locomotion has offered some evidence for the general importance of (a) automatically controlling the feasibility of the task distribution and (b) combining dynamics randomization with online system identification. Taking into account the broad course of legged locomotion research in recent years, these seem to be principles worth our continued attention. Continued evaluation of design choices across diverse control tasks will improve our understanding of how these considerations interact with problem structure.

Acknowledgements

We thank the members of the Improbable AI Lab at MIT for providing valuable feedback on the project direction and the manuscript. We also thank the members of the Biomimetic Robotics Laboratory at MIT for helpful discussions and hardware support for the Mini Cheetah robot. Finally, we are grateful to MIT SuperCloud and the Lincoln Laboratory Supercomputing Center for providing HPC resources.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The Mini Cheetah robot used in this work was donated by the MIT Biomimetic Robotics Laboratory and NAVER. 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, https://iaif.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. This work was supported by Defense Advanced Research Projects Agency under HR00111950005. 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.

ORCID iDs

Gabriel B. Margolis https://orcid.org/0000-0001-8191-4813
Pulkit Agrawal https://orcid.org/0000-0001-8463-9917

References

Agarwal A, Kumar A, Malik J, et al. (2022) Legged Locomotion in Challenging Terrains Using Egocentric Vision. In: Proceedings of Conference on Robot Learning (CoRL). Auckland, New Zealand, 403–415. arXiv preprint arXiv: 2211.07638.

Akkaya I, Andrychowicz M, Chociej M, et al. (2019) Solving Rubik’s Cube With a Robot Hand. arXiv preprint arXiv: 1910.07113. arXiv preprint DOI: 10.48550/arXiv.1910.07113.

Alexander RM (1984) The gaits of bipedal and quadrupedal animals. The International Journal of Robotics Research 3(2): 49–59. DOI: 10.1177/027836498400300205.

Bengio Y, Louradour J, Collobert R, et al. (2009) Curriculum learning. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, Montreal, Canada, 41–48. DOI: 10.1145/1553374.1553380.

Bledt G and Kim S (2020) Extracting legged locomotion heuristics with regularized predictive control. In: Proceedings of International Conference on Robotics and Automation. Virtual, 406–412. DOI: 10.1109/ICRA40945.2020.9197488.

Bosworth W, Whitney J, Kim S, et al. (2016) Robot locomotion on hard and soft ground: measuring stability and ground properties in-situ. In: Proceedings of International Conference on Robotics and Automation, Stockholm, Sweden, 3582–3589. DOI: 10.1109/ICRA.2016.7487541.

Chen D, Zhou B, Koltun V, et al. (2020) Learning by Cheating. In: Proceedings of Conference on Robot Learning (CoRL), Virtual, 66–75. DOI: 10.48550/arXiv.1912.12294.

Chen T, Xu J and Agrawal P (2021) A system for general in-hand object re-orientation. In: Proceedings of Conference on Robot Learning (CoRL), London, UK, 297–307. DOI: 10.48550/arXiv.2111.03043.

Chignoli M, Kim D, Stanger-Jones E, et al. (2021) The MIT humanoid robot: design, motion planning, and control for acrobatic behaviors. In: Proceedings under IEEE-RAS International Conference on Humanoid Robots. Munich, Germany, 1–8. DOI: 10.1109/HUMANOIDS47582.2021.9555782.

Choi S, Ji G, Park J, et al. (2023) Learning quadrupedal locomotion on deformable terrain. Science Robotics 8(74): eade2256.
