

A Proof of Theorem 3.1

Theorem. $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ be an MDP with state space \mathcal{S} , action space \mathcal{A} , transition function $P : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, reward function $R : \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ and discount factor γ . Let $V^1(s)$ be the value function of the phase 1 policy that is trained to be close to optimal value function $V^*(s)$, i.e., $|V^*(s) - V^1(s)| < \epsilon$ for all $s \in \mathcal{S}$, and $\pi^1(s)$ be the greedy phase 1 policy obtained from $V^1(s)$. Suppose the phase 2 policy operates in a different state space \mathcal{S}' given by an invertible mapping $f : \mathcal{S} \rightarrow \mathcal{S}'$. If the phase 2 policy is close to phase 1 $|\pi^1(s) - \pi^2(f(s))| < \eta \forall s$ and R, P are Lipschitz continuous, then the return of phase 2 policy is close to optimal everywhere, i.e., for all s ,

$$|V^*(s) - V^{\pi^2}(f(s))| < \frac{2\epsilon\gamma + \eta c}{1 - \gamma}$$

where $c \propto \sum_{s \in \mathcal{S}} V^*(s)$ is a large but bounded constant.

Proof. Since the function f maps the state spaces $\mathcal{S}, \mathcal{S}'$, we can assume for convenience that both policies π^1, π^2 operate in the same state space \mathcal{S} . To obtain an action for $s' \in \mathcal{S}'$, we can simply query $\pi^2(f^{-1}(s'))$. Assume that the reward function R and transition function P are Lipschitz

$$|R(s_t, a_t) - R(s_t, a'_t)| \leq L_R |a_t - a'_t| \quad (12)$$

$$|P(s_{t+1} | s_t, a_t) - P(s_{t+1} | s_t, a'_t)| \leq L_P |a_t - a'_t| \quad (13)$$

for all states s_t, s_{t+1} and actions a_t, a'_t . We generalize the structure of proof for the upper bound on the distance in approximate optimal-value functions [84, 85] to the teacher-student setting. Let s_0 be the point where the distance between V^* and V^{π^S} is maximal

$$s_0 = \operatorname{argmax}_s V^*(s) - V^{\pi^S}(s) \quad (14)$$

Let Q^T be the Q-function corresponding to V^T . π^T is obtained by maximizing $Q^T(s, a)$ over actions $\pi^T(s) = \operatorname{argmax}_a Q^T(s, a)$. Note that in general the value function V^{π^T} may be different from V^T . Let a^* be the action taken by the optimal policy at state s_0 , while $a^T = \pi^T(s_0)$ be the action taken by the teacher. Then, the return of the teacher's greedy action a^T must be highest under teacher's value function Q^T ,

$$Q^T(s_0, a^*) \leq Q^T(s_0, a^T) \quad (15)$$

We can expand each side of (15) above to get

$$R(s_0, a^*) + \gamma \sum_{s \in \mathcal{S}} P(s | s_0, a^*) V^T(s) \leq R(s_0, a^T) + \gamma \sum_{s \in \mathcal{S}} P(s | s_0, a^T) V^T(s) \quad (16)$$

Notice that $|V^*(s) - V^T(s)| < \epsilon$ implies that $V^T(s) \in [V^*(s) - \epsilon, V^*(s) + \epsilon]$ which we can plug into the inequality above to get

$$\begin{aligned} & R(s_0, a^*) - R(s_0, a^T) \\ & \leq \gamma \sum_{s \in \mathcal{S}} [P(s | s_0, a^T) V^*(s) - P(s | s_0, a^*) V^*(s) + \epsilon (P(s | s_0, a^T) + P(s | s_0, a^*))] \\ & \leq \gamma \sum_{s \in \mathcal{S}} [P(s | s_0, a^T) V^*(s) - P(s | s_0, a^*) V^*(s)] + 2\epsilon\gamma \end{aligned} \quad (17)$$

We can now write a bound for $V^*(s_0) - V^{\pi^S}(s_0)$. Let a^S be the action taken by the student policy at state s_0 . Then,

$$\begin{aligned} & V^*(s_0) - V^{\pi^S}(s_0) \\ & = R(s_0, a^*) - R(s_0, a^S) + \gamma \sum_{s \in \mathcal{S}} P(s | s_0, a^*) V^*(s) - P(s | s_0, a^S) V^{\pi^S}(s) \quad (\text{substitute (12)}) \\ & \leq R(s_0, a^*) - R(s_0, a^T) + L_R |a^S - a^T| + \gamma \sum_{s \in \mathcal{S}} P(s | s_0, a^*) V^*(s) - P(s | s_0, a^S) V^{\pi^S}(s) \\ & \leq R(s_0, a^*) - R(s_0, a^T) + L_R \eta + \gamma \sum_{s \in \mathcal{S}} P(s | s_0, a^*) V^*(s) - P(s | s_0, a^S) V^{\pi^S}(s) \end{aligned}$$

We can now use (17) to write

$$\begin{aligned}
& V^*(s_0) - V^{\pi^S}(s_0) \\
& \leq 2\epsilon\gamma + L_R\eta + \gamma \sum_{s \in \mathcal{S}} P(s | s_0, a^T) V^*(s) - P(s | s_0, a^S) V^{\pi^S}(s) \quad (\text{substitute (13)}) \\
& \leq 2\epsilon\gamma + L_R\eta + \gamma \sum_{s \in \mathcal{S}} P(s | s_0, a^S) (V^*(s) - V^{\pi^S}(s)) + \eta\gamma L_P \sum_{s \in \mathcal{S}} V^*(s) \\
& = 2\epsilon\gamma + L_R\eta + \gamma(V^*(s) - V^{\pi^S}(s)) + \eta\gamma L_P \sum_{s \in \mathcal{S}} V^*(s) \\
& \quad (\text{since } s_0 \text{ is the state with highest difference between } V^* \text{ and } V^{\pi^S}) \\
& \leq 2\epsilon\gamma + L_R\eta + \gamma(V^*(s_0) - V^{\pi^S}(s_0)) + \eta\gamma L_P \sum_{s \in \mathcal{S}} V^*(s)
\end{aligned}$$

Rearranging yields,

$$V^*(s_0) - V^{\pi^S}(s_0) \leq \frac{2\epsilon\gamma + \eta(L_R + \gamma L_P \sum_{s \in \mathcal{S}} V^*(s))}{1 - \gamma}$$

Since s_0 is the state at which the difference between V^* and V^{π^S} is maximal, we can claim

$$V^*(s) - V^{\pi^S}(s) \leq \frac{2\epsilon\gamma + \eta c}{1 - \gamma}$$

for all states $s \in \mathcal{S}$, where $c = (L_R + \gamma L_P \sum_{s \in \mathcal{S}} V^*(s))$ □

B Rewards

Previous work [3, 17] has shown that task agnostic energy minimization based rewards can lead to the emergence of stable and natural gaits that obey high-level commands. We use this same basic reward structure along with penalties to prevent behavior that can damage the robot on complex terrain. Now onwards, we omit the time subscript t for simplicity.

- *Absolute work penalty* $-|\tau \cdot \mathbf{q}|$ where τ are the joint torques. We use the absolute value so that the policy does not learn to get positive reward by exploiting inaccuracies in contact simulation.
- *Command tracking* $v_x^{\text{cmd}} - |v_x^{\text{cmd}} - v_x| - |\omega_z^{\text{cmd}} - \omega_z|$ where v_x is velocity of robot in forward direction and ω_z is yaw angular velocity (x, z are coordinate axes fixed to the robot).
- *Foot jerk penalty* $\sum_{i \in \mathcal{F}} \|\mathbf{f}_t^i - \mathbf{f}_{t-1}^i\|$ where \mathbf{f}_t^i is the force at time t on the i^{th} rigid body and \mathcal{F} is the set of feet indices. This prevents large motor backlash.
- *Feet drag penalty* $\sum_{i \in \mathcal{F}} \mathbb{I}[f_z^i \geq 1\text{N}] \cdot (|v_x^i| + |v_y^i|)$ where \mathbb{I} is the indicator function, and v_x^i, v_y^i is velocity of i^{th} rigid body. This penalizes velocity of feet in the horizontal plane if in contact with the ground preventing feet dragging on the ground which can damage them.
- *Collision penalty* $\sum_{i \in \mathcal{C} \cup \mathcal{T}} \mathbb{I}[\mathbf{f}^i \geq 0.1\text{N}]$ where \mathcal{C}, \mathcal{T} are the set of calf and thigh indices. This penalizes contacts at the thighs and calves of the robot which would otherwise graze against edges of stairs and discrete obstacles.
- *Survival bonus* constant value 1 at each time step to prioritize survival over following commands in challenging situations.

The scales for these are $-1\text{e-}4, 7, -1\text{e-}4, -1\text{e-}4, -1, 1$. Notice that these reward functions do not define any gait priors and the optimal gait is allowed emerge via RL. Target heading values h_t^{cmd} are sampled and commanded angular velocities is computed as $(\omega_z^{\text{cmd}})_t = 0.5 \cdot (h_t^{\text{cmd}} - h_t)$ where h_t is the current heading value. When walking on terrain, $(v_x^{\text{cmd}})_t = 0.35\text{m/s}$ and heading is varied in $h_t^{\text{cmd}} \in [-10^\circ, 10^\circ]$. On flat ground, one of three sampling modes are chosen uniformly at random - curve following, in-place turning and complete stop. In the curve following regime $(v_x^{\text{cmd}})_t \in [0.2\text{m/s}, 0.75\text{m/s}]$ while $h_t^{\text{cmd}} \in [-60^\circ, 60^\circ]$. For in-place turning, $(v_x^{\text{cmd}})_t = 0$ and $h_t^{\text{cmd}} \in [-180^\circ, 180^\circ]$. In complete stop, $(v_x^{\text{cmd}})_t = 0, h_t^{\text{cmd}} = 0$. This scheme is designed to mimic the distribution of commands the robot sees during operation. We terminate if the pitch exceeds 90° or if the base or head of the robot collides with an object.

Algorithm 1 Pytorch-style pseudo-code for phase 2

Require: Phase 1 policy $\pi^1 = (G^1, F^1, \beta)$, parallel environments E , max iterations M , truncated timesteps T , learning rate η
Initialize phase 2 policy $\pi^2 = (G^2, F^2, \gamma)$ with $G^2 \leftarrow G^1, F^2 \leftarrow F^1$.
 $n \leftarrow 0$
while $n \neq M$ **do**
 Loss $l \leftarrow 0$
 $t \leftarrow 0$
 while $t \neq T$ **do**
 $s \leftarrow E.\text{observations}$
 $a^1 \leftarrow \pi^1(s)$
 $a^2 \leftarrow \pi^2(s)$
 $l \leftarrow l + \|a^1 - a^2\|_2^2$
 $E.\text{step}(a^2)$
 $t \leftarrow t + 1$
 end while
 $\Theta_{\pi^2} \leftarrow \Theta_{\pi^2} - \eta \nabla_{\Theta_{\pi^2}} l$
 $\pi^2 \leftarrow \pi^2.\text{detach}()$
 $n \leftarrow n + 1$
end while

Observation	a	b	σ
Joint angles (left hips)	1.0	0.1	0.01
Joint angles (right hips)	1.0	-0.1	0.01
Joint angles (front thighs)	1.0	0.8	0.01
Joint angles (rear thighs)	1.0	1.0	0.01
Joint angles (calves)	1.0	-1.5	0.01
Joint velocity	0.05	0.0	0.05
Angular velocity	0.25	0.0	0.05
Orientation	1.0	0.0	0.02
Scandots height	5.0	0.0	0.07
Scandots horizontal location	–	–	0.01

Table 2: During training, ground truth observations \mathbf{o}_t are shifted, normalized and noised to get observations \mathbf{o}'_t which are passed to the policy. $\mathbf{o}'_t = a(\mathbf{o}_t - b) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma)$. We tabulate a, b, σ for each kind of observation above. a, b values for scandots horizontal locations are blank since these locations are fixed with respect to the robot and not passed to the policy.

C Experimental Setup and Implementation Details

C.1 Pseudo-code

Phase 1 is simply reinforcement learning using policy gradients. We describe the pseudo-code for the phase 2 training in Algorithm 1.

C.2 Hardware

We use the Unitree A1 robot pictured in Figure 2 of the main paper. The robot has 12 actuated joints, 3 per leg at hip, thigh and calf joints. The robot has a front-facing Intel RealSense depth camera in its head. The compute consists of a small GPU (Jetson NX) capable of ≈ 0.8 TFLOPS and an UPboard with Intel Quad Core Atom X5-8350 containing 4GB ram and 1.92GHz clock speed. The UPboard and Jetson are on the same local network. Since depth processing is an expensive operation we run the convolutional backbone on the Jetson’s GPU and send the depth latent over a UDP socket to the UPboard which runs the base policy. The policy operates at 50Hz and sends joint position commands which are converted to torques by a low-level PD controller running at 400Hz with stiffness $K_p = 40$ and damping $K_d = 0.5$.

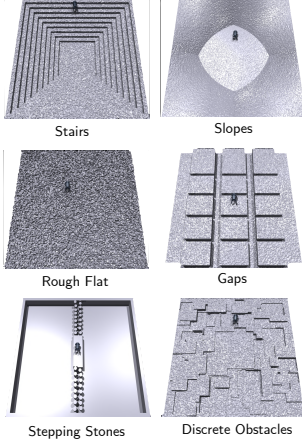


Figure 5: Set of terrain we use during training

Name	Range
Height map update frequency*	[80ms, 120ms]
Height map update latency*	[10ms, 30ms]
Added mass	[-2kg, 6kg]
Change in position of COM	[-0.15m, 0.15m]
Random pushes	Every 15s at 0.3m/s
Friction coefficient	[0.3, 1.25]
Height of fractal terrain	[0.02m, 0.04m]
Motor Strength	[90%, 110%]
PD controller stiffness	[35, 45]
PD controller damping	[0.4, 0.6]

Table 3: Parameter randomization in simulation. * indicates that randomization is increased to this value over a curriculum.

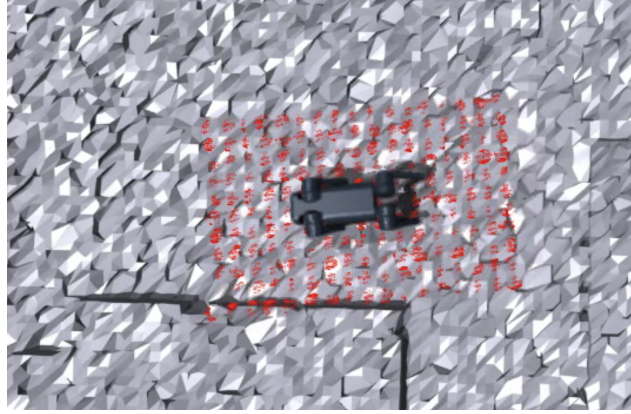


Figure 6: The privileged baseline receives terrain information from all around the robot including from around the hind feet.

Simulation Setup We use the IsaacGym (IG) simulator with the legged_gym library [18] to develop walking policies. IG can run physics simulation on the GPU and has a throughput of around $2e5$ time-steps per second on a Nvidia RTX 3090 during phase 1 training with 4096 robots running in parallel. For phase 2, we can render depth using simulated cameras calibrated to be in the same position as the real camera on the robot. Since depth rendering is expensive and memory intensive, we get a throughput of 500 time-steps per second with 256 parallel environments. We run phase 1 for 15 billion samples (13 hours) and phase 2 for 6 million samples (6 hours).

Environment We construct a large elevation map with 100 sub-terrains arranged in a 20×10 grid. Each row has the same type of terrain arranged in increasing difficulty while different rows have different terrain. Each terrain has a length and width of 8m. We add high fractals (upto 10cm) on flat terrain while medium fractals (4cm) on others. Terrains are shown in Figure 5 with randomization ranges described in Table 3.

Policy architecture The elevation map compression module β consists of an MLP with 2 hidden layers. The GRUs G^1, G^2 are single layer while the feed-forward networks F^1, F^2 have two hidden layers with ReLU non-linearities. The convolutional depth backbone γ consists of a series of 2D convolutions and max-pool layers.