A Training Details of Low-level Locomotion Control Policy

We use the IsaacGym simulator [57] for policy training and deploy 4,096 quadruped robot agents. The training process has two steps as shown in Fig. 5. Both steps use the Proximal Policy Optimization (PPO)[58] method, and both are trained 40,000 iterations of exploration and learning. The control policy within the simulator operates at a frequency of 50 Hz.

A.1 Problem Definition

We decompose the locomotion control problem into discrete locomotion dynamics. The environment can be fully represented as $\boldsymbol{x}_t$ at each time step t, with a discrete time step $d_t = 0.02s$.

State Space: The entire training process includes the following three types of observation information: proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$, privileged state $\boldsymbol{s}_t \in \mathbb{R}^4$, and terrain information $\boldsymbol{t}_t \in \mathbb{R}^{187}$. Proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$ contains gravity vector $\boldsymbol{g}_t^p \in \mathbb{R}^3$ and base angular velocity $\boldsymbol{\omega}_t^p \in \mathbb{R}^3$ from IMU, velocity command $\boldsymbol{c_t} = \left(v_x^{\text{cmd}}, v_y^{\text{cmd}}, \omega_z^{\text{cmd}}\right) \in \mathbb{R}^3$, joint positions $\boldsymbol{\theta}_t^p \in \mathbb{R}^{12}$, joint velocities $\boldsymbol{\theta'}_t^p \in \mathbb{R}^{12}$, last action $\boldsymbol{a}_{t-1}^p \in \mathbb{R}^{12}$. Privileged state $\boldsymbol{s}_t \in \mathbb{R}^4$ contains base linear velocity $\boldsymbol{v}_t^p \in \mathbb{R}^3$ and the ground friction $\boldsymbol{\mu}_t^p \in \mathbb{R}^1$. Note that although the base linear velocity can be obtained by integrating the acceleration data from IMU, it has significant errors and accumulates errors over time, hence it cannot be used in the real deployment. Terrain information contains height measurement $\boldsymbol{i}_t^e \in \mathbb{R}^{187}$, which includes 187 height values sampled from the grid surrounding the robot, refer to the yellow point grid surrounding the robot in Fig. 5. We have two training steps as shown in Fig. 5. Policy in the first step uses all information $\boldsymbol{p}_t \in \mathbb{R}^{45}$, $\boldsymbol{s}_t \in \mathbb{R}^4$, and $\boldsymbol{t}_t \in \mathbb{R}^{187}$ as observation. In the second step and in the real deployment, the policy uses only proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$ as observation.

Action Space: The policy outputs the target positions of 12 joints as the action space $\boldsymbol{a}_t \in \mathbb{R}^{12}$. During real robot deployment, the expected joint positions are sent to the lower-level joint PD controllers $(K_p = 40, K_d = 0.5)$ for execution via the ROS (Robot Operating System) platform.

A.2 Reward Function

The reward function is composed of four components: task reward r_t^T, survival reward r_t^A, performance reward r_t^E, and style reward r_t^S. And the total reward is the sum of them $r_t = r_t^T + r_t^A + r_t^E + r_t^S$. Specifically, the task reward mainly consists of the tracking of linear and angular velocities, formulated as the exponent of the velocity tracking error; the alive reward gives a reward to the robot for each step to encourage it not to fall over; the performance reward includes energy consumption, joint velocity, joint acceleration, and angular velocity stability; the style reward includes the time the feet are off the ground and the balance of the forces on the feet, with the hope that the robot can walk with a more natural gait. The details of each reward function are shown in Table 5.

Table 5: Reward Function

Type	Item	Formula	Weight
Task	Lin vel	$\exp\left(-\lVert\mathbf{v}_{t,xy}^{\text{des}} - \mathbf{v}_{t,xy}\rVert_2/0.25\right)$	3.0
	Ang vel	$\exp\left(-\lVert\omega_{t,z}^{\text{des}} - \omega_{t,z}\rVert_2/0.25\right)$	1.0
Safety	Alive	1	1.0
Performance	Energy	$-\lVert\dot{\mathbf{q}}\rVert_2 \cdot \lVert\tau\rVert_2$	1×10^{-6}
	Joint vel	$-\lVert\dot{\mathbf{q}}\rVert_2$	0.002
	Joint acc	$-\lVert\ddot{\mathbf{q}}\rVert_2$	2×10^{-6}
	Ang vel Stability	$-(\lVert\omega_{t,x}\rVert_2 + \lVert\omega_{t,y}\rVert_2)$	0.2
Style	Feet in air	$\sum_{i=0}^{3}\left(\mathbf{t}_{air,i} - 0.3\right) + 10 \cdot \max\left(0.5 - \mathbf{t}_{air,i}, 0\right)$	0.05
	Balance	$-\lVert F_{feet,0} + F_{feet,2} - F_{feet,1} - F_{feet,3}\rVert_2$	2×10^{-5}

A.3 Termination Conditions

We terminate the episode when the robot base's roll angle (the rotation around the forward axis) exceeds 0.8 rad, the robot base's pitch angle (the rotation around the vertical axis) exceeds 1.0 rad, or the robot's position does not change significantly for over 1 second. If the robot does not trigger any termination conditions within 20 seconds or successfully arrives at the edge of one terrain, we also finish this episode and mark this episode as time out.

A.4 Terrain Curriculum

Previous work [59] has demonstrated that training quadruped robot on various terrains can result in high generalizability and robustness to different ground surfaces. Due to the instability of reinforcement learning in its early stages, it is challenging for robots to directly acquire locomotion skills on complex terrains. Therefore, we employs and refines the "terrain curriculum" approach proposed in [60]. Specifically, we create 80 different terrains, distributed across an 8×10 grid. The terrains are divided into 8 categories, with each type ranging from easy to difficult, consisting of 10 variations. Each terrain measures 8 meters in length and width. The first category consists of ascending stairs, with stair heights uniformly increasing from 0 to 0.2 meters, and a fixed stair width of 0.3 meters, designed for training in climbing stairs continuously. The second category features descending stairs, with stair heights uniformly increasing from 0 to 0.2 meters, and a fixed stair width of 0.3 meters, intended for training in descending stairs continuously. The third category comprises ascending platforms, with platform heights uniformly increasing from 0.16 to 0.22 meters, and platform widths varying randomly from 0.8 to 1.5 meters, used for training to step up onto higher platforms. The fourth category includes descending platforms, with platform heights uniformly increasing from 0.16 to 0.22 meters, and platform widths varying randomly from 0.8 to 1.5 meters, for training to jump down from higher platforms. The fifth category is for ascending ramps, with ramp angles uniformly increasing from 0 to 30 degrees, aimed at training for climbing up ramps. The sixth category is for descending ramps, with ramp angles uniformly increasing from 0 to 30 degrees, aimed at training for descending ramps. The seventh category consists of flat ground with no obstacles, for training in walking on level surfaces. The eighth category is rough terrain, with the addition of Perlin noise with amplitudes uniformly increasing from 0 to 0.15 meters, for training on uneven surfaces such as rocky roads.

A.5 Dynamic Randomization

To enhance the robustness and reduce the gap between the simulation and reality, we have a series of randomizations including the mass, the center of gravity position, the initial joint positions, the motor strength, and the coefficient of friction, all of which are subject to random variation within a preset range. Details are in Table 6.

Table 6: Dynamic 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] $\times$nominal value	rad
Initial base velocity	[-1.0, 1.0] (all directions)	m/s
Motor strength	[0.9, 1.1] $\times$nominal value	-

In addition, the observation information obtained by the robot's sensors is also added with random Gaussian noise to simulate the sensor errors that may occur in a real environment. Details are in Table 7.

Table 7: 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

Furthermore, we randomly change the robot's velocity commands every 5 seconds and apply random external forces to the robot every 9 seconds.

A.6 Network Architecture

In the first step of training, the terrain encoder E_t and the low-level MLP E_{low} are both multilayer perceptrons (MLPs). In the second step of training, the estimator consists of a recurrent neural network (RNN) and a multilayer perceptron (MLP), with the type of recurrent neural network being a Long Short-Term Memory network (LSTM). The specific details of the network are shown in Table 8.

Table 8: Details of the network architecture

Network	Input	Hidden layers	Output
E_t(MLP)	$\boldsymbol{t}_t$	[128, 64]	$\boldsymbol{t}_{l_t}$
E_{low}(MLP)	$\boldsymbol{p}_t, \boldsymbol{s}_t, \boldsymbol{t}_t$	[512, 256, 128]	$\boldsymbol{a}_t$
Estimator LSTM	$\boldsymbol{p}_t$	[256, 256]	$\boldsymbol{h}_t$
Estimator MLP	$\boldsymbol{h}_t$	[256, 128]	$\boldsymbol{p}_t$
Critic(MLP)	$\boldsymbol{p}_t, \boldsymbol{s}_t, \boldsymbol{t}_t$	[512, 256, 128]	$\boldsymbol{V}_t$
Terrain Estimator (MLP)	$\boldsymbol{p}_t, \boldsymbol{s}_t, \boldsymbol{t}_t$	[256, 128]	$\boldsymbol{c}_t$

A.7 Hyperparameters

The hyperparameters of the PPO algorithm are shown in the Table. 9:

Table 9: 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

B Experiment setup details

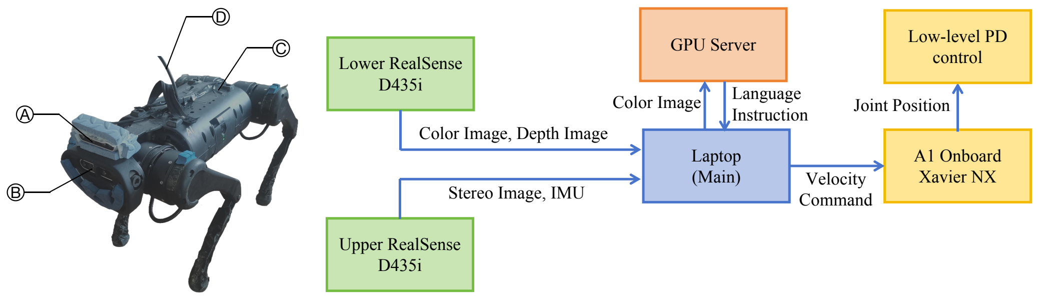

Fig. 8. Robot setup for experiment. A: Upper RealSense D435i camera. B: Lower RealSense D435i camera. C: Unitree A1 quadruped robot equipped with NVIDIA Jetson Xavier NX. D: Network cable for communication between quadruped robot and laptop. Based on ROS, we set up our communication system. We use a Ubuntu 20.04 laptop as the main computer, a GPU server with 8 NVIDIA RTX 3090 as the side computer, and NVIDIA Jetson Xavier NX as the onboard computer. The messages from cameras are directly sent to the laptop. We use the laptop to run the SLAM program and the system's main program. The GPU server runs the LLaVA program and communicates color image and language instruction with the laptop. The onboard Xavier NX receives the velocity command by the main program from the laptop through ROS message and runs the low-level control policy to predict desired joint positions for PD control.

We deploy two RealSense D435i cameras on the quadruped robot, the upper one for VIO to get robot odometry, and the lower one for high-level reasoning. We use LLaVA-34B[61] as our vision language foundation model, and VINS-Fusion[62] as the VIO algorithm. Detailed robot setup for experiment is shown in Fig. 8. Note that we paste some rectangle decorations on the white wall of the laboratory, which is only to provide feature points for the VIO algorithm and improve its stability.

C Extra Experiments Details of the Low-level Locomotion

C.1 Metric of velocity tracking ratio

$$\text{Linear velocity tracking ratio} = \exp(-\frac{\|v_{x,y} - v_{x,y}^{\text{target}}\|_2^2}{0.25}), \tag{5}$$

$$\text{Angular velocity tracking ratio} = \exp(-\frac{\|\omega_{\text{yaw}} - \omega_{\text{yaw}}^{\text{target}}\|_2^2}{0.25}). \tag{6}$$

C.2 Comparison Experiments

1) RMA [1]: A 1D-CNN is used as an adaptation module, employing asynchronously. The teacher-student training framework is used.

2) IL [50]: The first step of training is the same, the second step of training employs the teacher-student framework for imitation learning. The network architectures are the same.

3) Built-in MPC: The built-in Model Predictive Control (MPC) controller on the Unitree A1 robot (only in physical experiments).

4) Blind: The network architecture is the same as that in the second step of training. Trained only using proprioception directly in one step.

5) Concurrent [56]: The policy was trained concurrently with a state estimation network. The training process did not include any input regarding the terrain.

C.3 Ablation Experiments

1) Exp 0.9998: The selection probability decreases exponentially, with a base of 0.9998.

2) Exp 0.9995: The selection probability decreases exponentially, with a base of 0.9995.

3) No anneal: The selection probability is set to zero from the beginning, and the predicted hidden state values are used exclusively.

4) Cosine: The selection probability decreases in the shape of the cosine function on the interval $[0, \pi]$. The rate of probability decrease is initially slow and then accelerates.

5) Linear: The selection probability decreases in a linear function. The probability decreases uniformly.

Specifically, the annealing schedule of exponent, cosine, and linear is shown in the Fig. 9.

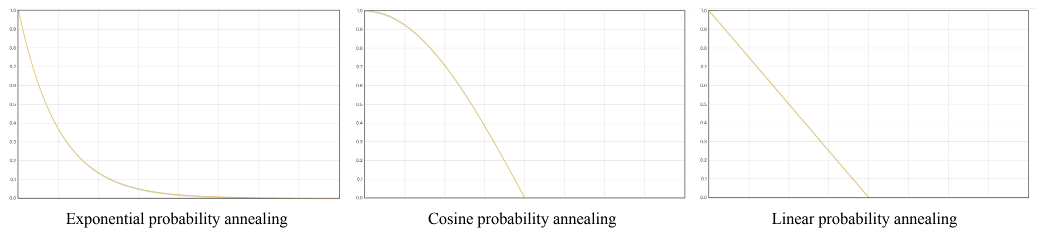

Fig. 9. Annealing schedule of different settings. Exponential annealing is fast initially and then slows down, cosine annealing is slow initially and then speeds up, and linear annealing is uniform all the process.

D Supplementary Materials

The real-world experiment demos can be found in the supplementary materials. ”main.mp4” is a brief introduction of our method and a display of the main experiment results. ”stair_indoor.mp4”, ”ramp_indoor.mp4”, ”gap.mp4”, ”door.mp4”, ”outdoor.mp4” is the detailed experiment results of each scene. The videos have undergone $4\times$ acceleration processing. ”locomotion.mp4” shows our extra experiments of the low-level locomotion part, including walking on rubble, grassland and continually climb the stair. ”vint.mp4” shows that previous work ViNT[36] cannot efficiently complete the task.