APPENDIX

In *Appendix*, we further illustrate the technical details of training, deploying, and experiment setting. *Appendix* mainly has these sections:

- **A. Reward Functions:** the formula of each reward, the definition and analysis of regularization reward, and style reward.
- **B. Network Architectures:** the network architectures of Actor RNN, Critic RNN, Estimator Module, Latent Encoder, and Contact Encoder.
- **C. Dynamic Randomization:** details of the domain randomization and Gaussian noise.
- **D. Trap Terrain Setting in Simulation:** the trap terrain details for training.
- **E. Training Hyperparameters:** the hyperparameters of PPO, contact force, and t-SNE visualization.
- **F. Importance Analysis:** the method and formula for importance analysis.
- **G. Real-world Experiment Settings and Additional Results:** details of the experimental equipment and deployment, and additional experiments in the low-light environment.

A. Reward Functions

The reward function has three components: task reward r_t^T, regularization reward r_t^R, and style reward r_t^S.

In Sec. III-C, we have already introduced the **task reward**, which plays a major role in the training. In addition, **regularization reward** is used to optimize the performance of the robot. "Stall" reward is used to prevent the robot from stopping in the middle of the journey. "Velocity limit" reward is used to slow down the robot and ensure safety. "Joint velocity" reward and "Joint acceleration" reward are used to make the joint movements more stable and smooth. "Angular velocity stability" reward is used to make the base of the robot more stable. "Feet in air" reward is used to improve the gait and prevent the feet from rubbing on the ground. "Balance reward" is used to improve the left-right symmetry. For **style reward**, we first collect a dataset using an MPC controller. The dataset contains a state transition (s_t, s_{t+1}), with a time interval same as the RL policy. $s_t \in \mathbb{R}^{19}$ includes joint positions, base height, base linear velocity, and base angular velocity. We randomly select 200 velocity commands in the simulator. Each command lasts for two seconds and is converted to the next command continually. Following [13], [37], we train a Discriminator $\mathcal{D}_{amp}$ by the following loss function:

$$\begin{aligned} L_{discriminator} = & \mathbb{E}_{(s_t,s_{t+1})\sim \text{MPC}}\left[\left(\mathcal{D}_{amp}(s_t, s_{t+1}) - 1\right)^2\right] \\ & + \mathbb{E}_{(s_t,s_{t+1})\sim \text{Policy}}\left[\left(\mathcal{D}_{amp}(s_t, s_{t+1}) + 1\right)^2\right] \\ & + \alpha^{gp}\mathbb{E}_{(s_t,s_{t+1})\sim \text{MPC}}\left[\left\|\nabla \mathcal{D}_{amp}(s_t, s_{t+1})\right\|_2\right], \end{aligned}$$

And then we use $\mathcal{D}_{amp}$ to score the gait performance from policy output (s_t, s_{t+1}):

$$r_{style} = \max\left[0, 1 - 0.25\left(\mathcal{D}_{amp}(s_t, s_{t+1}) - 1\right)^2\right]. \quad (12)$$

B. Network Architectures

The details of network architectures are shown in Tab. IV.

TABLE IV: Network architecture details

Network	**Type**	**Input**	**Hidden layers**	**Output**
Actor RNN	LSTM	$\boldsymbol{p}_t, \boldsymbol{l}_t, \boldsymbol{g}_t$	[512, 256]	$\boldsymbol{a}_t$
Critic RNN	LSTM	$\boldsymbol{p}_t, \hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t, \boldsymbol{g}_t$	[512, 256]	$\boldsymbol{V}_t$
Estimator Module	LSTM	$\boldsymbol{p}_t, \boldsymbol{g}_t$	[256, 256]	$\boldsymbol{h}_t$
Latent Encoder	MLP	$\boldsymbol{h}_t$	[256, 256]	$\boldsymbol{l}_t$
Contact Encoder	MLP	$\hat{\boldsymbol{c}}_t$	[32, 16]	$\boldsymbol{l}_{t_c}$

TABLE III: Reward functions

Type	**Item**	**Formula**	**Weight**
Task	Get goal	$\frac{1}{0.4 + \|\Delta G\|_2}$	5.0
	Heading	$\begin{cases} \frac{\Delta x}{\|\Delta G\|_2 + \epsilon}, & \|\Delta G\|_2 \neq 0 \\ 1, & \|\Delta G\|_2 = 0 \end{cases}$	3.0
	Finish vel	$(\|v\| + \|\omega\|) \cdot (\|\Delta G\|_2 < 0.2)$	-1.0
	Finish pos	$(\sum_{i=1}^{12} \lvert q - q_{default}\rvert) \cdot (\|\Delta G\|_2 < 0.2)$	-1.0
	Alive	1	3.0
Regularization	Stall	$(\|V_{x,y}\|_2 < 0.1) \cdot (\|\Delta G\|_2 > 0.25)$	-2.0
	Vel limit	$(\omega_z < \omega_{limit}) \cdot (\|v_{x,y}\|_2 < v_{limit})$	2.0
	Joint vel	$-\|\dot{\mathbf{q}}\|_2$	0.002
	Joint acc	$-\|\ddot{\mathbf{q}}\|_2$	2×10^{-6}
	Ang vel stability	$-(\|\omega_{t,x}\|_2 + \|\omega_{t,y}\|_2)$	0.2
	Feet in air	$\sum_{i=0}^{3}(\mathbf{t}_{air,i} - 0.3) + 10 \cdot \max(0.5 - \mathbf{t}_{air,i}, 0)$	0.05
	Balance	$\|F_{feet,0} + F_{feet,2} - F_{feet,1} - F_{feet,3}\|_2$	-2×10^{-5}
Style	AMP	$\max\left[0, 1 - 0.25\left(\mathcal{D}_{amp}(s_t, s_{t+1}) - 1\right)^2\right]$	0.1

C. Dynamic Randomization

For better sim-to-real transfer, we introduce dynamic randomization including domain randomization and Gaussian noise. We have a series of domain randomizations including base mass, mass position, friction, initial joint positions, initial base velocity, and motor strength. the random ranges are shown in the Tab. V.

TABLE V: Domain randomization

Parameters	Range	Unit
Base mass	[0, 3]	kg
Mass position of X axis	[-0.2, 0.2]	m
Mass position of Y axis	[-0.1, 0.1]	m
Mass position of Z axis	[-0.05, 0.05]	m
Friction	[0, 2]	-
Initial joint positions	[0.5, 1.5] × nominal value	rad
Initial base velocity	[-1.0, 1.0] (all directions)	m/s
Motor strength	[0.9, 1.1] × nominal value	-
Proprioception latency	[0.2, 0.4]	s

Besides, we add Gaussian noise to the input observation, as shown in Tab. VI. This aims to simulate the noise of real robot sensors. Lots of experiments show that with dynamic randomization, the policy can be easily transferred from simulation to the real world without additional training.

TABLE VI: Gaussian noise

Observation	Gaussian Noise Amplitude	Unit
Linear velocity	0.05	m/s
Angular velocity	0.2	rad/s
Gravity	0.05	m/s^2
Joint position	0.01	rad
Joint velocity	1.5	rad/s

D. Trap Terrain Setting in Simulation

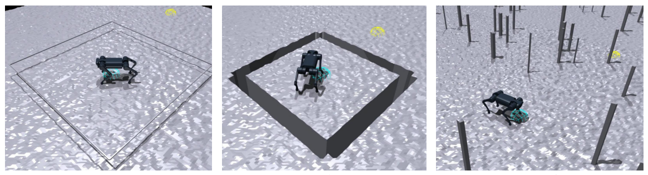

Fig. 13: Trap Terrain setting.

We employ "Terrain Curriculum" introduced in previous work [43] for better policy training. Due to the instability of reinforcement learning in early training, it is difficult for the policy to learn the movement in complex traps at once. Therefore, we design a trap curriculum to guide the policy from easy to difficult. The terrain is distributed in 10 rows and 10 columns. The terrains are divided into 4 categories. Each categorey has different traps ranging from easy to difficult, consisting of 10 variations. The column numbers of Bar, Pit, Pole, and Plane are 3,2,3,2. To prevent the robot from cheating by detouring, we put the bar and pit in a circle. As shown in Fig. 13, the robot is born inside the circle (blue point) and needs to reach outside the circle (yellow point). The height of the bar increases evenly from 0.05m to 0.25m, with a width randomizing in the range [0.025m, 0.1m]. The width of the pit increases evenly from 0.05m to 0.30m. The number of the pole increases evenly from 10 to 60, with a width randomizing in the range [0.025m, 0.1m]. In addition, we add perlin noise to all of the terrains with an amplitude in the range [0.05m, 0.15m].

E. Training Hyperparameters

In our work, we conduct a Policy Optimization algorithm (PPO) as our reinforcement learning method. The hyperparameters are shown in Tab. VII.

TABLE VII: PPO hyperparameters

Hyperparameter	Value
clip min std	0.05
clip param	0.2
desired kl	0.01
entropy coef	0.01
gamma	0.99
lam	0.95
learning rate	0.001
max grad norm	1
num mini batch	4
num steps per env	24

In the training step, we clip the contact force to $[0N, 100N]$.

In addition, we conduct t-SNE visualization in our additional experiments. The hyperparameters are shown in Tab. VIII.

TABLE VIII: T-SNE hyperparameters

Hyperparameter	Value
init	'random'
perplexity	30
learning rate	200

F. Importance Analysis

Assume the input $I \in \mathbb{R}^N$ and the output (action) $O \in \mathbb{R}^M$.

$$O = \text{Policy}(I), \tag{13}$$

First, we obtain the Jacobian matrix $J \in \mathbb{R}^{M\times N}$ by calculating the partial derivative.

$$J = \begin{bmatrix} \frac{\partial O_1}{\partial I_1} & \frac{\partial O_1}{\partial I_2} & \cdots & \frac{\partial O_1}{\partial I_n} \\ \frac{\partial O_2}{\partial I_1} & \frac{\partial O_2}{\partial I_2} & \cdots & \frac{\partial O_2}{\partial I_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial O_m}{\partial I_1} & \frac{\partial O_m}{\partial I_2} & \cdots & \frac{\partial O_m}{\partial I_n} \end{bmatrix}, \tag{14}$$

We take the absolute value for each term of the matrix.

$$J_{abs} = |J|, \tag{15}$$

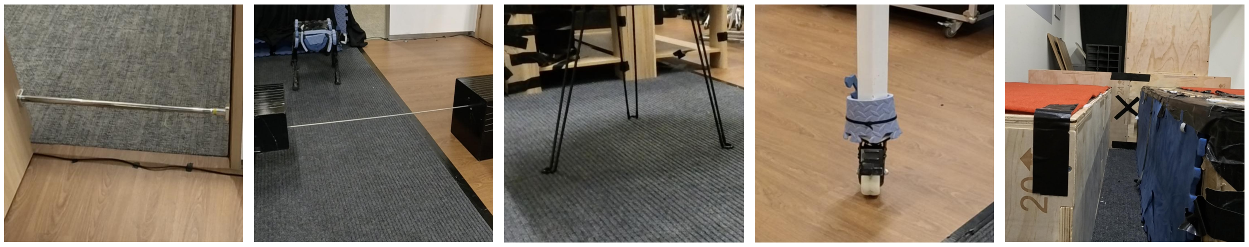
Fig. 14: Real-world traps: Thick Bar, Thin Bar, Thin Pole, Thick Pole, Pit.

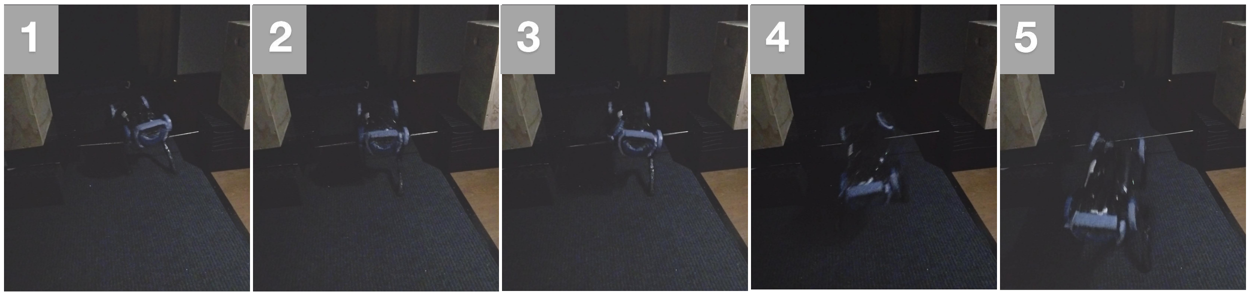
Fig. 15: Crossing *Bar* in low-light environment.

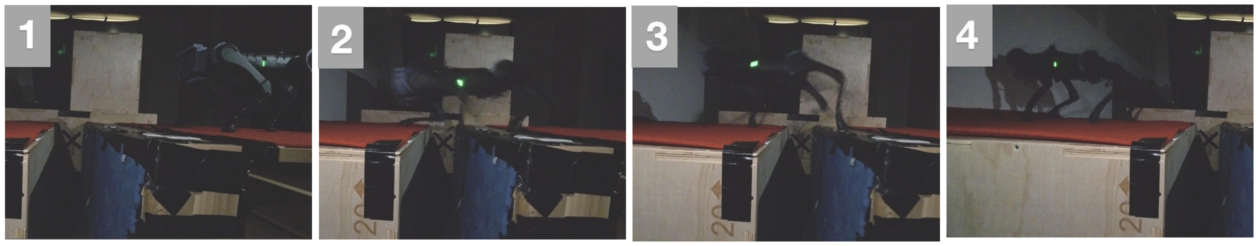
Fig. 16: Crossing *Pit* in low-light environment.

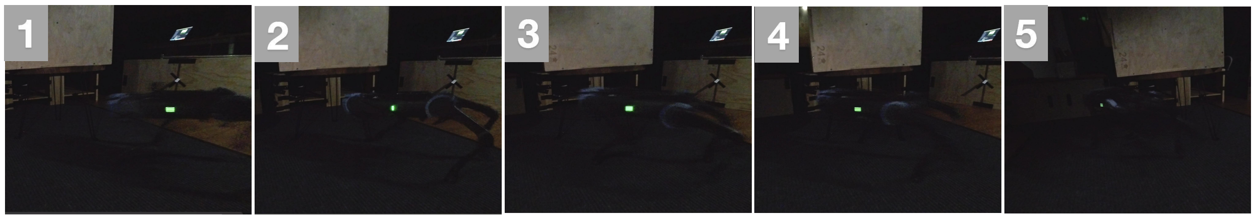
Fig. 17: Crossing *Pole* in low-light environment.

For each input I_i, we sum the corresponding output dimensions and align them with the upper U_i and lower bounds L_i to get the importance vector $S \in \mathbb{R}^N$.

$$S_i = (U_i - L_i) \cdot \sum_{j=1}^{M} J_{abs}(j,i), \quad i = 1,2,\cdots,N \tag{16}$$

For a group of input $\mathcal{G} \in I$, such as Contact force, Linear velocity, etc, we average importance for every input in $\mathcal{G}$.

$$S_{\mathcal{G}} = \frac{\sum_{I_i \in \mathcal{G}} S_i}{\text{num}(I_i \in \mathcal{G})}, \tag{17}$$

The group $\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_k$ is for comparison, we normalize them to get relative importance $R \in \mathbb{R}^k$ for each group.

$$R_i = \frac{S_{\mathcal{G}_i}}{\sum_{j=1}^{N} S_{\mathcal{G}_j}}, \quad i = 1,2,\cdots,k \tag{18}$$

G. *Real-world Experiment Settings and Additional Results*

We use common easily accessible items as traps for our real-world experiments, as shown in Fig. 14. For the *Bar* trap, there are two variations: thin bars and thick bars. The thin bars have a diameter of 6mm, while the thick bars measure 20mm in diameter. For the *Pit* trap, we separate two wooden boxes with a height of 40mm by some distance. For the *Pole* trap, there are also thin and thick poles. The thin poles are the legs of a iron table with a diameter of 8mm, while the thick poles include a range of obstacles made from thick iron poles and sticks of varying diameters.

We also conduct experiments in low-light environment, as shown in Fig. 15, Fig. 16, and Fig. 17. The robot can robustly move through different traps even if there is little light. The results demonstrate the effectiveness and importance of proprioception locomotion in scenarios where there is no visual input, such as during nighttime.