APPENDIX

In *Appendix*, we introduce technical details of our system implementation, training and experiments. The *Appendix* includes three parts for each, as listed below:

- **A. Details of High-level Reasoning and Task Execution**: the implementation details of our high-level system, including the VLM prompts, sub-task set and the deployed algorithm.
- **B. Training Details of Low-level Locomotion Control Policy**: the training details of our low-level locomotion control policy, including the problem definition, reward function, training strategy and parameters.
- **C. Extra Experiments Details of the Low-level Locomotion**: the supplementary experiment of our low-level locomotion control policy, including the metrics, comparison with baselines and ablation studies.

A. Details of High-level Reasoning and Task Execution

1) VLM Details and Prompts: We use pretrained LLaVA-34B for VLM inference. The VLM takes RGB images at a resolution of 848×480 and instruction prompts as input. The prompts given for different stages are listed below:

1) **Planning** "Ignore anything on the wall. You are a robot dog. The intermediation may be a stair, a ramp, a gap, or a door frame. The task is *task*. First answer the question: 1. What is the only intermediation you need to cross or climb to finish the task? Based on previous questions, decompose this task into a sequence of subtasks. The subtask is (Action, Ending). Action is one of ['move', 'climb']. The ending is one of ['facing intermediation', 'across intermediation', and 'to the goal']. Replace the intermediation with the answer to question 1."
2) **Perception** "Where is the *intermediation*? Answer in [x0,y0,x1,y1] format, don't say anything else."
3) **Discriminator** "Is there any *intermediation*? Just answer yes or no." or "Is the *task* finished at current state?"

Note after the output of prompt 1, all the *intermediation* in prompts will be replaced by the specific intermediation in answer 1. One example output with a specific terrain can be found in Fig. 2

2) Sub-task Set and Deployment Details: The sub-task is defined as an *(Action, Ending)* pair. *Action* is one of ['move', 'climb'], *Ending* is one of ['facing intermediation', 'across intermediation', and 'to the goal']. In real deployments, we find the VLM does not output irrational sub-tasks "climb facing intermediation" and "climb to the goal", so only four pairs are used: "Move facing intermediation", "Move across intermediation", "Move to the goal", and "Climb across intermediation". The sub-task execution module can access sub-task instructions from VLM, RGB-D images, odometry from VIO SLAM, and output a velocity command (V_x, V_y, V_{yaw}) to the PAS control policy. The sub-tasks are executed in a closed loop and double-checked by VLM. While one sub-task is *Not* at the **End Point** judged by VLM, the system consistently executes it.

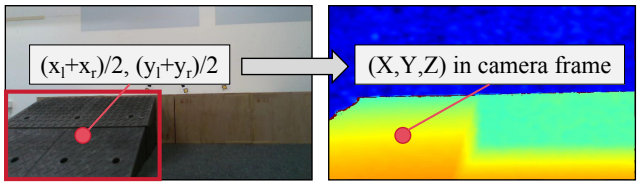


Fig. 8: Illustration of the trajectory refinement module combined with depth image.

Algorithm 1: **"Move facing intermediation".** The VLM is capable of detecting intermediations and outputting accurate bounding boxes for them. Combining with a depth image input, we can obtain the exact 3D position of the intermediation by $\{X,Y,Z\} = \{\frac{(i-x_0)\cdot d_{ij}}{f_x}, \frac{(j-y_0)\cdot d_{ij}}{f_y}, d_{ij}\}$, and output an accurate sub-goal based on that, as shown in Fig. 8.

Algorithm 2: **"Move across intermediation".** Move forward to the sub-goal located at the same horizontal line as the final goal.

Algorithm 3: **"Move freely to goal".** Integrated with the localization module, move freely to the final goal until the distance between the robot and the sub-goal is less than 0.1m.

Algorithm 4: **"Climb across intermediation".** The generalizable locomotion control policy consistently classifies whether the robot is on the plane or is crossing an intermediation, which is discussed in detail in Section V-B.6. If the terrain estimator identifies sudden changes from terrain to plane, the "done" signal is thrown by the task execution module.

Algorithm 1 Move facing intermediation

Input: RGB image C; Aligned depth image D; Odometry O
Output: Velocity command cmd_{vel};
Query VLM for intermediation bounding box $\{x_l, x_r, y_l, y_r\}$;
Obtain intermediation center in pixel $\{x_0, y_0\} = \{(x_l + x_r)/2, (y_l + y_r)/2\}$;
while *Not* $x_0 < thereshold$ **do**
  Obtain intermediation center in real 3D position $\{X,Y,Z\} = \{\frac{(i-x_0)\cdot d_{ij}}{f_x}, \frac{(j-y_0)\cdot d_{ij}}{f_y}, d_{ij}\}$;
  Transfer sub-goal $\{0, X, 0\}$ in robot frame to fixed world frame;
  while Distance error greater than 0.1m **do**
    Calculate command velocity cmd_{vel} by PD control based on the sub-goal;
    Send cmd_{vel} to PAS control policy;
  end while
  Query VLM for intermediation bounding box $\{x_l, x_r, y_l, y_r\}$;
  Obtain intermediation center in pixel $\{x_0, y_0\} = \{(x_l + x_r)/2, (y_l + y_r)/2\}$;
end while

Algorithm 2 Move across intermediation

Input: Odometry O
Output: Velocity command cmd_{vel};
  while *Not* **invisible of the intermediation** judged by VLM discriminator **do**
    Obtain final goal X, Y, Z in robot frame;
    Transfer sub-goal $\{X, 0, 0\}$ in robot frame to fixed world frame;
    while Distance error greater than 0.1m **do**
      Calculate command velocity cmd_{vel} by PD control based on the sub-goal;
      Send cmd_{vel} to PAS control policy;
    end while
  end while

Algorithm 3 Move freely to the goal

Input: Odometry O
Output: Velocity command cmd_{vel};
  while *Not* **finished the task defined by language** $\mathcal{L}$ judged by VLM discriminator **do**
    Obtain final goal $\{X, Y, Z\}$;
    while Distance error greater than 0.1m **do**
      Calculate command velocity cmd_{vel} by PD control based on the final goal;
      Send cmd_{vel} to PAS control policy;
    end while
  end while

Algorithm 4 Climb across intermediation

Input: Terrain Classification; Odometry O
Output: Velocity command cmd_{vel};
  while *Not* **invisible of the intermediation** judged by VLM discriminator **do**
    $TerrainChangeCount = 0$;
    while $Count < 2$ **do**
      Calculate command velocity cmd_{vel} by PD control to keep moving and facing straight;
      Send cmd_{vel} to PAS control policy;
      $Count = Count + Classification \oplus Last\_Classification$;
    end while
  end while

B. Training Details of Low-level Locomotion Control Policy

We use the IsaacGym simulator [65] for policy training and deploy 4,096 quadruped robot agents. The training process has two steps as shown in Fig. 4. Both steps use the Proximal Policy Optimization (PPO) [66] method, and both are trained in 40,000 iterations of exploration and learning. The control policy within the simulator operates at a frequency of 50 Hz.

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. 4. We have two training steps as shown in Fig. 4. 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.

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 V.

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.

4) Terrain Curriculum: Previous work [67] 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 employ and refine the "terrain curriculum" approach proposed in [68]. Specifically, we create 80 different terrains, distributed across an 8×10 grid. The terrains are divided into 8 categories,

TABLE V: Reward Function

Type	Item	Formula	Weight
Task	Lin vel	$\exp\left(-\|\mathbf{v}_{t,xy}^{\text{des}} - \mathbf{v}_{t,xy}\|_2/0.25\right)$	3.0
	Ang vel	$\exp\left(-\|\omega_{t,z}^{\text{des}} - \omega_{t,z}\|_2/0.25\right)$	1.0
Safety	Alive	1	1.0
Performance	Energy	$\|\dot{\mathbf{q}}\|_2 \cdot \|\tau\|_2$	-1×10^{-6}
	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
Style	Feet in air	$\sum_{i=0}^{3}(\mathbf{t}_{air,i} - 0.3) + 10 \cdot \min(0.5 - \mathbf{t}_{air,i}, 0)$	0.05
	Balance	$\|F_{feet,0} + F_{feet,2} - F_{feet,1} - F_{feet,3}\|_2$	-2×10^{-5}

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.

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 VI.

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 VII.

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

TABLE VI: 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] ×nominal value	rad
Initial base velocity	[-1.0, 1.0] (all directions)	m/s
Motor strength	[0.9, 1.1] ×nominal value	-

TABLE VII: Gaussian noise

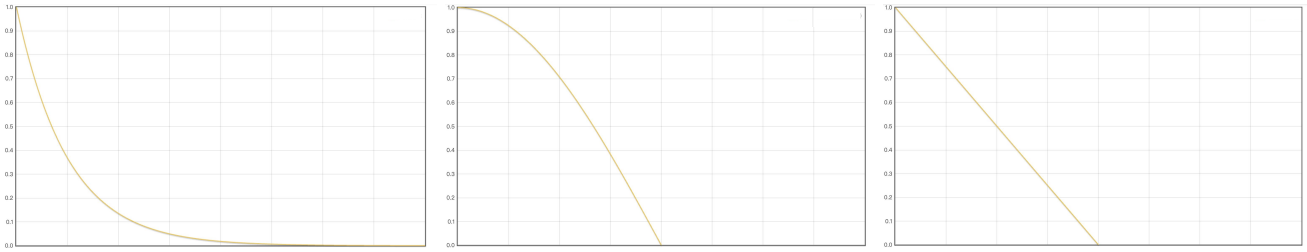
Observation	Gaussian Noise Amplitude	Unit
Linear velocity	0.05	m/s
Angular velocity	0.2	rad/s
Gravity	0.05	m/s^2
Joint position	0.01	rad
Joint velocity	1.5	rad/s

6) Terrain Classification: After finishing the training process of the PAS control policy, we freeze all the network weight in the PAS control policy and add head to output a boolean terrain classification. The input of the network is $\boldsymbol{p}_t$, $\boldsymbol{s}_t$, $\boldsymbol{t}_t$, and only to predict robot is on the plane or not (on the intermediation). We use BCE Loss as the loss function.

7) 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 VIII.

TABLE VIII: 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]	$\boldsymbol{c}_t$



Exponential probability annealing | Cosine probability annealing | Linear probability annealing

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.

8) Hyperparameters: The hyperparameters of the PPO algorithm are shown in the Table. IX:

TABLE IX: 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

C. Extra Experiments Details of the Low-level Locomotion

1) Metric of Velocity Tracking Ratio:

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

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

2) Comparison Experiments:

- RMA [1]: A 1D-CNN is used as an adaptation module, employing asynchronously. The teacher-student training framework is used.
- IL [55]: 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.
- Built-in MPC: The built-in Model Predictive Control (MPC) controller on the Unitree A1 robot (only in physical experiments).
- Blind: The network architecture is the same as that in the second step of training. Trained only using proprioception directly in one step.
- Concurrent [64]: The policy was trained concurrently with a state estimation network. The training process did not include any input regarding the terrain.

3) Ablation Experiments:

- Exp 0.9998: The selection probability decreases exponentially, with a base of 0.9998.
- Exp 0.9995: The selection probability decreases exponentially, with a base of 0.9995.
- No anneal: The selection probability is set to zero from the beginning, and the predicted hidden state values are used exclusively.
- 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.
- 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 Fig. 9.