Science Robotics
AAAS

robotics.sciencemag.org/cgi/content/full/5/47/eabc5986/DC1

Supplementary Materials for

Learning quadrupedal locomotion over challenging terrain

Joonho Lee*, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, Marco Hutter

*Corresponding author. Email: jolee@ethz.ch

Published 21 October 2020, *Sci. Robot.* **5**, eabc5986 (2020)
DOI: 10.1126/scirobotics.abc5986

The PDF file includes:

Other Supplementary Material for this manuscript includes the following:

(available at robotics.sciencemag.org/cgi/content/full/5/47/eabc5986/DC1)

Movie S3 (.mp4 format). Step experiment.
Movie S4 (.mp4 format). Payload experiment.
Movie S5 (.mp4 format). Foot slippage experiment.

Supplementary materials

S1. Nomenclature

$\hat{(\cdot)}$	normalized vector
$\dot{(\cdot)}$	first derivative
$\bar{(\cdot)}$	teacher's quantity
$(\cdot)_T$	target quantity
${}^{C}_{AB}v$	linear velocity of B frame with respect to A frame expressed in C frame
c_T	terrain parameter vector
ω	angular velocity
τ	joint torque
θ	joint angle
ψ	yaw angle
ϕ	leg phase
f	leg frequency
r_f	linear position of a foot
e_g	gravity vector
H	horizontal frame
g_i	gap function of the i-th possible contact pair
I_c	index set of all contacts
$I_{c,body}$	index set of body contacts
$I_{c,foot}$	index set of foot contacts
I_{swing}	index set of swing legs
$\lvert\cdot\rvert$	cardinality of a set or l_1 norm
$\lvert\lvert\cdot\rvert\rvert$	l_2 norm

S2. Implementation details

The RaiSim simulator (*30*) is used for rigid-body and contact dynamics simulation. The actuator networks (*12*) are trained for each robot to simulate Series Elastic Actuators (SEA) (*47*) at the joint. The input to the actuator model is a 6-dimensional real-valued vector consisting of the joint position error and velocity at current time step t and two past states corresponding to $t - 0.01\,\mathrm{s}$ and $t - 0.02\,\mathrm{s}$. The feature selection is done as in (*12*).

As several studies have shown that randomization of dynamic properties improves the robustness of the policy (*12, 16*), we also randomized several physical quantities, and the teacher policy has access to these values during training. We applied disturbances, randomized friction coefficients between the feet and the terrain, and additive noise to the observations during training.

The training process for the teacher policy is depicted in Algorithm S1. Hyperparameters are given in Table S3. In our implementation of the terrain curriculum, we update the curriculum every $N_{evaluate}$ policy iterations to reduce variance. We assume that within $N_{evaluate}$ iterations, the performance of the policy is similar. With the slower update rate, the measurement probability of Eq.6 becomes

$$\Pr(y_j^k | c_{T,j}^k) \approx \sum^{N_{evaluate}} \sum^{N_{traj}} \frac{\mathbb{1}(Tr(c_{T,j}^k, \pi) \in [0.5, 0.9])}{N_{traj} N_{evaluate}}. \tag{9}$$

Additionally, we leverage replay memory to prevent degeneration of the particle filter and to avoid catastrophic forgetting.

The controller is implemented with a state machine to switch between the "standing still" state and the locomotion state. We set the base frequency f_0 to zero when the zero command is given for 0.5 s, which stops FTGs, and the robot stands still on the terrain. f_0 is set to 1.25 Hz

when the direction command is given or the linear velocity of the base exceeds 0.3 m/s for the disturbance rejection. The state machine is included in the training environment.

During the deployment, the base velocity and orientation are estimated by the state estimator that relies on inertial measurements and leg kinematics (*37*).

The neural network policy runs at 400 Hz on an onboard CPU (Intel i7-5600U, 2.6 – 3.2GHz, dual-core 64-bit) integrated into the robot. The Tensorflow C++ API is used for on-board inference.

S3. Foot trajectory generator

The foot trajectory is defined as

$$F(\phi_i) = \begin{cases} (h(-2k^3 + 3k^2) - 0.5)^{H_i} z & k \in [0, 1] \\ (h(2k^3 - 9k^2 + 12k - 4) - 0.5)^{H_i} z & k \in [1, 2] \\ -0.5^{H_i} z & \text{otherwise,} \end{cases} \tag{10}$$

where $k = 2(\phi_i - \pi)/\pi$ and h is a parameter for the maximum foot height. Each segment during the swing phase ($k \in [0, 2)$) is a cubic Hermite spline connecting the highest and lowest points with a zero first derivative at the connecting points. Other periodic functions such as $h_i \sin(\phi_i)$ can be used for the FTG. With a set of reasonably tuned f_0, h and $\phi_{i,0}$, a quadruped can stably step in place. In our setting, f_0 = 1.25, h = 0.2 m, and $\phi_{i,0}$ are sampled from $U(0, 2\pi)$.

S4. Reward function for teacher policy training

The reward function is defined as $0.05r_{lv} + 0.05r_{av} + 0.04r_b + 0.01r_{fc} + 0.02r_{bc} + 0.025r_s + 2 \cdot 10^{-5} r_\tau$. The individual terms are defined as follows.

- Linear Velocity Reward (r_{lv}): This term maximizes the $v_{pr} = ({}^B_{IB}v)_{xy} \cdot ({}^B_{IB}\hat{v}_T)_{xy}$, which is the base linear velocity projected onto the command direction.

$$r_{lv} \coloneqq \begin{cases} \exp\left(-2.0(v_{pr} - 0.6)^2\right) & v_{pr} < 0.6 \\ 1.0 & v_{pr} \geq 0.6 \\ 0.0 & \text{zero command} \end{cases}. \tag{11}$$

The velocity threshold is defined as 0.6 m/s which is the maximum speed reachable on the flat terrain with the existing controller (*27*).

- Angular Velocity Reward (r_{av}): We motivate the agent to turn as fast as possible along the base z-axis when $({}^{B}_{IB}\hat{\omega}_T)_z$ is nonzero. It is defined as

$$r_{av} \coloneqq \begin{cases} \exp\left(-1.5(\omega_{pr} - 0.6)^2\right) & \omega_{pr} < 0.6 \\ 1.0 & \omega_{pr} \geq 0.6 \end{cases}, \tag{12}$$

  where $\omega_{pr} = ({}^{B}_{IB}\omega)_z \cdot ({}^{B}_{IB}\hat{\omega}_T)_z$.

- Base Motion Reward (r_b): This term penalizes the velocity orthogonal to the target direction and the roll and pitch rates such that the base is stable during the locomotion.

$$r_b \coloneqq \exp(-1.5v_o^2) + \exp(-1.5||({}^{B}_{IB}\omega)_{xy}||^2) \tag{13}$$

  where $v_o = ||({}^{B}_{IB}v)_{xy} - v_{pr} \cdot ({}^{B}_{IB}\hat{v}_T)_{xy}||$. When the stop command is given, v_o is replaced by $||{}^{B}_{IB}v||$.

- Foot Clearance Reward (r_{fc}): When a leg is in swing phase, i.e., $\phi_i \in [\pi, 2\pi)$, the robot should lift the corresponding foot higher than the surroundings to avoid collision. We first define the set of such collision-free feet as $\mathcal{F}_{clear} = \{i : r_{f,i} > max(H_{scan,i}), i \in I_{swing}\}$, where $H_{scan,i}$ is the set of scanned heights around the i-th foot. Then the clearance cost is defined as

$$r_{fc} \coloneqq \sum_{i \in I_{swing}} (\mathbb{1}_{\mathcal{F}_{clear}}(i)/|I_{swing}|) \in [0.0, 1.0]. \tag{14}$$

- Body Collision Reward (r_{bc}): We want to penalize undesirable contact between the robot's body parts and the terrain to avoid hardware damage.

$$r_{bc} \coloneqq -|I_{c,body} \backslash I_{c,foot}|. \tag{15}$$

- Target Smoothness Reward (r_s): The magnitude of the second order finite difference derivatives of the target foot positions are penalized such that the generated foot trajectories become smoother.

$$r_s \coloneqq -||(r_{f,d})_t - 2(r_{f,d})_{t-1} + (r_{f,d})_{t-2}||. \tag{16}$$

- Torque Reward (r_τ): We penalize the joint torques to prevent damaging joint actuators during the deployment and to reduce energy consumption ($\tau \propto$ electric current).

$$r_\tau \coloneqq -\sum_{i \in joints} |\tau_i|. \tag{17}$$

S5. Parameterized terrains

It is important to generate training environments that can pose representative challenges such as foot slippage and foot-trapping. To efficiently synthesize random terrains, we use procedural generation techniques (*48*). This method allows us to generate a large number of different terrains by changing a set of terrain parameters $c_T \in \mathcal{C}$. In the following, we describe the three terrain generators used in this work. See Fig. 4B for a visualization of the terrains and Table S2 for the definition of parameter spaces $\mathcal{C}$.

- The *Hills* terrain is based on Perlin noise (*49*). The terrain is generated via three parameters: roughness, frequency of the Perlin noise, and amplitude of the Perlin noise. The height of each element of the output height map hm is defined as $hm[i,j] \coloneqq Perlin(c_{T,2}, c_{T,3})[i,j] + U(-c_{T,1}, c_{T,1})$. A policy experiences smooth slopes and foot slippage on this terrain during training.

- The *Steps* terrain consists of square steps of random height. For every $c_{T,1}$ by $c_{T,1}$ blocks, the height is sampled from $U(0, c_{T,2})$. A policy experiences discrete elevation changes and foot-trapping on this terrain.

- The *Stairs* terrain is a staircase with fixed width and height. The robot is initialized at the flat segment in the middle of the staircase (see Fig. 4B).

The ranges are defined considering the kinematics of the robot, e.g., a step height should be lower than leg length. During training, the terrain is regenerated every episode with a different random seed.

S6. Qualitative evaluation of the adaptive terrain curriculum

The behavior of adaptive curriculum is illustrated in Fig. S1. Fig. S1A-C focuses on the *Hills* terrain type. There are three parameters for this terrain: roughness, frequency, and amplitude. The relationship between traversability (Eq. 3) and desirability (Eq. 4) is illustrated in Fig. S1A-B. Undesirable terrains are either too easy or too difficult, as shown in the leftmost and rightmost panels of Fig. S1A. Fig. S1B-C shows that the particle filter fits the latent distribution of desirable terrains, which has a bow shape in the frequency-amplitude marginal (middle). Fig. S1D focuses on the *Stairs* terrain type and shows the evolution of terrain parameters during training. The particle filter rejects parameters that represent short and steep steps (upper-left area). The curriculum initially focuses on wide and shallow steps (middle panels, particularly Iter. 50-60), and then broadens the distribution to include narrower steps (rightmost panels).

S7. Reconstruction of the privileged information in different situations

In Fig. S2, we provide the decoded privileged information in different situations. Fig. S2A shows the estimated friction coefficient between the feet and the terrain when traversing a wet, slippery whiteboard, as shown in Movie S5. The estimate decreases as soon as the first foot starts slipping (i), remains low throughout the traversal (ii) and increases about 2 s after the robot returns to normal ground (iii). The external disturbance and terrain information can also be reconstructed from the TCN embedding. As shown in Fig. S2B, the decoder detects downward

external force when an unknown 10 kg payload is applied. While traversing dense vegetation as shown in Fig. S2C, it detects a force opposite the motion direction, which makes the policy to counteract and push through the vegetation. The uncertainty of the elevation estimates are notably high in the natural terrains shown in Fig. S2C and Fig. S2D, which indicates that the TCN policy encodes the roughness of the terrain.

S8. Recurrent neural network student policy

We use the TCN architecture for the proprioceptive policy (*22*). For comparison, we also evaluated a recurrent network with gated recurrent units (GRU) (*50*). The architectures are specified in Tables S5 and S6. The loss function for training a GRU student policy is defined as

$$\mathcal{L} \coloneqq (\bar{a_t}(o_t, x_t) - a_t(o_t))^2 + (\bar{l_t}(o_t, x_t) - l_t(o_t))^2. \tag{18}$$

To improve the performance and computational efficiency of the training, we have implemented Truncated Backpropagation Though Time (Truncated BPTT) (*51*).

Performance on the diagnostic settings presented in Fig. 5A is given in Fig. S3. Overall, the performance of the GRU-based controller is between that of TCN-20 and TCN-100. The performance is comparable to TCN-100 in the slope setting, but the GRU-based controller fails to achieve the performance of TCN-100 in step experiments.

The chief advantage of the TCN is in training efficiency. The training time for the TCN is much faster in comparison to the GRU. The computation times are reported in Table S6.

S9. Ablation of the latent representation loss for student training

We examine the effect of the second term in the loss function for student policy training presented in Eq. 1, which is a squared error loss for the latent vector l_t. As a baseline, we train a student policy using the following loss function:

$$\mathcal{L} \coloneqq (\bar{a_t}(o_t, x_t) - a_t(o_t, H))^2, \tag{19}$$

which simply imitates the output of the teacher.

The result is reported in Fig. S3 as ‘TCN-100 naive IL’. The performance is comparable in the uniform slope setting and under external disturbances. On the other hand, the ablated version has lower success rates on steps.

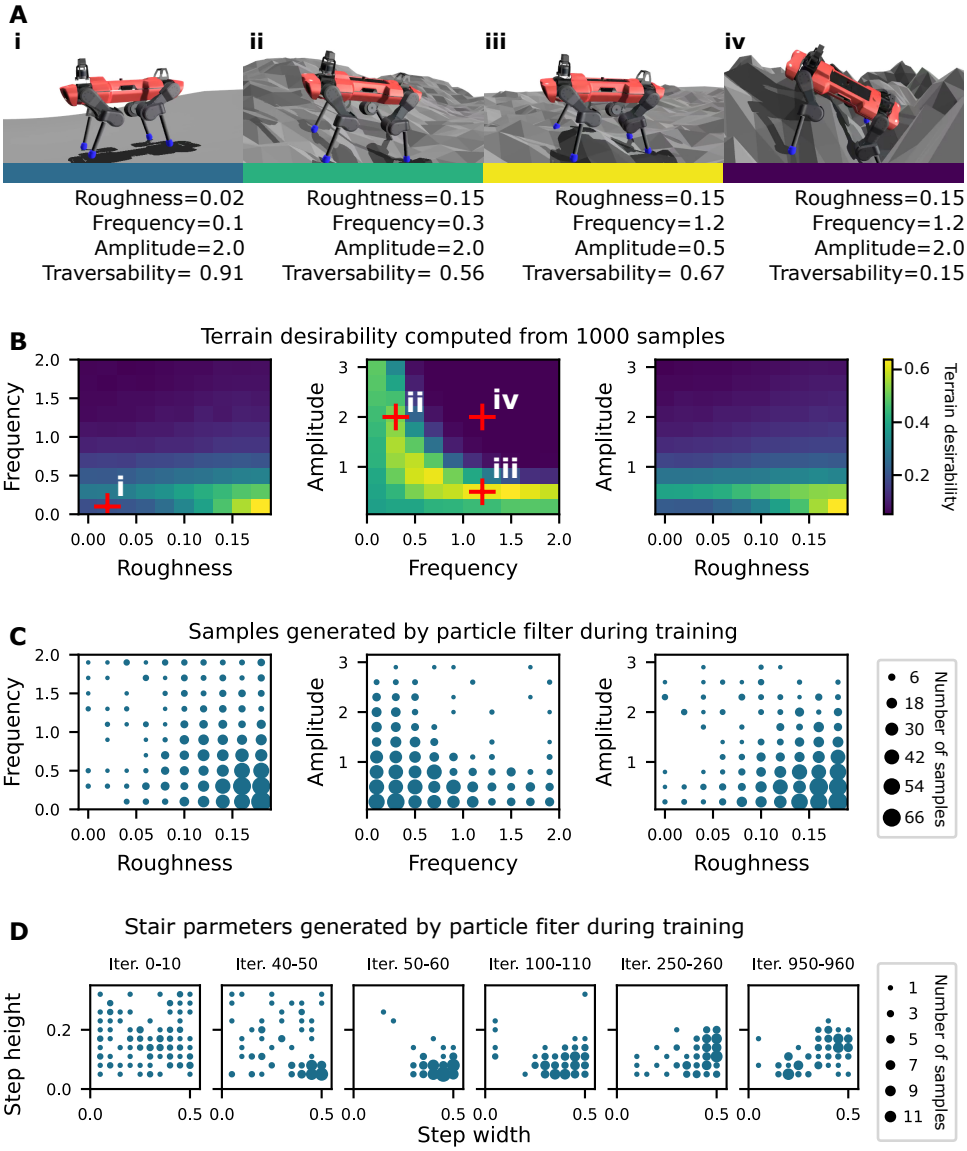

Figure S1: **Illustration of the adaptive curriculum.** (**A**) Examples of *Hills* terrains. The color bar indicates desirability; dark blue represents low desirability. (**B**) Terrain desirability estimated from 1000 trajectories generated by a fully trained teacher policy. The red crosses correspond to the examples presented in A. (**C**) The distribution of terrain profiles sampled by the particle filter during the last 100 iterations of teacher training. (**D**) Evolution of *Stairs* terrain parameters during training.

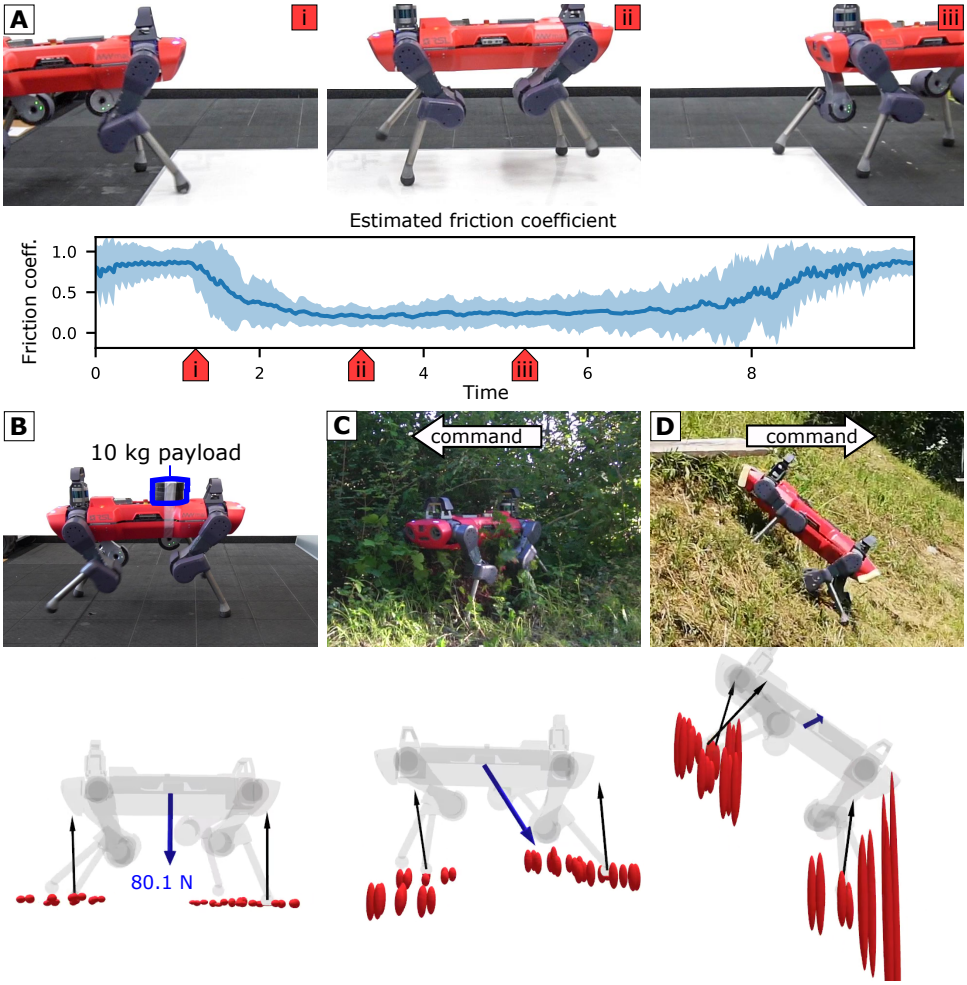

Figure S2: **Reconstructed privileged information in different situations.** (**A**) Estimated friction coefficient between the feet and the terrain while traversing a wet whiteboard. The shaded area denotes 95 % confidence interval. (**B-D**) Reconstruction of the external disturbance and terrain information in different scenarios. Blue arrow: estimated external force applied to the torso. Red ellipsoid: estimated terrain shape around the foot. The center of the ellipsoid refers to the estimated terrain elevation and the vertical length represents uncertainty (1 standard deviation). For each foot, 8 ellipsoids are symmetrically placed along a circle with 10 cm radius. Black arrow: terrain normal at the in-contact foot.

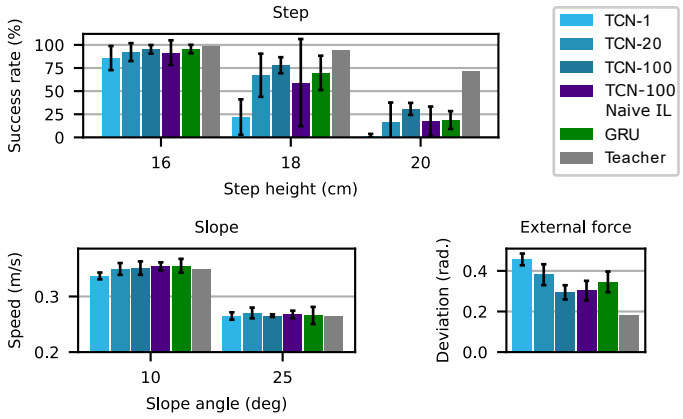

Figure S3: **Comparison of neural network architectures for the proprioceptive controller.** We trained each model 5 times using different random seeds. The error bars denote 95% confidence intervals. 'TCN-100 naive IL' denotes the TCN-100 network trained using a naive imitation learning method without the latent representation loss (Eq. 19).

Name	Time
Teacher policy training	$\approx$ 12 hrs
Student policy training	$\approx$ 4 hrs
Adaptive terrain curriculum	2.9 s

Table S1: **Computation time for training**. The TCN-100 architecture is used for the student policy. The training is conducted on a desktop machine with i7-8700K CPU and a Geforce RTX 2080 GPU.

Terrain	grid size	friction coefficient	parameters (c_T)	range
Hills	0.2 m	$\mathcal{N}(0.7, 0.2)$	roughness (m)	$[0.0, 0.05]$
			frequency	$[0.2, 1.0]$
			amplitude (m)	$[0.2, 3.0]$
Slippery Hills	0.2 m	$\mathcal{N}(0.3, 0.1)$	roughness (m)	$[0.0, 0.05]$
			frequency	$[0.2, 1.0]$
			amplitude (m)	$[0.2, 3.0]$
Steps	0.02 m	$\mathcal{N}(0.7, 0.2)$	step width (m)	$[0.1, 0.5]$
			step height (m)	$[0.05, 0.3]$
Stairs	0.02 m	$\mathcal{N}(0.7, 0.2)$	step width (m)	$[0.1, 0.5]$
			step height (m)	$[0.02, 0.2]$

Table S2: **Parameter spaces $\mathcal{C}$ for simulated terrains.** $\mathcal{N}(m, d)$ denotes that the value is sampled from the Gaussian distribution of mean m and stardard deviation d. The friction coefficient is clipped to be above 0.1.

Parameter	value
Number of particles ($N_{particle}$)	10 per terrain type
Transition probability ($p_{transition}$)	0.8
Trajectories per particle (N_{traj})	6
Update rate of the terrain parameters ($N_{evaluate}$)	10
Probability of sampling from replay memory (P_{replay})	0.05

Table S3: **Hyperparameters for automatic terrain curriculum.**

Data	dimension	x_t	o_t	h_t
Desired direction ($({}^{B}_{IB}\hat{v}_d)_{xy}$)	2		✓	✓
Desired turning direction ($({}^{B}_{IB}\hat{\omega}_d)_z$)	1		✓	✓
Gravity vector (e_g)	3		✓	✓
Base angular velocity (${}^{B}_{IB}\omega$)	3		✓	✓
Base linear velocity (${}^{B}_{IB}v$)	3		✓	✓
Joint position/velocity (θ_i, $\dot{\theta}_i$)	24		✓	✓
FTG phases ($\sin(\phi_i)$, $\cos(\phi_i)$)	8		✓	✓
FTG frequencies ($\dot{\phi}_i$)	4		✓	✓
Base frequency (f_o)	1		✓	
Joint position error history	24		✓	
Joint velocity history	24		✓	
Foot target history ($(r_{f,d})_{t-1,t-2}$)	24		✓	
Terrain normal at each foot	12	✓		
Height scan around each foot	36	✓		
Foot contact forces	4	✓		
Foot contact states	4	✓		
Thigh contact states	4	✓		
Shank contact states	4	✓		
Foot-ground friction coefficients	4	✓		
External force applied to the base	3	✓		

Table S4: **State representation for proprioceptive controller (top) and the privileged information (bottom).**

Layer	Teacher		TCN-N Student		GRU Student		Decoder
input	o_t	x_t	o_t	h (60×N)	o_t	o_t	$\langle o_t, l_t \rangle$
1	id	tanh(72)	id	1D conv dilation 1	id	GRU(68)	relu(196)
2	id	tanh(64)	id	1D conv stride 2	concatenate		Output
3	concatenate		id	1D conv dilation 2	tanh(256)*		-
4	tanh(256)*		id	1D conv stride 2	tanh(128)*		-
5	tanh(128)*		id	1D conv dilation 4	tanh(64)*		-
6	tanh(64)*		id	1D conv stride 2	Output*		-
7	Output*		id	tanh(64)	-		-
8	-		concatenate		-		-
9	-		tanh(256)*		-		-
10	-		tanh(128)*		-		-
11	-		tanh(64)*		-		-
12	-		Output*		-		-

Table S5: **Neural network architectures.** Unless specified otherwise, the dilation and stride are 1 for convolutional layers. The filter size is fixed to 5. The layers marked with $*$ are copied from the teacher to learners after the teacher training. id refers to the identity map. The TCN-N architecture uses dilated causal convolution (*22*). Each convolutional layer is followed by a relu activation function.

Model	seq. length	# channels	# param.	SGD time (s)
TCN-1	1	60	161960	9.22e-3 (±1.78e-3)
TCN-20	20	44	158300	2.11e-2 (±1.24e-3)
TCN-100	100	34	158070	5.07e-2 (±1.94e-3)
GRU	100*	-	159640	1.52e-1 (±1.89e-2)

Table S6: **Network parameter settings and training time for student policies.** SGD time refers to the computation time required for one stochastic gradient descent update with the batch size given in Table S8. The computation times are presented as empirical means with standard deviations. *The sequence length for the GRU network stands for the sequence length used for Truncated BPTT (*51*).

Parameter	Value
discount factor	0.995
KL-d threshold	0.01
max. episode length	400
CG damping	1e-1
CG iteration	50
discount factor	0.995
batch size	80000
total iterations	10000

Table S7: **Hyperparameters for teacher policy training.**

Parameter	TCN-N	GRU
initial learning rate	5e-4	2e-4
learning rate decay	exp(0.995, 100)	
max. episode length	400	
batch size	20000	10000
minibatches	5	
epochs	4	
total iteration	4000	

Table S8: **Hyperparameters for student policy training.** exp(a,b) denotes exponential decay, which is defined as $lr_0 * a^{updates/b}$. The Adam (*52*) optimizer is used.

Parameter	values
initial learning rate	1e-4
learning rate decay	exp(0.99, 100)
batch size	20000
minibatches	2
epochs	10
total iteration	1000
weight decay	$l2$-norm, 1e-4

Table S9: **Hyperparameters for decoder training.** exp(a,b) denotes exponential decay, which is defined as $lr_0 * a^{updates/b}$. The Adam (*52*) optimizer is used.

Algorithm S1 Teacher training with automatic terrain curriculum

Initialize a replay memory, Sample $N_{particle}$ $c_{T,0}$s uniformly from $\mathcal{C}$ (Table S2), $i, j = 0$.
repeat
  for $0 \leq k \leq N_{evaluate}$ **do**
    for $0 \leq l \leq N_{particle}$ **do**
      for $0 \leq m \leq N_{traj}$ **do**
        Generate terrain using $c^l_{T,j}$
        Initialize robot at random position
        Run policy π_i
        Compute traverability label for each state transition (Eq. 2)
        Save the scores and the trajectory
      end for
    end for
    Update policy using TRPO (*36*)
    $i = i + 1$
  end for
  for $0 \leq l \leq N_{particle}$ **do**
    Compute measurement probability for each parameter $c^l_{T,j}$s (Eq. 9)
  end for
  for $0 \leq l \leq N_{particle}$ **do**
    Update weights $w_j = \frac{P(y^l_i|c^l_{T,j})}{\sum_m P(y^m_i|c^m_{T,j})}$
  end for
  Resample $N_{particle}$ parameters
  Append $c_{T,j}$s to the replay memory
  for $0 \leq l \leq N_{particle}$ **do**
    by p_{replay} probability, sample from replay memory
    by $p_{transition}$ probability, move $c^l_{T,j}$ to an adjacent value in $\mathcal{C}$.
  end for
  $j = j + 1$
until Convergence