SARO: Space-Aware Robot System for Terrain Crossing *via* Vision-Language Model

Shaoting Zhu*[1,5], Derun Li*[2,5], Linzhan Mou[4,5], Yong Liu[3], Ningyi Xu[2], Hang Zhao†[1,5]

Fig. 1: SARO achieves space-aware navigation capability for 3D terrain crossing. Our system utilizes the reasoning and motion planning ability of the VLM model, with a design of task decomposition and a closed-loop sub-task execution module. While traditional navigation approaches (yellow) fail in reasoning 3D environment, our system (red) guides the quadruped robot to cross the accessible intermediation towards the goal.

***Abstract*— The application of vision-language models (VLMs) has achieved impressive success in various robotics tasks. However, there are few explorations for these foundation models used in quadruped robot navigation through terrains in 3D environments. In this work, we introduce SARO (Space-Aware Robot System for Terrain Crossing), an innovative system composed of a high-level reasoning module, a closed-loop sub-task execution module, and a low-level control policy. It enables the robot to navigate across 3D terrains and reach the goal position. For high-level reasoning and execution, 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 Probability Annealing Selection (PAS) method to effectively train a control policy by reinforcement learning. Numerous experiments show that our whole system can accurately and robustly navigate across several 3D terrains, and its generalization ability ensures the applications in diverse indoor and outdoor scenarios and terrains. *Appendix* and *Videos* can be found in project page: https://saro-vlm.github.io/.**

I. 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. It 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.

[1]IIIS, Tsinghua University, Beijing, China
[2]SEIEE, Shanghai Jiao Tong University, Shanghai, China
[3]CSE, Zhejiang University, Hangzhou, China
[4]GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA
[5]Shanghai Qi Zhi Institute, Shanghai, China
* These authors contributed equally to this work.
† Corresponding author. E-mail: hangzhao@mail.tsinghua.edu.cn

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]–[9]. 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 and travel in the 3D world?* In this work, we design a system called SARO, composed of a high-level reasoning module, a close-loop sub-task execute module, and a low-level control policy. The system enhances the 3D reasoning, motion planning, and locomotion ability of the robots. Our design uses zero-shot VLM common sense reasoning to overcome the lack of training data and utilizes the closed-loop sub-task execution to invest a memory-free mechanism to transfer agents stage by stage in the navigation process.

Besides, the low-level locomotion control policy needs to be adaptable to different terrains and robust against diverse environments. Traditional control methods such as SLIP [10], VMC [11], MPC [12], [13] can handle some specific terrain tasks, but they have poor robustness against complex real-world situations. Adaptation learning [1] and teacher-student framework [14] 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 Oracle policy transfer problem caused by mimic learning. It comprehensively learns the ability to cross various types of real-world 3D terrains.

We test our method across several different categories of terrains and intermediations. In addition, we showcase

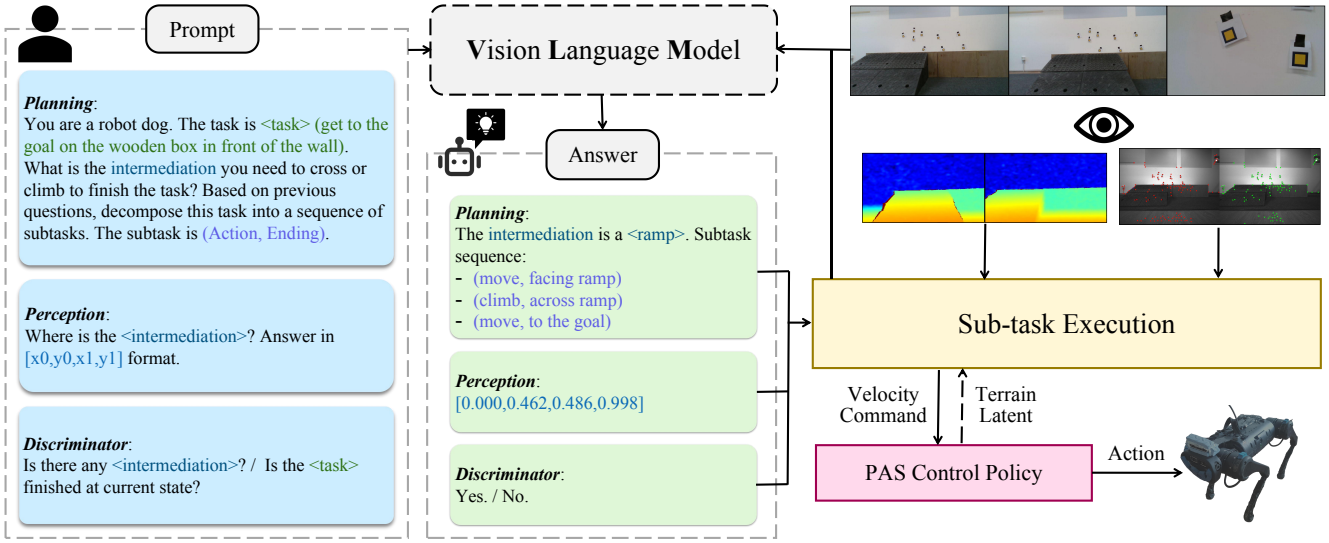


Fig. 2: Overview of the SARO system. The robot needs to complete a goal-tracking task while autonomously navigating through a 3D terrain. The pretrained vision-language foundation model (VLM) takes as input RGB images and prompts querying the 3D environment perception to decompose the task into sub-tasks. After that, the system executes these sub-tasks in a closed-loop taking advantage of VLM discriminator double-check. The well-designed sub-task execution module connects the high-level VLM and the low-level control policy and receives depth image and stereo image information to help localization.

the capacity of our low-level locomotion policy in extra experiments. Since the restricted view range of the front view camera input, we define the task as goal-tracking with a simple setting of only one intermediation chosen from "stair", "ramp", "gap", and "door" for each task in experiments. Our experiment results demonstrate the generalization and common-sense visual motion reasoning abilities of SARO in quadruped robot 3D navigation. Our method exhibits generalization ability and robustness in real-world scenarios, as demonstrated in our real-world videos.

In summary, our contributions are as follows:

1) We innovatively present SARO, a system that is composed of a high-level reasoning module, a closed-loop sub-task execution module, and a robust low-level control policy. This novel system enhances ego-view 3D 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) We conduct experiments across extensive terrains. The results demonstrate our system's ability to complete our specially defined goal-tracking task across 3D terrains.

II. RELATED WORKS

A. Foundation Models in Robotics

In recent years, foundation models [15]–[19], 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 [20]–[32]. Several works [23], [33]–[35] have utilized open-vocabulary pretrained models for robotic tasks, Others [3]–[5], [36], [37] have employed powerful VLMs, such as GPT-4V, for robotics tasks. Moreover, some researchers have attempted to apply foundation models to quadruped robots. Saytap [38] uses large language models (LLMs) to translate natural language commands into foot contact patterns for quadrupedal robots. ViNT [39] trains a universal policy from a large-scale visual navigation dataset using the Transformer [40]. CognitiveDog [41] integrates a Large Multi-modal Model (LMM) with a quadruped robot. GeRM [42] trains a generalist model for quadruped robots in vision-language tasks. QuadrupedGPT [43] and Commonsense [9] utilize large models for movement in simple scenes. Nevertheless, all these methods are only suitable for tasks on planar surfaces and do not fully utilize the 3D terrain capabilities of quadruped robots.

B. Locomotion Control of Quadruped Robots

Traditional locomotion control methods for quadruped robots [10], [11], [44]–[46] 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 [1], [14], [47]. They utilize the privileged training paradigm to train quadruped robots without extra sensors. Also, some work [48]–[52] integrate proprioceptive and exteroceptive states to achieve agile locomotion. Mimic learning is frequently used in previous works. Methods of adaptation learning [1], [2], [53] and teacher-student framework learning [14], [54], [55] 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 locomotion efficiency. DayDreamer [56] learns a "world model" to synthesize infinite interactions, while DreamWaQ [57] implicitly infers terrain properties and adapts its gait accordingly by learning a VAE model. Traditional control

methods are combined with deep reinforcement learning [58]–[60] 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 Oracle policy transfer problem without mimic learning, while fully taking advantage of privileged information in simulation with two-stage training.

III. METHOD

A. Task Definition

The task is defined as a goal-tracking task for the quadruped robot to autonomously navigate through a 3D environment with various terrains. A terrain $\mathcal{T}$ is composed of two platforms $\mathcal{P}_1$, $\mathcal{P}_2$ and one intermediation $\mathcal{I}$ connecting two platforms in 3D space. In the beginning, the robot is located on $\mathcal{P}_1$, and the task is to reach a specified goal $\mathcal{G}$ on $\mathcal{P}_2$ defined as (x, y, z, yaw) relative to the robot's starting position, combined with a language description $\mathcal{L}$ of the goal. The robot needs to cross intermediation $\mathcal{I}$ to reach the goal on platform $\mathcal{P}_2$. The 3D intermediation in this work includes "stairs", "ramps", "gaps", and "doors", which the robot does not take extreme action to come across. The quadruped robot can only access the sensors onboard, including proprioception, ego-view RGB image, and depth image. In summary, the tasks can be represented as:

$$\textit{Find the way and navigate through: } \{\mathcal{P}_1 \rightarrow \mathcal{I} \rightarrow \mathcal{P}_2\}$$
$$\textit{under the condition of } \mathcal{G} \textit{ and } \mathcal{L}$$

As an example, in Fig. 1, the robot begins on the floor ($\mathcal{P}_1$) in front of a higher platform ($\mathcal{P}_2$) and needs to get the goal $\mathcal{G}$ located on the platform. The goal point is "$(3.0m, 0.0m, 0.4m, 0.0rad)$". The language instruction $\mathcal{L}$ is "getting to the goal on the wooden box in front of the wall". We list all of our experiment scenario details in Section IV.

B. High-level Reasoning and Task Execution

Task Decomposition: Our system works as a state machine and decomposes the multi-step navigation into a sub-task sequence composed of movement actions and the ending point by prompting the VLM. The prompt is based on the task that defines the robot's ability and instructs it to cross the terrain. 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 $\mathcal{I}$ associated with the task instruction $\mathcal{L}$. After that, the VLM can further generate the decomposed sub-task sequence. 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"]. The full prompts and examples are shown in the *Appendix A.1*.

Sub-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 discriminator first judges whether the sub-task is finished based on the *Ending*. If it is unfinished, a velocity command will be sent to low-level policy based on the *Action* and VLM's language instruction. The predefined execution workflow determines how to complete this *Action* until the *Ending* point, which is expanded in detail in the *Appendix A.2*. Both sub-task execution workflow and VLM discriminator can judge the *Ending* point, but the system will only be permitted to conduct the next sub-task if the VLM discriminator outputs the ending signal, which we call double-check. It is worth noting that one sub-task may take several processes to execute because the low-level workflow may not predict an accurate *Ending* point. For example, if the intermediation is not completely within the field of view, several adjustments are required to do center alignment. This closed-loop module and double-check mechanism fully leverage sensor input from the quadruped robot, enhancing the robustness and safety of our system.

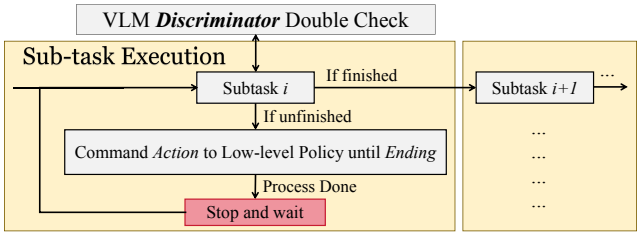


Fig. 3: After task decomposition, the system executes the sub-tasks one by one. The double-check closed-loop module improves the robustness of the system.

C. Low-level Locomotion Control Policy

We enable the quadruped robot to track expected linear and angular velocities over various terrains with only one policy. 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 the 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 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 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

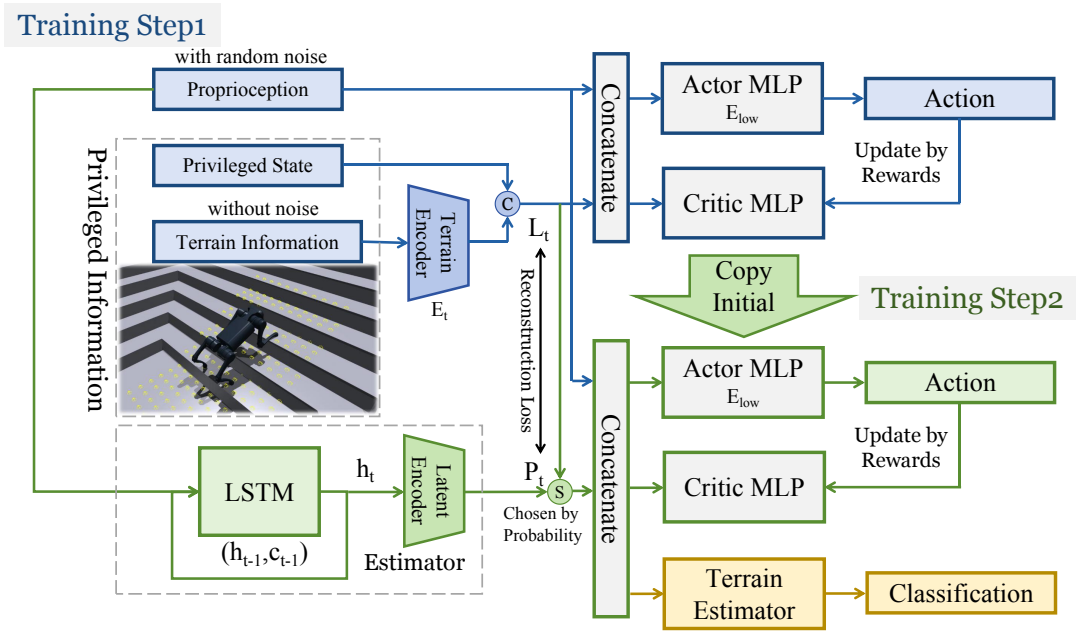


Fig. 4: Overview of the low-level locomotion control policy. In the first training step, we train an oracle policy using proprioception $\boldsymbol{p}_t \in \mathbb{R}^{45}$, privileged state $\boldsymbol{s}_t \in \mathbb{R}^4$, and terrain information $\boldsymbol{t}_t \in \mathbb{R}^{187}$. In the second training 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 intermediation.

state prediction $\boldsymbol{p}_t \in \mathbb{R}^{36}$. At the beginning of the second training step, the estimator and the 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,

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

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

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

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

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 B*.

IV. EXPERIMENTS

A. 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.

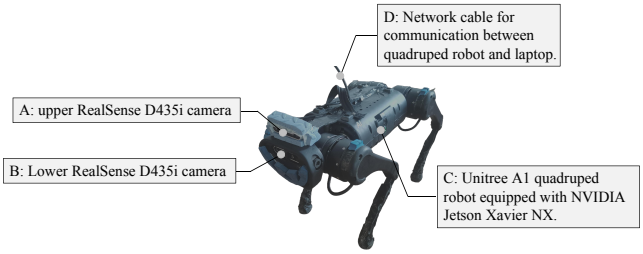


Fig. 5: Robot setup for experiment.

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. The upper one takes stereo images at a resolution of 640×480. The lower one takes an aligned RGB-D image at a resolution of 848×480. We use LLaVA-34B [61] as both vision language model and vision language model discriminator, VINS-Fusion [62] as the VIO algorithm. Detailed robot setup for experiment is shown in Fig. 5. Based on ROS, we set up our communication system as shown in Fig. 6. 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 RGB images and language instructions with the laptop. The onboard Xavier NX receives the velocity commands by the main program from the laptop

TABLE I: The success rate of high-level navigation in versatile real-world experiments. Gray ones indicate the success rates of only crossing terrain sub-tasks (no need to get stop at the goal).

Intermediation	Overall	Stable Loc	w/o Closed-Loop	NoMaD	LSTM	Across Terrains	ViNT
Stair	60%	**88%**	10%	0%	0%	70%	0%
Ramp	25%	**67%**	10%	0%	0%	50%	0%
Gap	45%	**94%**	20%	20%	0%	80%	0%
Door	30%	**63%**	15%	70%	0%	50%	0%

through ROS messages and runs the low-level control policy to predict desired joint positions for PD control. 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.

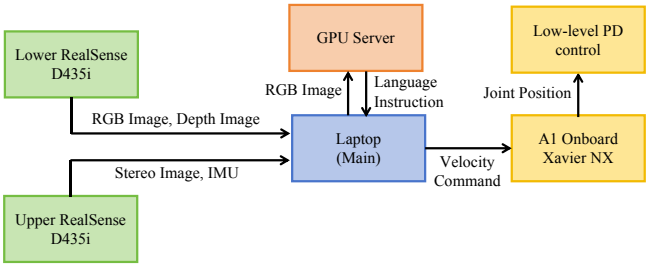


Fig. 6: Communication system setup of robot deployment.

B. Results of High-level Reasoning

Indoor Experiments: To evaluate our crossing system, experiments are conducted based on versatile routes in the real world, shown in Fig 7. We test the robustness on different terrains including stairs, ramps, gaps, and doors. For every terrain, the goals are at 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 [39], and NoMaD [63]. 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 [39] to simulate our goal-pursing task. We implement NoMaD [63] for terrain crossing tasks using its exploration function. The results are shown in Table I.

We observe relatively good results for versatile intermediations, especially for stairs, which demonstrates the effectiveness and robustness of our crossing 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 SARO 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 sub-task execution modules that leverage rich information from other robot sensors. The entire system is capable of generalizing to multiple 3D terrains and diverse environments. Additionally, it can be robustly embedded in real-world quadruped robots.

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.

C. 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. 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.

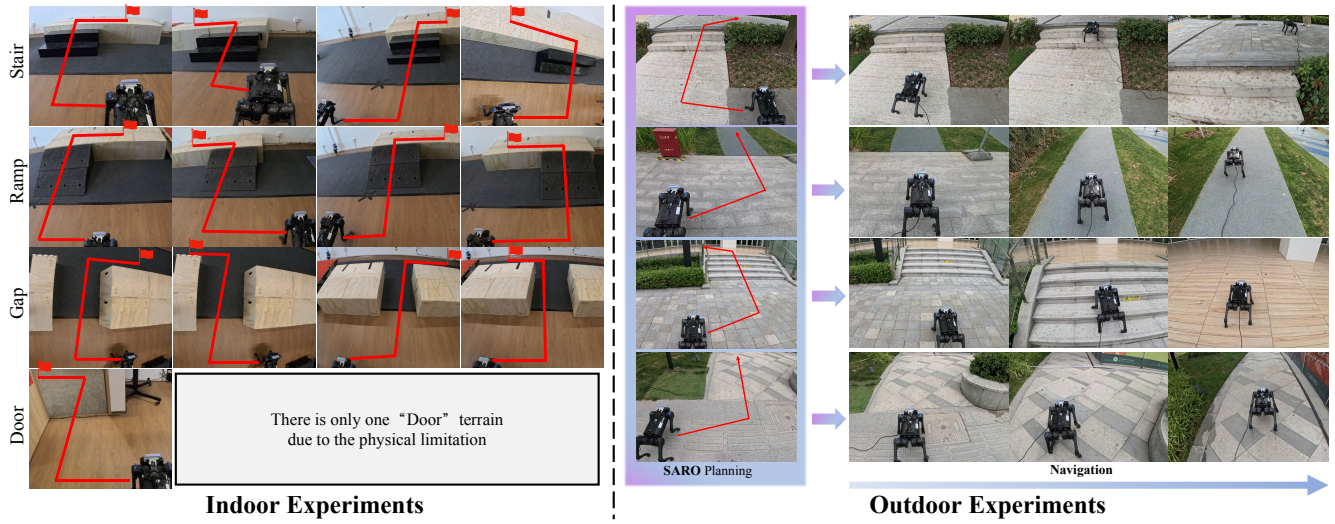


Fig. 7: Indoor and outdoor 3D navigation experiment results on diverse terrains.

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 II: Comparison experiment results in simulation

Metric	PAS(ours)	RMA [1]	IL [55]	Blind	Concurrent [64]
SR (Avg)	**85.31%**	49.30%	58.05%	77.34%	70.40%
L vel (Avg)	**0.8790**	0.6848	0.8029	0.8555	0.8330
A vel (Avg)	**0.7815**	0.5895	0.7370	0.7627	0.7349

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.

TABLE III: Ablation experiment results in simulation

Metric	Exp 0.9998	Exp 0.9995	No anneal	Cosine	Linear
SR (Avg)	**85.31%**	83.60%	83.33%	83.26%	84.00%
L vel (Avg)	**0.8790**	0.8710	0.8687	0.8730	0.8715
A vel (Avg)	**0.7815**	0.7741	0.7733	0.7466	0.7660

Real-world Results: We conduct experiments across a series of 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 as a success. As the results shown in Table IV, our robot can pass through terrains blindly and is highly competitive with previous methods.

TABLE IV: 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%

V. CONCLUSION

We present a space-aware robot system (SARO) for vision navigation in 3D environments. The high-level module utilizes task decomposition and closed-loop sub-task execution module 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.

For limitation, 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, we only deploy our system on short-term simple tasks, which is limited by ego-view perception and lack of memory. In the future, we believe that better perception and localization approaches can benefit our pipeline. Topological maps or semantic maps can be integrated with VLM to help investigate more complex tasks.

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.

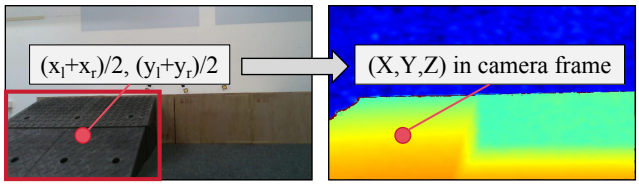


Fig. 8: Illustration of the trajectory refinement module combined with depth image.

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

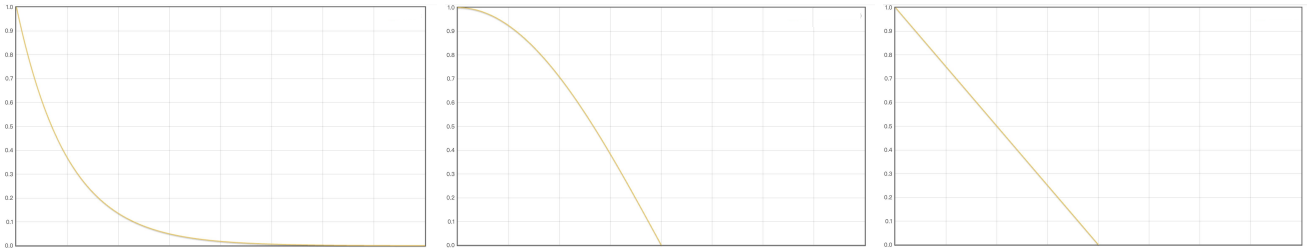
Observation	Gaussian Noise Amplitude	Unit
Linear velocity	0.05	m/s
Angular velocity	0.2	rad/s
Gravity	0.05	m/s^2
Joint position	0.01	rad
Joint velocity	1.5	rad/s

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.

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*. PMLR, 2023, pp. 22–31.

[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] V. Myers, B. C. Zheng, O. Mees, S. Levine, and K. Fang, "Policy adaptation via language optimization: Decomposing tasks for few-shot imitation," *arXiv preprint arXiv:2408.16228*, 2024.

[8] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," *arXiv preprint arXiv:2409.01652*, 2024.

[9] A. S. Chen, A. M. Lessing, A. Tang, G. Chada, L. Smith, S. Levine, and C. Finn, "Commonsense reasoning for legged robot adaptation with vision-language models," *arXiv preprint arXiv:2407.02666*, 2024.

[10] 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*, vol. 25, no. 7, pp. 669–687, 2006.

[11] J. Pratt, P. Dilworth, and G. Pratt, "Virtual model control of a bipedal walking robot," in *Proceedings of international conference on robotics and automation*, vol. 1. IEEE, 1997, pp. 193–198.

[12] 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)*. IEEE, 2018, pp. 2245–2252.

[13] 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)*. IEEE, 2018, pp. 1–9.

[14] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[15] 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.

[16] 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*, vol. 24, no. 240, pp. 1–113, 2023.

[17] 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.

[18] 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*. PMLR, 2022, pp. 12 888–12 900.

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

[20] 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.

[21] 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*. PMLR, 2023, pp. 287–318.

[22] 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.

[23] 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)*. IEEE, 2023, pp. 11 509–11 522.

[24] 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*. PMLR, 2022, pp. 9118–9147.

[25] 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.

[26] 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.

[27] 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)*. IEEE, 2023, pp. 9493–9500.

[28] 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)*. IEEE, 2023, pp. 11 523–11 530.

[29] 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.

[30] 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)*. IEEE, 2023, pp. 5006–5012.

[31] P. Mahmoudieh, D. Pathak, and T. Darrell, "Zero-shot reward specification via grounded natural language," in *International Conference on Machine Learning*. PMLR, 2022, pp. 14 743–14 752.

[32] 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*. PMLR, 2023, pp. 23 301–23 320.

[33] 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.

[34] 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*, 2023, pp. 23 171–23 181.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

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

[41] 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*, 2024, pp. 712–716.

[42] 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.

[43] Y. Wang, Y. Mei, S. Zheng, and Q. Jin, "Quadrupedgpt: Towards a versatile quadruped agent in open-ended worlds," *arXiv preprint arXiv:2406.16578*, 2024.

[44] 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*. IEEE, 2010, pp. 3406–3412.

[45] 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*, vol. 6, no. 3, pp. 4688–4695, 2021.

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

[47] 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*, vol. 4, no. 26, p. eaau5872, 2019.

[48] 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*, vol. 7, no. 62, p. eabk2822, 2022.

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

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

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

[52] S. Luo, S. Li, R. Yu, Z. Wang, J. Wu, and Q. Zhu, "Pie: Parkour with implicit-explicit learning framework for legged robots," *arXiv preprint arXiv:2408.13740*, 2024.

[53] 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)*. IEEE, 2022, pp. 4708–4714.

[54] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024.

[55] 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.

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

[57] 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)*. IEEE, 2023, pp. 5078–5084.

[58] 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.

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

[60] S. Chen, Z. Wan, S. Yan, C. Zhang, W. Zhang, Q. Li, D. Zhang, and F. U. D. Farrukh, "Slr: Learning quadruped locomotion without privileged information," *arXiv preprint arXiv:2406.04835*, 2024.

[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. [Online]. Available: 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.

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

[64] 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*, vol. 7, no. 2, pp. 4630–4637, 2022.

[65] 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.

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

[67] 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.

[68] 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*. PMLR, 2022, pp. 91–100.