# Cross Anything: General Quadruped Robot Navigation through Complex Terrains

Shaoting Zhu*,[1,2], Derun Li*,[1,3], Yong Liu[2], Ningyi Xu[3], and Hang Zhao[1,4]

[1]Shanghai Qi Zhi Institute, [2]Zhejiang Univerisity,
[3]Shanghai Jiao Tong University, [4]Tsinghua University

**Abstract:** The application of vision-language models (VLMs) has achieved impressive success in various robotics tasks, but there are few explorations for foundation models used in quadruped robot navigation. We introduce **C**ross **A**nything **S**ystem (CAS), an innovative system composed of a high-level reasoning module and a low-level control policy, enabling the robot to navigate across complex 3D terrains and reach the goal position. For high-level reasoning and motion planning, we propose a novel algorithmic system taking advantage of a VLM, with a design of task decomposition and a closed-loop sub-task execution mechanism. For low-level locomotion control, we utilize the **P**robability **A**nnealing **S**election (PAS) method to train a control policy by reinforcement learning. Numerous experiments show that our whole system can accurately and robustly navigate across complex 3D terrains, and its strong generalization ability ensures the applications in diverse indoor and outdoor scenarios and terrains. Project page: https://cross-anything.github.io/

**Keywords:** Quadruped robot, 3D terrain navigation, Vision-language model

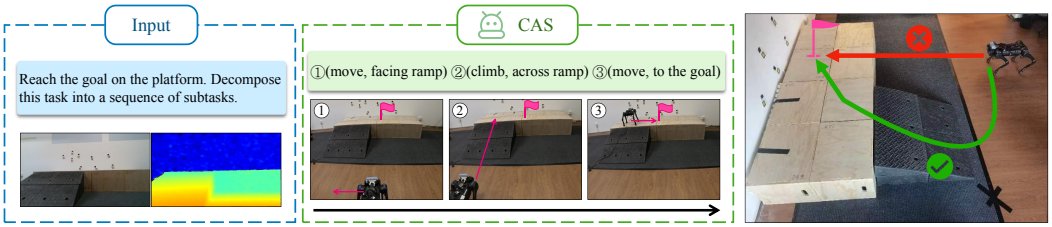


Fig. 1. CAS achieves general autonomous complex terrain navigation capability in the 3D environment. Our system utilizes the motion planning ability of the VLM model, with a design of task decomposition and closed-loop sub-task execution mechanism. While traditional navigation approaches (red) fail in reasoning 3D environment, our system (green) guides the quadruped robot to cross the accessible terrain towards the goal.

## 1 Introduction

The athletic intelligence of animals is concentrated in their understanding of complex wild environments and their ability to reach invisible destinations. This significantly challenges the capacity for 3D scene understanding and traversability across various terrains, which should be considered as the potential advantage for quadruped robot agents. Although much progress has been made in specific locomotion skills [1, 2], the autonomy of robots should be improved at the system level. To the best of our knowledge, we are the first to build a fully autonomous system using a vision-language model, auxiliary modules, and a low-level control policy to cover the complex tasks of perception, 3D motion planning, and locomotion, achieving good performance in reaching targets hidden in complex 3D terrains, as shown in Figure 1.

Vision-language models (VLMs) have shown advancements in common sense reasoning and remarkable generalization in vision tasks, significantly boosting the progress of robotic learning[3, 4,

∗ Indicates equal contribution. Under review.

5, 6]. However, VLMs suffer from the limitations of training data perspectives and the lack of a memory information bank, which is believed to curtail their usage in robot navigation tasks. Our motivation originates from this important question: *How can we design a system to fully activate the potential of VLMs' visual common sense on robots to enable them to observe and understand the 3D world?* In this work, we design a system called **C**ross **A**nything **S**ystem (CAS), composed of a zero-shot VLM and several auxiliary motion modules to enhance the 3D reasoning and motion planning ability.

Cooperating with our high-level motion planning using VLMs, the locomotion control policy needs to be adaptable to different terrains and robust against diverse environments. Traditional control methods such as SLIP[7], VMC[8], MPC[9, 10] can handle some specific terrain tasks, but they have poor robustness against complex real-world situations. Adaptation learning [1] and teacher-student framework[11] are employed to address the transferring from simulation to real-world. However, they are prone to significant performance degradation when deployed in the real-world. In our work, we propose a novel method called **P**robability **A**nnealing **S**election (PAS) to solve the sim-to-real transfer problem caused by mimic learning.

Our experiment results demonstrate the generalization and common-sense visual motion reasoning abilities of CAS in quadruped robot 3D navigation. We test our method across different categories of terrains and routes, and also showcase the capacity of our locomotion policy in extra experiments. In real-world scenarios, our method exhibits strong generalization ability and robustness, as demonstrated in our real-world videos. In summary, our contributions are as follows:

**1)** We innovatively present CAS, a system that is composed of a vision-language model and some auxiliary modules, enhancing ego-view 3D generalizable navigation for quadruped robots.

**2)** We introduce a novel reinforcement learning-based locomotion control policy, **P**robability **A**nnealing **S**election (PAS) to overcome various 3D terrain challenges.

**3)** Experiments across extensive categories of terrains and routes demonstrate the effectiveness, robustness, and generalization ability of our system.

## 2 Related Work

**Foundation Models in Robotics:** In recent years, foundation models [12, 13, 14, 15, 16], including large language models (LLMs) and vision-language models (VLMs), have achieved significant advancements. Furthermore, some of these foundation models have been adapted to the domain of robotics [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]. Several works [30, 20, 31, 32] have utilized open-vocabulary pretrained models for robotic tasks, Others [33, 3, 4, 5, 34] have employed powerful VLMs, such as GPT-4V, for robotics tasks. Moreover, some researchers have attempted to apply foundation models to quadruped robots. Saytap [35] uses large language models (LLMs) to translate natural language commands into foot contact patterns for quadrupedal robots. ViNT [36] trains a universal policy from a large-scale visual navigation dataset using the Transformer [37]. CognitiveDog [38] integrates a Large Multi-modal Model (LMM) with a quadruped robot. GeRM [39] trains a generalist model for quadruped robots in vision-language tasks. Nevertheless, all these methods are only suitable for tasks on planar surfaces and do not fully utilize the 3D complex terrain capabilities of quadruped robots.

**Locomotion Control of Quadruped Robots:** Traditional locomotion control methods for quadruped robots [7, 8, 40, 41, 42] is one way for locomotion control, but they often suffer the unstable problem in real-world deployment. Reinforcement learning has shown remarkable capabilities in recent years[43, 11, 1]. They utilize the privileged training paradigm to train quadruped robots without extra sensors. Also, some work [44, 45, 46, 47] integrate proprioceptive and exteroceptive states to achieve agile locomotion. Mimic learning is frequently used in previous works. Methods of adaptation learning [1, 48, 2] and teacher-student framework learning [11, 49, 50] are used to solve the sim-to-real transfer problem, but they suffer from high performance reduction during real deployment. Meanwhile, some other works propose innovative methods to improve lo-

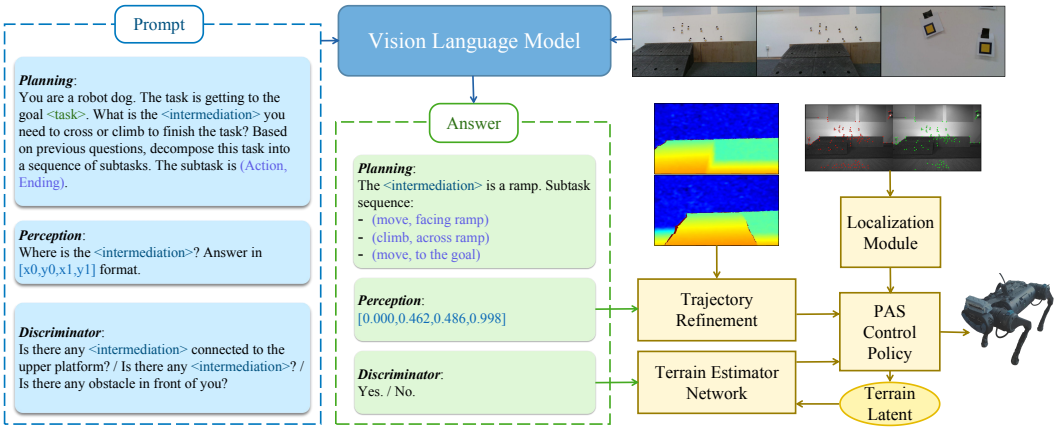


Fig. 2. Overview of the high-level quadruped robot navigation system. The pretrained vision-language foundation model (VLM) takes as input color images and prompts querying the 3D environment perception. The auxiliary modules include a localization module, a trajectory refinement module, and a terrain estimator network. The VLM decomposes the task into sub-tasks and executes each task in closed loops in sequence.

comotion efficiency. DayDreamer [51] learns a "world model" to synthesize infinite interactions, while DreamWaQ [52] implicitly infers terrain properties and adapts its gait accordingly by learning a VAE model. Traditional control methods are combined with deep reinforcement learning [53, 54] to accelerate training speed, but they do not fully utilize privileged information in simulation with only one-stage training. In our work, we propose a novel method to solve the sim-to-real transfer problem without mimic learning, while fully taking advantage of privileged information in simulation with two-stage training.

## 3 Method

We build a long-term motion planning and control system to guide the quadruped robot navigating to the goal in the complicated 3D environment with terrains. The system is composed of a VLM for high-level reasoning and motion planning, auxiliary modules for localization and trajectory refinement, and a low-level locomotion controlling policy to cover movements across different terrains, as shown in Fig. 2.

### 3.1 Task Definition

The task entails enabling the quadruped robot to autonomously navigate through a 3D environment with various terrains. The goal is to reach a specified point defined as $(x, y, z, yaw)$ relative to the robot's starting position and a language description of the goal. The 3D terrain encompasses stairs, ramps, gaps, and doors. However, the robot cannot just walk straight to the goal; it needs to find the best way to pass through the terrain. The quadruped robot can only access the sensors onboard, including proprioception, ego-view color image, and depth image.

### 3.2 High-level Reasoning and Motion Planning

**Task Decomposition:** Since we do not make use of maps such as a semantic map and a topological map, our system works as a state machine and decomposes the long-term navigation into a sub-task sequence composed of movement actions and the ending point by prompting the VLM. The prompt is based on the role and the task and defines intermediation as the terrain to cross. As illustrated in Fig. 2, we use the pre-trained VLM to perform zero-shot inference on ego-view image inputs. It first recognizes the intermediation associated with the task. Then, based on the intermediation observation, the VLM can further generate the decomposed sub-task sequence. The sub-task is

defined as an *(Action, Ending)* pair. In the following process, the robot sequentially executes each sub-task and finally completes the entire task.

**Task Execution:** We extensively explore the perception abilities of VLM to aid in fine-grained trajectory guidance and judgment of sub-task states. As shown in Fig. 3, for each sub-task, the VLM first judge whether the sub-task is finished based on the *Ending*. If it is unfinished, then a specific skill will be selected by the VLM's language instruction based on *Action*. The system executes the skill until a "done" signal is thrown. Meanwhile, the VLM judges the sub-task state again to complete the closed-loop feedback. The system will only be permitted to conduct the next sub-task if the current sub-task is finished. It is worth noting that one sub-task may take several processes to execute. For example, if a terrain is not completely within the field of view, several adjustments are required to do center alignment. This closed-loop design and dual insurance mechanism fully leverage sensor input from the quadruped robot, enhancing the robustness and safety of our system.

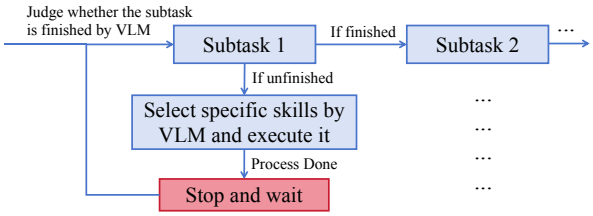


Fig. 3. Closed-loop sub-task execution process.

**Specific Skills with Auxiliary Modules:** For different moving skills, a sub-goal is required relative to the current position. For the "Move facing" skill, 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 $\{X, Y, Z\} = \{\frac{(i-u_0)\cdot d_{ij}}{f_x}, \frac{(j-v_0)\cdot d_{ij}}{f_y}, d_{ij}\}$, and output an accurate sub-goal based on that, as shown in Fig. 4. For the "Move across" skill, the sub-goal is at the same horizontal line as the final goal. For "Move freely to goal", the sub-goal is the same as the final goal relative to the current position. Integrated with the localization module, the auxiliary module refines the trajectory and computes the velocity command by PD control similar to ViNT[36]. The "done" signal is thrown when the distance between the robot and the sub-goal is less than 0.1m.

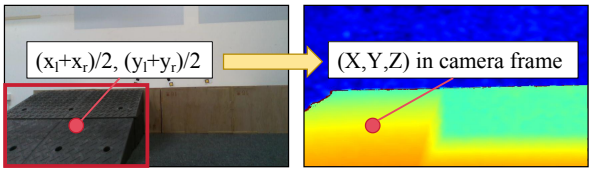


Fig. 4. Illustration of the trajectory refinement module combined with depth image.

For the "Climb across" skill, the generalizable locomotion control policy consistently classifies whether the robot is on the plane or is crossing a terrain, which is discussed in detail in Section 3.3. If the terrain estimator identifies sudden changes from terrain to plane, the "done" signal is thrown.

### 3.3 Low-level locomotion control policy

Our policy enables the quadruped robot to track expected linear and angular velocities over complex terrains. The reinforcement learning policy only trains on proprioception, without any additional external perception such as depth camera and Lidar.

**Oracle Policy Training:** In the first training step, we train an Oracle policy. All information serves as policy observation, including proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$, privileged state $\boldsymbol{s}_t \in \mathbb{R}^4$, and terrain information $\boldsymbol{t}_t \in \mathbb{R}^{187}$. The terrain information is first encoded by terrain encoder $E_t$ into terrain latent $\boldsymbol{t}_{l_t} \in \mathbb{R}^{32}$. Then it is concatenated with $\boldsymbol{s}_t \in \mathbb{R}^4$ into full latent state $\boldsymbol{l}_t \in \mathbb{R}^{36}$. This accelerates the training convergence and enhances the training stability. The input of the low-level network $E_{low}$ $\boldsymbol{o} \in \mathbb{R}^{81}$ is composed of proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$ and $\boldsymbol{l}_t \in \mathbb{R}^{36}$. The actor outputs

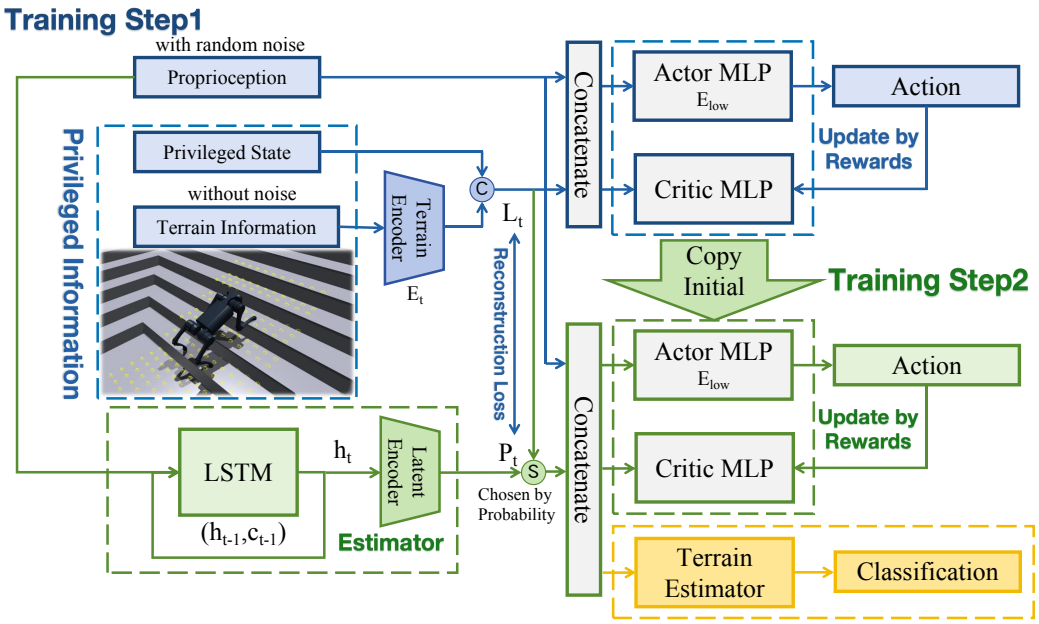


Fig. 5. Overview of the low-level locomotion control policy training process. In the first step, we train an oracle policy using proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$, privileged state $\boldsymbol{s}_t \in \mathbb{R}^4$, and terrain information $\boldsymbol{t}_t \in \mathbb{R}^{187}$. Then in the second step, we use the probability annealing selection (PAS) method to train the final actor network, which only uses proprioception as input. After the policy training process is finished, we exclusively train a terrain estimator to classify whether the robot is on the plane or is climbing the terrain.

the desired joint positions $\boldsymbol{a}_t \in \mathbb{R}^{12}$. The critic is also an MLP, but it directly uses the concatenation of three different types of information $\boldsymbol{o}_c \in \mathbb{R}^{236}$. Due to the use of privileged information, the quadruped robot can quickly and effectively learn locomotion skills on various complex terrains. While training the Oracle policy, we also train a state estimator network $E_e$ concurrently. Mean Squared Error (MSE) loss is used to reconstruct the latent of privileged information $\boldsymbol{l}_t \in \mathbb{R}^{36}$.

**Partial Observation Policy Training:** In the second training step, we train the estimator network and the low-level MLP network jointly using the **P**robability **A**nnealing **S**election (PAS) method. The input of the estimator is proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$, after passing through the LSTM network, the output is then fed into the MLP encoder to get the latent state prediction $\boldsymbol{p}_t \in \mathbb{R}^{36}$. At the beginning of the second training step, the estimator and low-level MLP are copied to initialize from the first training step. Then, the loss of reinforcement learning is utilized to simultaneously optimize both the estimator and the low-level MLP. Then, the (PAS) method is used. Specifically, at the beginning, the latent state vectors input to the lower-level MLP are mostly real values $\boldsymbol{l}_t$, with a small portion of predicted values $\boldsymbol{p}_t$. As the training iterations increase, the probability of selecting real values gradually decreases, and the probability of selecting predicted values gradually increases. Ultimately, only predicted values are used. Specifically,

$$\boldsymbol{p}_t = E_e(\boldsymbol{o}_t^p), \tag{1}$$

$$\boldsymbol{i}_t = \text{Probability Selection } (\boldsymbol{p}_t, \boldsymbol{l}_t), \tag{2}$$

$$\boldsymbol{a}_t = E_{low}(\boldsymbol{i}_t, \boldsymbol{o}), \tag{3}$$

$$\text{Probability } \boldsymbol{P}_t = \boldsymbol{\alpha}^{iteration}. \tag{4}$$

To ensure consistency in the same continuous action, the probability selection is based on the number of robots. Our training method ensures the stability of the training process and enhances the performance of the final policy. Training details including problem definition, reward function, termination

conditions, terrain curriculum, dynamic randomization, network architecture, and hyper-parameters are shown in the Appendix A.

# 4 Experimental Results

## 4.1 Experiment setup

We deploy our method on Unitree A1 quadruped robot with NVIDIA Jetson Xavier NX as the onboard computer. Also, we use a laptop and a GPU server as the computation platform. The low-level locomotion control policy runs on onboard Xavier NX at 50 Hz. More details can be found in appendix. B

## 4.2 Results of high-level reasoning

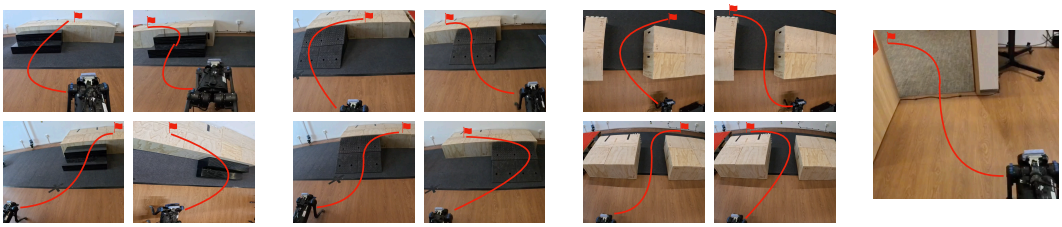


Fig. 6. Illustration of our real-world 3D navigation experiment settings. There are four different directions for routes across stairs, ramps, and gaps.

**Indoor experiments:** To evaluate our navigation system, experiments are conducted based on versatile routes in the real world, shown in Fig 6. We test the robustness on different terrains including stairs, ramps, gaps, and doors. For every terrain, the goals are in various directions. 20 trials are conducted for each situation and we record the success rate of the whole process, only crossing terrains and stable localization, which means that the localization module always works accurately, respectively as the evaluating metrics. We compare our results with three baselines: naive LSTM network, ViNT[36], and NoMaD[55]. We record a real-world dataset containing 50 trajectories including routes on the plain, across stairs, and through ramps, with a frame rate of 15Hz and length of about 10 seconds. A naive baseline is implemented, composed of a CNN image encoder, an LSTM backbone, and an MLP decoder. We train this baseline on the dataset for 500 epochs using MSE loss. We also collect the topological maps and the goal images in the real world for ViNT[36] to simulate our goal pursing task. We implement NoMaD[55] for terrain crossing task using its exploration function. The results are shown in Table 1.

Table 1: The success rate of high-level navigation in versatile real-world experiments. Gray ones indicate the success rates of crossing terrain subtask.

| Intermediation | Overall | Stable Loc | NoMaD | LSTM | Across Terrains | ViNT |
|---|---|---|---|---|---|---|
| Stair | 60% | **88%** | 0% | 0% | 70% | 0% |
| Ramp | 25% | **67%** | 0% | 0% | 50% | 0% |
| Gap | 45% | **94%** | 20% | 0% | 80% | 0% |
| Door | 30% | **63%** | 70% | 0% | 50% | 0% |

We observe relatively good results for versatile intermediations, especially for stairs, which demonstrates the effectiveness and robustness of our navigation system. The task definition is much harder than NoMaD and ViNT since the starting point is closer to the intermediations and it requires faster planning and adjustment to achieve the correct position and direction, which blocks the baselines from reacting well. Also, all three baselines lack the 3D reasoning and planning capability as CAS

holds. It is worth noting that our overall success rate relies heavily on the accuracy of our localization module. Our ablations on crossing intermediation and the ideal stable localization module show a large margin of improvement in all of the situations. This encourages our valuable system design and great cooperation of generalizable control policy and other modules. Meanwhile, we point out that most of the localization errors occur when the input images are blurred due to the high-dynamic motion, especially for the ramps which makes the quadruped robot lean upwards.

VLM actively participates in every step of the 3D navigation task and demonstrates its strong power of common sense reasoning and motion estimation. While several challenges, such as transforming ego-view 2D image information into 3D environment interactions, arise in the application of VLM, we manage to bridge the gap by designing system auxiliary modules that leverage rich information from other robot sensors. The entire system is capable of generalizing to multiple complex 3D terrains and diverse environments. Additionally, it can be robustly embedded in real-world quadruped robots.

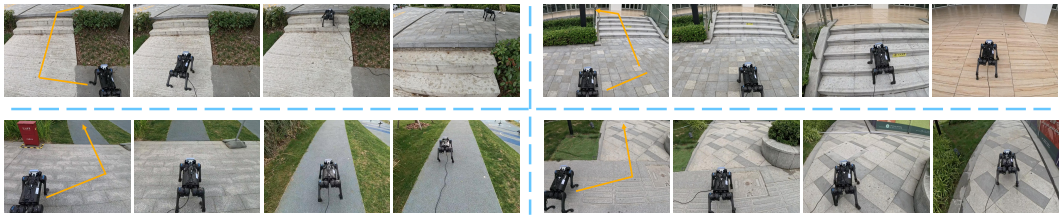

Fig. 7. Details of our outdoor 3D navigation experiment results. We conduct the experiment on diverse terrains (the upper is stairs and the lower is ramps) and routes.

**Outdoor experiments:** Additionally, we test our system outdoors to show the generalizing ability. As shown in Fig 7, we spot that our framework can be easily extended to the wild environment and the versatile 3D terrain conditions can be covered by our robust perception, planning, and control pipeline. More details can be referred to in our appended demo videos.

### 4.3 Results of low-level locomotion

To further evaluate our proposed PAS method, we conduct extra experiments on the lower-level locomotion control. We select a variety of challenging terrains, including uneven ground, stairs, ramps, and unseen terrains, to test the success rate and speed tracking ratio. Both metrics are the higher the better.

**Simulation Results:** For success rate, we conduct 4,096 independent experiments for each type of terrain. If the robot reaches the edge of the terrain or is alive for over 20 seconds, it is marked as a success. The velocity tracking ratio shows the performance for tracking velocity commands, and we also conducted 4,096 experiments for each terrain.

Table 2: Comparison experiment results in simulation

| **Metric** | **PAS(ours)** | **RMA**[1] | **IL**[50] | **Blind** | **Concurrent**[56] |
|---|---|---|---|---|---|
| Success rate (Avg) | **85.31%** | 49.30% | 58.05% | 77.34% | 70.40% |
| Lin vel tracking ratio (Avg) | **0.8790** | 0.6848 | 0.8029 | 0.8555 | 0.8330 |
| Ang vel tracking ratio (Avg) | **0.7815** | 0.5895 | 0.7370 | 0.7627 | 0.7349 |

First, we compare our method with several previous baselines using only proprioception. The details of methods used for comparison can be found in appendix. C.2. Results show that our method significantly outperforms previous methods. Both RMA (latent space) and the "teacher-student" framework (action space) only utilize imitation learning in the second step of training without reward feedback, and they perform worse than those of pure blind training and concurrent training. Pure blind training, which does not use privileged information for training, performs relatively well on

flat ground and simpler terrains. But its performance drops rapidly once the terrain becomes more complex. Concurrent training only uses partial perceptual information, so it is difficult to estimate an accurate latent state.

Table 3: Ablation experiment results in simulation

| Metric | Exp 0.9998 | Exp 0.9995 | No anneal | Cosine | Linear |
|---|---|---|---|---|---|
| Success rate (Avg) | **85.31%** | 83.60% | 83.33% | 83.26% | 84.00% |
| Lin vel tracking ratio (Avg) | **0.8790** | 0.8710 | 0.8687 | 0.8730 | 0.8715 |
| Ang vel tracking ratio (Avg) | **0.7815** | 0.7741 | 0.7733 | 0.7466 | 0.7660 |

We also conduct ablation experiments for different annealing settings. Results show that the exponential annealing with the base of 0.9998 performs the best in most scenarios. Exponential annealing with an annealing probability of 0.9995 may be too rapid. No annealing results in the robot being unable to adapt at the start of learning, necessitating a period of re-exploration, during which a significant amount of the oracle policy's capabilities are lost. Furthermore, we find that although cosine annealing intuitively follows a slow-fast pattern, its performance is the worst, especially in terms of angular velocity tracking ratio, which may be due to a rapid collapse of the network at a certain point.

**Real-world Result:** We conduct experiments across a series of complex terrains. For experiments on each type of terrain, we continuously conduct 20 trials. For each trial, if the robot could start from the beginning, and reach the end without falling or getting stuck, it is considered a success. As the results shown in Table 4, our robot can pass through complex terrains blindly and is highly competitive with previous methods.

Table 4: Success rate results in real-world experiments

| Terrain type | PAS(ours) | RMA | IL | Built-in MPC |
|---|---|---|---|---|
| Stair | **100%** | 75% | 80% | 0% |
| Ramp | **90%** | 70% | 80% | 55% |
| Rubble | **95%** | 70% | 75% | 0% |
| Grassland | **100%** | 80% | 95% | 60% |
| Unseen obstacle | **95%** | 65% | 80% | 0% |

## 5 Conclusion

We present a general quadruped robot navigation system with the vision-language foundation model. The high-level system utilizes task decomposition and closed-loop sub-task execution mechanisms to boost the 3D scene understanding and motion planning. The low-level control policy PAS is designed as a novel reinforcement learning method that efficiently learns a partial policy from the oracle policy and facilitates the quadruped robot crossing versatile 3D terrains. Our extensive experiments in both simulator and real-world demonstrate the effectiveness and robustness of the whole system as well as the locomotion control policy. However, due to the high-frequency vibrations of the quadruped robot bringing errors to IMU and blurring the image, the current commonly used SLAM method is not as stable as we expect, and it damages the reliability of our whole system. In addition, our system does not integrate memory like the topological map used in ViNT[36], which may tackle the problem brought by the ego-view perception task. In the future, we believe that better perception and localization approaches can benefit our pipeline. Topological map or semantic map can be integrated with VLM to help investigate the general navigation agents.

## References


[1] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[2] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

[3] X. Li, M. Zhang, Y. Geng, H. Geng, Y. Long, Y. Shen, R. Zhang, J. Liu, and H. Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. *arXiv preprint arXiv:2312.16217*, 2023.

[4] F. Liu, K. Fang, P. Abbeel, and S. Levine. Moka: Open-vocabulary robotic manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024.

[5] H. Huang, F. Lin, Y. Hu, S. Wang, and Y. Gao. Copa: General robotic manipulation through spatial constraints of parts with foundation models. *arXiv preprint arXiv:2403.08248*, 2024.

[6] I. Kapelyukh, Y. Ren, I. Alzugaray, and E. Johns. Dream2Real: Zero-shot 3D object rearrangement with vision-language models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[7] I. Poulakakis, E. Papadopoulos, and M. Buehler. On the stability of the passive dynamics of quadrupedal running with a bounding gait. *The International Journal of Robotics Research*, 25(7):669–687, 2006.

[8] J. Pratt, P. Dilworth, and G. Pratt. Virtual model control of a bipedal walking robot. In *Proceedings of international conference on robotics and automation*, volume 1, pages 193–198. IEEE, 1997.

[9] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[10] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9. IEEE, 2018.

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[14] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[15] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

[16] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning, 2023.

[17] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, Z. Zhao, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023.

[18] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR, 2023.

[19] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[20] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler. Open-vocabulary queryable scene representations for real world planning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11509–11522. IEEE, 2023.

[21] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.

[22] M. Xu, P. Huang, W. Yu, S. Liu, X. Zhang, Y. Niu, T. Zhang, F. Xia, J. Tan, and D. Zhao. Creative robot tool use with large language models. *arXiv preprint arXiv:2310.13065*, 2023.

[23] Y. Ouyang, J. Li, Y. Li, Z. Li, C. Yu, K. Sreenath, and Y. Wu. Long-horizon locomotion and manipulation on a quadrupedal robot with large language models. *arXiv preprint arXiv:2404.05291*, 2024.

[24] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[25] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[26] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[27] M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid. Learning reward functions for robotic manipulation by observing humans. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5006–5012. IEEE, 2023.

[28] P. Mahmoudieh, D. Pathak, and T. Darrell. Zero-shot reward specification via grounded natural language. In *International Conference on Machine Learning*, pages 14743–14752. PMLR, 2022.

[29] Y. J. Ma, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pages 23301–23320. PMLR, 2023.

[30] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, S. Kirmani, B. Zitkovich, F. Xia, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.

[31] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023.

[32] N. H. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[33] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023.

[34] X. Tian, J. Gu, B. Li, Y. Liu, C. Hu, Y. Wang, K. Zhan, P. Jia, X. Lang, and H. Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

[35] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada. Saytap: Language to quadrupedal locomotion. *arXiv preprint arXiv:2306.07580*, 2023.

[36] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[38] A. Lykov, M. Litvinov, M. Konenkov, R. Prochii, N. Burtsev, A. A. Abdulkarim, A. Bazhenov, V. Berman, and D. Tsetserukou. Cognitivedog: Large multimodal model based system to translate vision and language into action of quadruped robot. In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 712–716, 2024.

[39] W. Song, H. Zhao, P. Ding, C. Cui, S. Lyu, Y. Fan, and D. Wang. Germ: A generalist robotic model with mixture-of-experts for quadruped robot. *arXiv preprint arXiv:2403.13358*, 2024.

[40] M. Mistry, J. Buchli, and S. Schaal. Inverse dynamics control of floating base systems using orthogonal decomposition. In *2010 IEEE international conference on robotics and automation*, pages 3406–3412. IEEE, 2010.

[41] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter. A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, 6 (3):4688–4695, 2021.

[42] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Transactions on Robotics*, 2023.

[43] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[44] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[45] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023.

[46] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

[47] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023.

[48] A. Agrawal, S. Chen, A. Rai, and K. Sreenath. Vision-aided dynamic quadrupedal locomotion on discrete terrain using motion libraries. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4708–4714. IEEE, 2022.

[49] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024.

[50] J. Wu, G. Xin, C. Qi, and Y. Xue. Learning robust and agile legged locomotion using adversarial motion priors. *IEEE Robotics and Automation Letters*, 2023.

[51] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023.

[52] I. M. A. Nahrendra, B. Yu, and H. Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084. IEEE, 2023.

[53] J. Long, Z. Wang, Q. Li, J. Gao, L. Cao, and J. Pang. Hybrid internal model: A simple and efficient learner for agile legged locomotion. *arXiv preprint arXiv:2312.11460*, 2023.

[54] J. Long, W. Yu, Q. Li, Z. Wang, D. Lin, and J. Pang. Learning h-infinity locomotion control. *arXiv preprint arXiv:2404.14405*, 2024.

[55] A. Sridhar, D. Shah, C. Glossop, and S. Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. *arXiv pre-print*, 2023. URL https://arxiv.org/abs/2310.07896.

[56] G. Ji, J. Mun, H. Kim, and J. Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2):4630–4637, 2022.

[57] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[58] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[59] N. Heess, D. Tb, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[60] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[61] H. Liu, C. Li, Y. Li, B. Li, Y. Zhang, S. Shen, and Y. J. Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

[62] T. Qin, J. Pan, S. Cao, and S. Shen. A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

# 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 | $-\lVert\dot{\mathbf{q}}\rVert_2 \cdot \lVert\tau\rVert_2$ | $1 \times 10^{-6}$ |
| | Joint vel | $-\lVert\dot{\mathbf{q}}\rVert_2$ | 0.002 |
| | Joint acc | $-\lVert\ddot{\mathbf{q}}\rVert_2$ | $2 \times 10^{-6}$ |
| | Ang vel Stability | $-(\lVert\omega_{t,x}\rVert_2 + \lVert\omega_{t,y}\rVert_2)$ | 0.2 |
| **Style** | Feet in air | $\sum_{i=0}^{3}(\mathbf{t}_{air,i} - 0.3) + 10 \cdot \max(0.5 - \mathbf{t}_{air,i}, 0)$ | 0.05 |
| | Balance | $-\lVert F_{feet,0} + F_{feet,2} - F_{feet,1} - F_{feet,3}\rVert_2$ | $2 \times 10^{-5}$ |

### 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 $8 \times 10$ 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

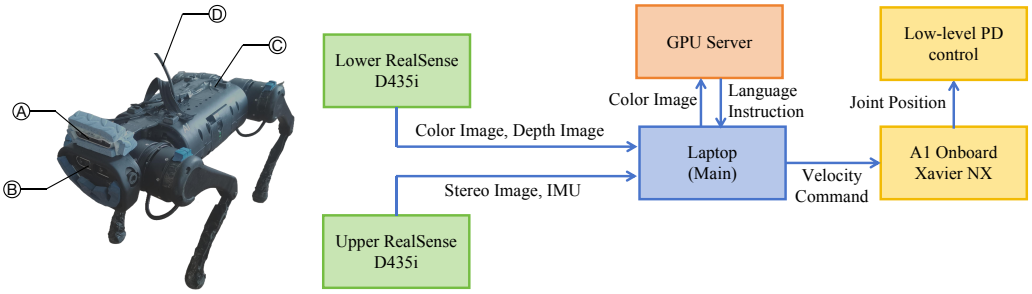


Fig. 8. Robot setup for experiment. A: Upper RealSense D435i camera. B: Lower RealSense D435i camera. C: Unitree A1 quadruped robot equipped with NVIDIA Jetson Xavier NX. D: Network cable for communication between quadruped robot and laptop. Based on ROS, we set up our communication system. We use a 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

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

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

### 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.

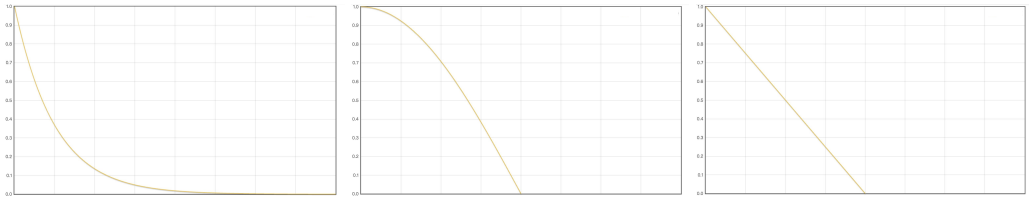


Fig. 9. Annealing schedule of different settings. Exponential annealing is fast initially and then slows down, cosine annealing is slow initially and then speeds up, and linear annealing is uniform all the process.