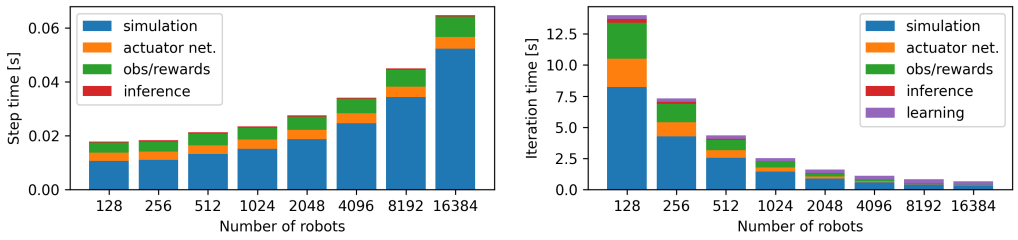
A Appendix



(a) (b)

Figure 1: (a) Computational time of an environment step. (b) Total time for a learning iteration with a batch size of $B = 98304$ samples.

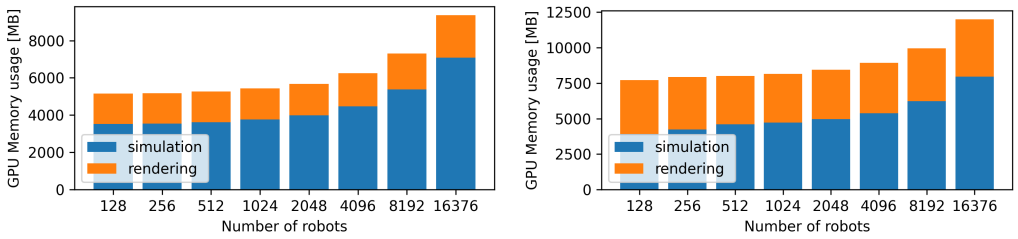
A.1 Simulation Throughput Analysis

In Fig. 1, we show the computational time of each part of an environment step, as well as the total times required for a learning iteration for different numbers of robots. In Fig. 1 (a) we observe that the simulation is the most time consuming step and its time slowly increases with the number of robots. The time spent on the computation of observations and rewards is the second slowest step and also slowly increases with the number of robots, while inference of the policy and the actuator network are computed in nearly constant time. Fig. 1 (b) shows the total time required to collect a fixed number of samples and perform the policy update. Increasing the number of parallel robots decreases the total time of all sub parts except the learning step which is independent on the number of robots. In Fig. 2 we examine the GPU VRAM required to train different number of robots both with and without graphical rendering. We see that $9\,\mathrm{Gb}$ are required to run 4096 robots with rendering enabled. Without a graphical output, $6\,\mathrm{Gb}$ are sufficient. On flat terrain these numbers are reduced to $7\,\mathrm{Gb}$ and $5\,\mathrm{Gb}$ respectively.

In Sec. A.1.1 and Sec. A.1.2 we describe additional techniques that we use to optimize the simulation throughput.

A.1.1 Time Step

The simulation time step needs to be maximized to get the maximum throughput. For each policy step, which we run at $50\,\mathrm{Hz}$, we need to perform multiple actuator and simulator steps to obtain a stable simulation. Since these added steps directly scale the amount of computation, we aim to reduce them as much as possible. We find that we can not use a time step smaller than $0.005\,\mathrm{s}$ which corresponds to four simulation steps per policy step. This limit is imposed by the actuator network (approximating a discrete-time PD-controller) becoming unstable and not by the simulation itself.



(a) Flat Terrain (b) Rough Terrain

Figure 2: GPU VRAM usage for different number of robots during training with a batch size of $B = 98304$ samples on flat and rough terrains.

A.1.2 Contact Handling

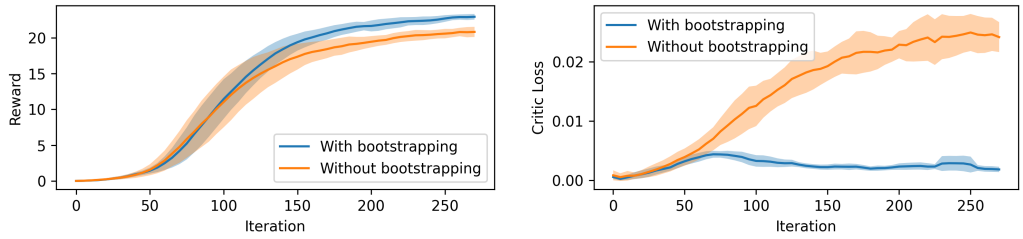
A large part of the simulator's computational time is spent on contact detection and handling. Reducing the number of potential contact pairs increases simulation throughput. We optimize the model of the robot to keep only the necessary collision bodies: feet, shanks, knees, and the base.

The resolution of the terrain plays an important role in contact optimization. A height field is a very common type of terrain representation, in which heights are defined on a uniform grid. However one of its unfortunate properties is that it is impossible to get vertical surfaces. In order to get a steep slope approximating a vertical step, we need high resolution, which degrades the simulation performance. Instead, we convert a low-resolution height field to a triangle mesh and correct the vertical surfaces.

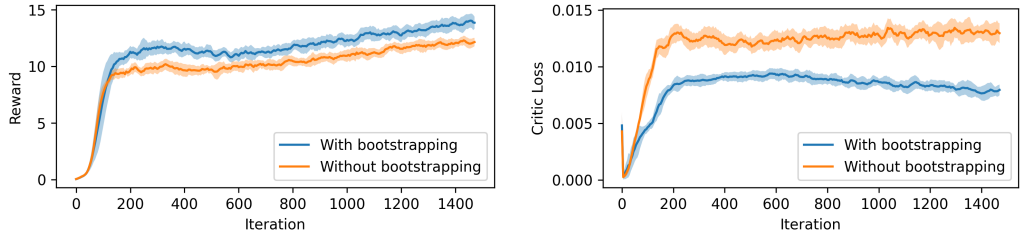
Finally, in PhysX (Isaac Gym's physics engine), even though contacts between the different robots are ignored, they are still detected. As such, the placement of robots in the terrain influences the computational load. Spreading the robots further apart from each other is highly beneficial. In the curriculum presented here, we need to place many robots close together at the beginning of training, but they quickly disperse as the training progresses. Additionally, once the robots learn to avoid crashes, there are fewer contacts with the base and knees and fewer costly resets. We see a factor of two difference in simulation time between the first minutes and the end of the training.

A.2 Effect of Time-out Bootstrapping

We analyze the effects of bootstrapping the reward at timeouts as described in Sec. 2.2.2 by comparing the total reward and the critic loss with and without the particular handling of timeouts. Fig. 3 shows the results for both flat and rough terrain tasks. We see that the critic loss is higher without bootstrapping, and correspondingly, the total reward is lower. Even though learning can be successful without this addition, it greatly reduces the critic loss and improves the total reward by approximately $10\,\%$ to $20\,\%$ for both tasks.



(a) Flat Terrain



(b) Rough Terrain

Figure 3: Comparison of total reward and critic loss, when training with and without reward bootstrapping on time-outs.

A.3 Reward Terms

Joint positions	$\mathbf{q}_j$
Joint velocities	$\dot{\mathbf{q}}_j$
Joint accelerations	$\ddot{\mathbf{q}}_j$
Target joint positions	$\mathbf{q}_j^*$
Joint torques	$\boldsymbol{\tau}_j$
Base linear velocity	$\mathbf{v}_b$
Base angular velocity	$\boldsymbol{\omega}_b$
Commanded base linear velocity	$\mathbf{v}_b^*$
Commanded base angular velocity	$\boldsymbol{\omega}_b^*$
Number of collisions	n_c
Feet air time	$\mathbf{t}_{air}$
Environment time step	dt

Table 1: Definition of symbols.

	definition	weight
Linear velocity tracking	$\phi(\mathbf{v}_{b,xy}^* - \mathbf{v}_{b,xy})$	$1dt$
Angular velocity tracking	$\phi(\boldsymbol{\omega}_{b,z}^* - \boldsymbol{\omega}_{b,z})$	$0.5dt$
Linear velocity penalty	$-\mathbf{v}_{b,z}^2$	$4dt$
Angular velocity penalty	$-\|\|\boldsymbol{\omega}_{b,xy}\|\|^2$	$0.05dt$
Joint motion	$-\|\|\ddot{\mathbf{q}}_j\|\|^2 - \|\|\dot{\mathbf{q}}_j\|\|^2$	$0.001dt$
Joint torques	$-\|\|\boldsymbol{\tau}_j\|\|^2$	$0.00002dt$
Action rate	$-\|\|\mathbf{q}_j^*\|\|^2$	$0.25dt$
Collisions	$-n_{collision}$	$0.001dt$
Feet air time	$\sum_{f=0}^{4}(\mathbf{t}_{air,f} - 0.5)$	$2dt$

Table 2: Definition of reward terms, with $\phi(x) := \exp(-\frac{||x||^2}{0.25})$. The z axis is aligned with gravity.

A.4 PPO Hyper-Parameters

Batch size	98304 (4096x24)
Mini-bach size	24576 (4096x6)
Number of epochs	5
Clip range	0.2
Entropy coefficient	0.01
Discount factor	0.99
GAE discount factor	0.95
Desired KL-divergence kl^*	0.01
Learning rate α	adaptive*

Table 3: PPO hyper-parameters used for the training of the tested policy. (*) Similarly to [1], we use an adaptive learning rate based on the KL-divergence, the corresponding algorithm is described in Alg. 1

Algorithm 1 Adaptive learning rate computation.

$kl \leftarrow KL(\pi_{new}, \pi_{old})$
if $kl > 2kl^*$ **then**
  $\alpha \leftarrow \max(10^{-5}, \alpha/1.5)$
else
  if $kl < 0.5kl^*$ **then**
    $\alpha \leftarrow \min(10^{-2}, 1.5\alpha)$
  end if
end if

A.5 Noise Level in Observations

Joint positions	$\pm 0.01\,\mathrm{rad}$
Joint velocities	$\pm 1.5\,\mathrm{rad/s}$
Base linear velocity	$\pm 0.01\,\mathrm{m/s}$
Base angular velocity	$\pm 0.2\,\mathrm{rad/s}$
Projected gravity	$\pm 0.05\,\mathrm{rad/s^2}$
Commanded base linear velocity	$0\,\mathrm{m/s}$
Commanded base angular velocity	$0\,\mathrm{rad/s}$
Measured terrain heights	$\pm 0.1\,\mathrm{m}$

Table 4: Noise scale for the different components of the observations. For each element, the noise value is sampled from a uniform distribution with the given scale and added to the observations.

References

[1] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.