Versatile Legged Locomotion Adaptation through Vision-Language Grounding

I Made Aswin Nahrendra, Seunghyun Lee, Dongkyu Lee, and Hyun Myung*, *Senior Member, IEEE*

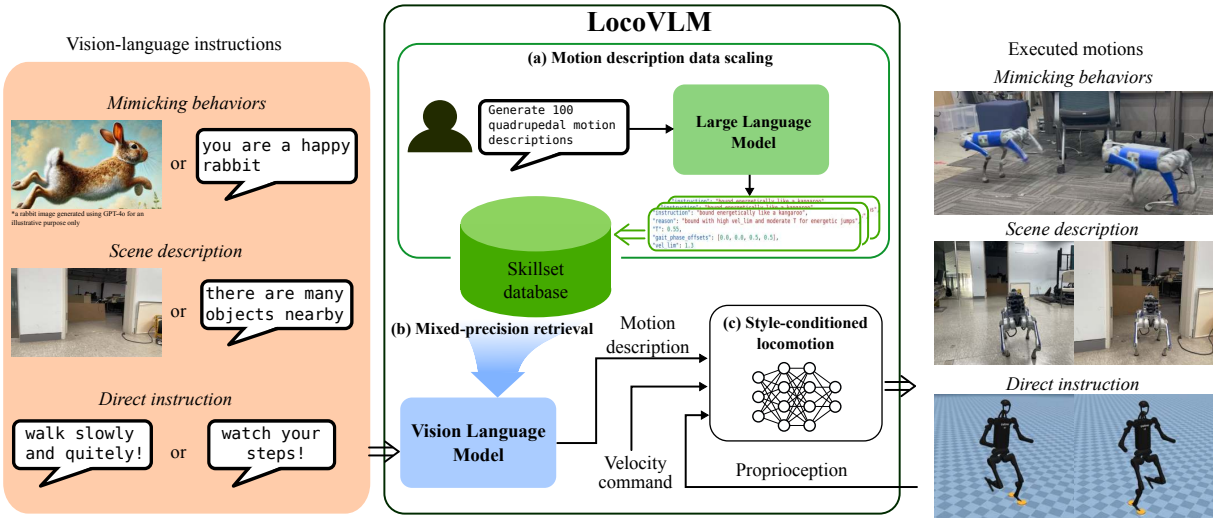

Fig. 1. LocoVLM receives vision-language instructions as its input and hierarchically grounds them into versatile locomotion skills. (a) An LLM is used to scale up motion descriptor data generation and store it into a skill database. (b) During inference, a VLM retrieves the most relevant motion descriptor from the database using the proposed mixed-precision retrieval mechanism to give a style reference to the locomotion controller. (c) Finally, a pre-trained style-conditioned locomotion controller executes the robot's motion to realize the given vision-language instructions.

***Abstract*— Recent advances in legged locomotion learning are still dominated by the utilization of geometric representations of the environment, limiting the robot's capability to respond to higher-level semantics such as human instructions. To address this limitation, we propose a novel approach that integrates high-level commonsense reasoning from foundation models into the process of legged locomotion adaptation. Specifically, our method utilizes a pre-trained large language model to synthesize an instruction-grounded skill database tailored for legged robots. A pre-trained vision-language model is employed to extract high-level environmental semantics and ground them within the skill database, enabling real-time skill advisories for the robot. To facilitate versatile skill control, we train a style-conditioned policy capable of generating diverse and robust locomotion skills with high fidelity to specified styles. To the best of our knowledge, this is the first work to demonstrate real-time adaptation of legged locomotion using high-level reasoning from environmental semantics and instructions with instruction-following accuracy of up to 87% without the need for online query to on-the-cloud foundation models. Demonstration videos are available at `https://locovlm.github.io`.**

This work was supported by the Korea Evaluation Institute of Industrial Technology (KEIT) funded by the Korea Government (MOTIE) under grant No. 20019216, "Development of Mobile Intelligence SW for Autonomous Navigation of Legged Robots in Dynamic and Atypical Environments for Real Application." The students are supported by BK21 FOUR.

The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea (e-mail: {anahrendra, kevin9709, dklee98, hmyung}@kaist.ac.kr).

*Corresponding author: Hyun Myung.

I. Introduction

Legged robots hold immense potential for real-world applications owing to their ability to navigate complex terrains and environments. Over the past decade, they have become increasingly developed for their potential to assist humans in various tasks such as inspection, last-mile delivery, and search-and-rescue missions [1]–[3]. Recent advancements in deep reinforcement learning (RL) have significantly accelerated research in this domain. RL, in particular, has shown promising results in generating robust locomotion policies capable of adapting to diverse terrains and tasks [4]–[9]. A critical challenge in legged locomotion lies in perceiving the environment and acting accordingly while fully utilizing the robot's physical capabilities.

Typically, legged robot locomotion is governed by controllers that rely on geometric representations of the environment. These representations are derived either explicitly from exteroceptive sensors or implicitly from proprioceptive sensors [10]–[13]. However, geometric representations fail to capture environmental semantics, such as object affordances, social awareness, or task-specific details. Understanding environmental semantics is essential for robots to synthesize interactive and skillful behaviors, which are crucial for executing more complex high-level tasks [14]–[17].

Recent advancements in foundation models, such as large language models (LLMs) and vision-language mod-

els (VLMs), have created opportunities to integrate higher-level reasoning with low-level robot control by using vision-language observations [18]–[20]. However, the advantages of modern foundation models are often hindered by challenges in integrating these large models with control policies that require real-time decision-making [21], [22].

We propose LocoVLM, a novel framework for legged locomotion that brings the vast knowledge of foundation models into the field of legged locomotion. LocoVLM grounds image and language inputs to adapt versatile locomotion policies in real time, yielding a robust and interactive legged locomotion system.

Fig. 1 provides an overview of our proposed LocoVLM framework, which can be viewed as a hierarchical system following a teacher-student paradigm for the high-level policy. The LLM functions as a high-level teacher policy, generating high-level commands for the robot, while the VLM serves as a high-level student policy that extracts a subset of domain knowledge from the teacher policy. The low-level policy is the style-conditioned locomotion controller that executes the given motions based on the high-level commands.

In summary, the contributions of this paper aim to address the following key challenges in integrating vision-language models with legged robot locomotion:

1) **Robust style-conditioned locomotion.** A locomotion policy learning framework that efficiently conditions locomotion styles while maintaining robustness, enabling the robot to accurately follow semantic instructions without compromising stability.
2) **Foundation model knowledge distillation.** A scalable data generation pipeline that utilizes an LLM to generate a database of instructions and their corresponding executable motion descriptor, allowing real-time inference and adaptation of the robot's locomotion policy without requiring in-the-loop queries to an LLM.
3) **Fast and accurate retrieval.** A vision-language grounding approach that facilitates real-time retrieval of motion descriptor from the database using a pre-trained VLM, enabling the robot to adapt its locomotion policy based on human instructions or robot-centric observations.
4) **Generalization.** An analysis of the framework's generalization capabilities across different tasks and embodiments, highlighting its potential for further applications in real-world scenarios.

II. Related Work

A. Locomotion Skill Control

Recent advancements in learning methods for legged locomotion have enabled robust and skillful control, significantly improving the robot's adaptability and versatility. The learning of these locomotion skills is generally guided by reference commands in the form of base velocity [4], [5], [8], [10], foot trajectory [23], or body pose [9], [11]. Recent works in learning-based locomotion control have also nurtured the learning of controllable skills using heuristics defined in the reward functions. These heuristic rewards have facilitated the learning of gait style-conditioned [24]–[27] or contact-scheduled [14], [28] policies. However, these heuristic reward functions often undermines the controller's robustness, as the reward design may tend to satisfy the heuristics rather than maintaining robustness against disturbances.

B. LLM as Robot Policies

The success of LLMs in reasoning and generating human-like text has sparked interest in using them as robot policies. A common approach runs the LLM in the control loop to generate executable actions, such as code [29], sub-task plans [15], [30], rewards [31], or high-level commands [14], [21], [32]. However, deploying LLMs for real-time control remains challenging, especially in field deployments, where a dedicated cloud connectivity is often unavailable. As a result, most deployments are constrained to controlled environments with dedicated networks [14], [15], [17], or must trade off smooth real-time performance [21], [33].

C. LLM as a Data Generator

Another paradigm for utilizing LLMs is to employ them as data generators. The motivation behind this approach is: *if LLMs can generate human-like texts, can they also annotate or code like humans?* This idea has been recently explored to scale up data generation for training robot policies with minimal human effort in data collection [34], [35] and to accelerate code composition for training environment generation [36]–[38]. Leveraging LLMs as data generators can rapidly scale up data for various tasks, reducing the manual effort required for data annotation.

III. Methodology

A. Versatile Quadrupedal Locomotion

1) Style-Conditioned Locomotion Policy: We trained a style-conditioned locomotion policy using a blind velocity-conditioned locomotion learning framework [4], [5], [7], [8], [24], augmented with a style parameter vector that parameterizes gait style using a gait cycle duration T and gait phase offsets $\psi \in \mathbb{R}^4$. A gait phase encoding vector was used as a clock input for the policy, defined as $\boldsymbol{\phi}(t) = [\sin\left(2\pi\frac{t}{T}\right), \cos\left(2\pi\frac{t}{T}\right)]$, where t denote the current time step.

2) Compliant Contact Tracking: Style-conditioned contact tracking often compromises locomotion robustness to enhance style accuracy. Therefore, we propose a compliant contact tracking method that allows accurate gait tracking while preserving the robustness of the locomotion controller. This behavior is achieved by incorporating a compliance term into the contact tracking reward function, which imposes zero penalty when the foot is not in the correct contact state within the compliance threshold. The compliance term is defined as:

$$\phi_{\text{error}}^{\text{comply}} = \begin{cases} 0, & \text{if } \phi_{\text{error}} \leq \delta, \\ \phi_{\text{error}}, & \text{otherwise}, \end{cases} \tag{1}$$

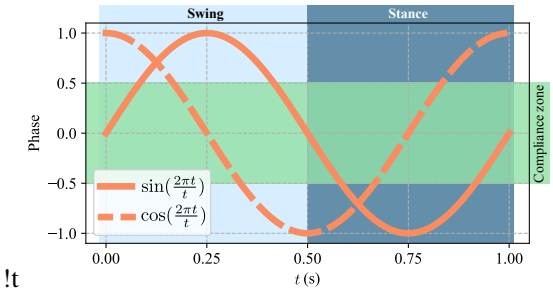

Fig. 2. Gait phase encoding for a cycle duration of $T = 1$. The gait phase encoding vector is a two-dimensional representation of the current phase in the gait cycle.

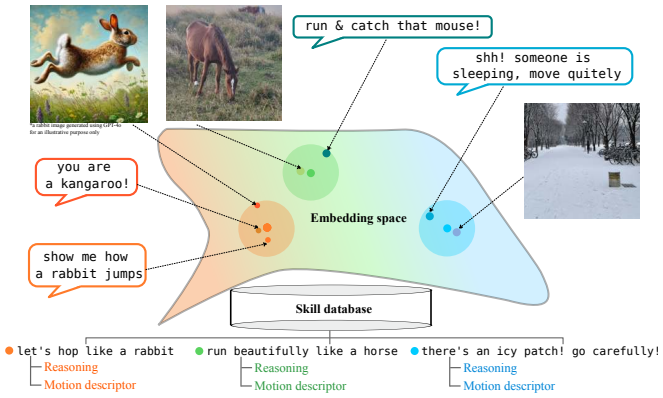

Fig. 3. Skill database retrieval process. The instruction query is encoded by a VLM into a learned embedding space. A VLM is used instead of a sentence encoder to enable multimodal retrieval from text or image inputs. The closest instruction embedding in the database is retrieved to obtain the corresponding motion descriptor.

where δ is the compliance threshold, and ϕ_{error} represents the error between the desired and actual contact states. The desired contact states for the swing and stance phases are zero and one, respectively. Fig. 2 illustrates the gait phase encoding for $T = 1$ with $\delta = 0.5$. Within the compliance zone (green shade), the policy is not rewarded for tracking the cycle phase (orange curves), allowing it to compliantly adapt to disturbances.

B. Scaling Up Motion Description Data

We utilize an LLM to generate a database of instructions and motion descriptors for the robot's locomotion policy. By creating a database of instructions and their corresponding motion descriptors, we distill the commonsense knowledge of the LLM into a structured instruction set tailored for the robot's locomotion policy.

1) Instruction Description Generation: We propose a two-stage data generation pipeline to efficiently scale up the data generation process. In the first stage of the data generation process, we prompt the LLM to generate a set of brief instructions and/or scene descriptions $\mathbf{I}$. These instructions are categorized into three types: (1) mimicking behaviors, (2) responding to a scene, and (3) following direct instructions. More details regarding the instruction generation and the system prompts used in this paper is supplemented Appendices I and II, respectively.

2) Instruction-Grounded Motion Description: The LLM-generated instructions and motion descriptors are stored in a skill database $\mathcal{D}$. During deployment, a pre-trained VLM encodes the input instruction, provided as text or an image, into an embedding space. The closest instruction embedding in the database is then retrieved to obtain the corresponding motion descriptor, as illustrated in Fig. 3.

Each entry in the database is a tuple $d = (\mathcal{I}, \mathbf{M})$, where $\mathcal{I} \in \mathbf{I}$ is the instruction text and $\mathbf{M}$ is the motion descriptor. The motion descriptor is defined as follows:

$$\mathbf{M} = \begin{bmatrix} T \\ \psi \\ v_x^{\text{limit}} \end{bmatrix}, \tag{2}$$

where ψ is the gait phase offsets and v_x^{limit} is the maximum velocity along the x-axis. Without loss of generality, we use only the velocity limit along the x-axis in the motion descriptor. However, extending the descriptor to include velocity limits along the y-axis and the yaw rate is straightforward and could benefit the generation of more diverse motions. In this paper, we limit the motion descriptor to three elements, representing the minimum parameters required to synthesize various locomotion styles.

3) Prompted Reasoning for Motion Description Generation: Generating low-level motion descriptions from direct instructions (e.g., "`walk forward`") is relatively straightforward, as these can be directly mapped to parameters such as gait type, gait cycle period, and velocity limit. However, vague instructions (e.g., "`move like a hippo`") or contextual cues (e.g., "`you are in a library`") pose greater challenges.

To circumvent this issue, we introduce a prompted reasoning method that guides the LLM in generating diverse, executable skills. The key idea is to translate high-level instructions into detailed and technical descriptions. This is achieved by prepending a system prompt that asks the LLM to first produce reasoning before outputting a motion description. As shown in Fig. 3, each skill entry consists of an instruction, its reasoning, and the resulting motion descriptor. A complete listings of the system prompt for the skill generation with prompted reasoning is provided in Appendix III.

C. Vision-Language Model as a Motion Advisor

1) Mixed-Precision Retrieval: During deployment, LocoVLM retrieves the most similar instruction from the skill database and forwards it to the locomotion policy. We used the encoder of a pre-trained BLIP-2 model [39] to extract embeddings from both text and image queries. These embeddings are then compared with the instructions in the database to identify the closest match.

A key challenge in database scaling is retrieval efficiency. Accurate retrieval using BLIP-2's image-text-matching (ITM) head requires exhaustively comparing the query against all database entries. In contrast, computing cosine similarity between query and instruction embeddings

Algorithm 1 Mixed-Precision Retrieval

Input: $\mathcal{I}_{\text{query}}$, $\mathcal{D}$, f_{BLIP}, f_{ITM}, K
$\mathbf{I}^K \leftarrow \arg\max_{\mathcal{I}\in\mathcal{D}} \text{cossim}\left(f_{\text{BLIP}}(\mathcal{I}_{\text{query}}), f_{\text{BLIP}}(\mathbf{I})\right)$
$p_1(\mathbf{I}^K) \leftarrow \text{softmax}\left(\text{cossim}\left(f_{\text{BLIP}}(\mathcal{I}_{\text{query}}), f_{\text{BLIP}}(\mathbf{I}^K)\right)\right)$
for $\mathcal{I}^k \in \mathbf{I}^K$ **do**
    $p_2(\mathcal{I}_k) \leftarrow \text{softmax}\left(f_{\text{ITM}}(\mathcal{I}_{\text{query}}, \mathcal{I}_k)\right)$
end for
$\mathcal{I}^* \leftarrow \arg\max_{\mathcal{I}_k} \left(p_1(\mathbf{I}^K) + p_2(\mathbf{I}^k)\right)$
Output: $\mathcal{I}^*$

is more scalable but less accurate, as it does not leverage the ITM head's pre-trained feature fusion [39].

Therefore, we introduce a mixed-precision retrieval method that combines the best of both schemes. Our mixed-precision retrieval method breaks down the retrieval process into two stages. The problem of retrieving an instruction given a query is formulated as follows:

$$\mathcal{I}^* = \arg\max_{\mathcal{I}\in\mathcal{D}} \text{sim}\big(\mathcal{I}_{\text{query}}, \mathbf{I}\big), \tag{3}$$

where $\mathcal{I}_{\text{query}}$ represents the instruction query, $\mathcal{I}^*$ is the retrieved instruction, $\mathcal{D}$ is the skill database, and $\text{sim}()$ is the similarity metric. A pseudocode summarizing the proposed mixed-precision retrieval method is provided in Algorithm 1.

2) Text as Image Helps Sentence Understanding: State-of-the-art VLMs are generally trained on large image-text datasets using contrastive methods [39]–[41], which align image-text pairs but ignore relationships between text pairs. To address this limitation, we hypothesize that representing text as an image can improve VLM performance. Specifically, we render the text query as a *text image*, a white background with foreground text—using a simple plotting tool like Matplotlib [42]. This image is then fed into the VLM to perform image-based text retrieval by comparing it with instruction images in the skill database. This approach leverages the VLM's strength in image-text matching to maximize LocoVLM 's retrieval performance.

IV. Experiments

The primary objective of the proposed LocoVLM framework is to realize a versatile and interactive locomotion policy grounded in vision-language instructions. To evaluate its effectiveness, we conducted a series of experiments to validate our contributions by addressing the following questions:

1) **Robust style-conditioned locomotion.** Is the proposed compliant contact tracking method capable of executing various locomotion styles while maintaining robustness? (Section IV-A)
2) **Foundation model knowledge distillation.** Is the LLM-generated database of instructions and their corresponding executable motion descriptors effective for real-time inference and adaptation of the robot's locomotion policy without requiring the LLM during deployment? (Section IV-B)
3) **Fast and accurate retrieval.** Can the proposed retrieval method quickly and accurately retrieve the most relevant motion descriptor from the database using a pre-trained VLM? (Section IV-C)
4) **Generalization.** How well does the proposed LocoVLM framework generalize across different tasks and embodiments without further fine-tuning? (Section IV-D)

For details on the system setup, please refer to Appendix IV.

TABLE I
Performance comparison between contact tracking with and without compliance. The mean and standard deviation of the traveled distances for 1,000 rollouts are reported for each terrain. The shaded values indicate the best among the two methods.

Gait	δ	Traveled distance (m) ↑ Rough	Discrete	Stairs
Pronk	0	15.13 ± 3.61	17.34 ± 6.41	13.56 ± 2.53
	0.5	15.59 ± 3.38	18.42 ± 6.29	14.06 ± 2.74
Trot	0	16.72 ± 3.53	16.62 ± 6.50	14.92 ± 2.82
	0.5	16.84 ± 3.72	18.29 ± 6.18	16.34 ± 2.41
Pace	0	17.11 ± 3.51	17.48 ± 5.89	16.83 ± 2.35
	0.5	16.31 ± 3.95	19.49 ± 5.77	18.18 ± 2.49
Bound	0	14.57 ± 2.91	14.53 ± 5.64	14.17 ± 2.29
	0.5	16.04 ± 3.57	17.43 ± 5.16	15.48 ± 2.92
Rotary gallop	0	16.67 ± 3.56	16.71 ± 6.20	16.40 ± 2.88
	0.5	18.18 ± 3.91	18.61 ± 5.08	18.04 ± 2.10

A. Gait Tracking Performance

We measured the average traveled distance for different gaits with and without compliance threshold to assess the trade-off between accuracy and robustness in the proposed compliant contact tracking method. Table I summarizes the results obtained from simulation using $1{,}000$ rollouts. For each robot, we set a command velocity of $1.2\ \text{m/s}$ and a gait cycle period of $0.4\ \text{s}$. The maximum episode length of the simulation was set to $20\ \text{s}$.

The traveled distance of the robot consistently increases when using the compliant tracking method compared with the baseline ($\delta = 0$) [24]. This performance gain is attributed to the compliance introduced in the foot contact tracking, which allows the robot to adapt to follow the desired gait pattern, but also comply to violate the gait pattern when necessary to overcome obstacles. The performance improvement is more pronounced on discrete and stairs terrains, where the robot is more vulnerable to external disturbances due to frequent foot collisions with the terrain.

The proposed compliant contact tracking allows our robot to track foot contact states accurately. A qualitative performance evaluation is presented in Fig. 4. The top row displays the foot contact states, while the bottom row shows snapshots of the robot for five different gaits. The robot accurately and robustly tracks the desired foot contact positions for all gaits.

B. Motion Description Data Scaling

We compare the skill database statistics of 300 data points generated by the LLM in Fig. 5. As a baseline, we generated motion descriptors using a method similar to SayTap [14],

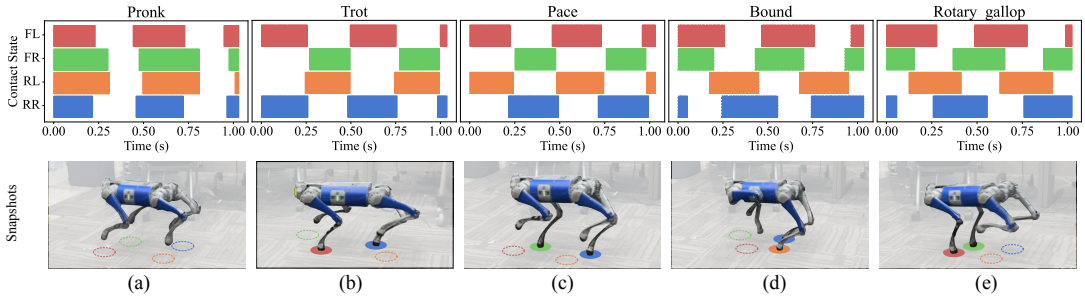

Fig. 4. Gait tracking performance of the locomotion policy. The top row shows the foot contact states, while the bottom row displays snapshots of the robot. The robot accurately follows the desired gait patterns for various gaits, namely: (a) pronk, (b) trot, (c) pace, (d) bound, and (e) rotary gallop.

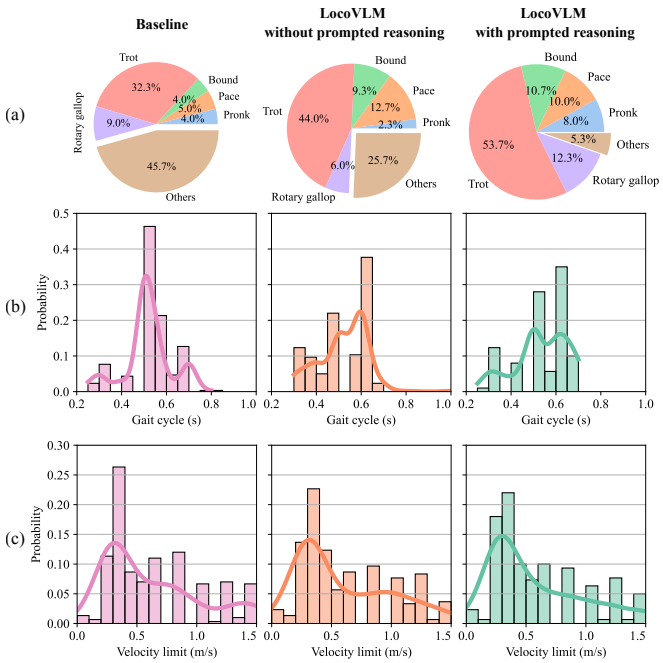

Fig. 5. Statistics of the skill database generated with baseline [14], LocoVLM without prompted reasoning, and LocoVLM with prompted reasoning. Each database contains 300 data points and was generated using the same LLM-generated instructions. The statistics include (a) categorical gait distribution, (b) histograms of gait cycle period distribution, and (c) histograms of velocity limit distribution. The histograms are normalized to have a total probability of 1.

where the LLM is prompted to generate one motion descriptor at each step without any reasoning. We then compare this baseline with variants of LocoVLM: one without prompted reasoning and another with prompted reasoning.

The key difference between the baseline and LocoVLM without prompted reasoning is how the motion descriptors are generated. LocoVLM queries the VLM to generate motion descriptors in batches, significantly reducing computational and monetary costs. For instance, the total cost of generating 300 motion descriptors using the baseline, LocoVLM without prompted reasoning, and LocoVLM with prompted reasoning is approximately 1.16, 0.21, and 0.25 USD, respectively, when using the GPT-4o model. Furthermore, this batch generation process provides the LLM with a form of memory, preventing duplicate motion descriptors from being generated across queries.

1) Categorical Gait Distribution: As shown in Fig. 5(a), the baseline database contains many unstructured gait phase offsets (marked as *others*), which may compromise robot stability. LocoVLM without prompted reasoning reduces these unstructured gaits by generating motion descriptors in batches, decreasing repetition and instability. Prompted reasoning further improves the distribution, reducing unstructured gaits and promoting a more even spread across the five standard gaits. This structured understanding is especially beneficial for vague commands lacking explicit gait cues, such as "`shh! someone is sleeping, move quietly`".

2) Motion Descriptors Statistics: The histogram in Fig. 5(b) shows that prompted reasoning leads to a more balanced distribution of gait cycle periods within 0.2 to 0.7 sec. In contrast, the baseline and LocoVLM without reasoning cluster around 0.5 sec and include more outliers, with unstable values up to $T = 1.0$ sec. As for the velocity limit (Fig. 5(c)), there is no notable difference across methods, likely because this parameter is more intuitively inferred from instructions than the gait period or phase offset.

TABLE II
RETRIEVAL ACCURACY OF LOCOVLM USING DIFFERENT RETRIEVAL METRICS FOR 100 INSTRUCTIONS FROM THE DATABASE.

Retrieval Metric	Text as String	Text as Image	Average
Cosine similarity	21/100	30/100	20.5%
Top-K similarity	27/100	48/100	37.5%
Top-K to ITM	51/100	57/100	54.0%
Mixed-precision	**72/100**	**87/100**	**79.5%**

C. Retrieval Performance

1) Accuracy: To quantitatively evaluate the retrieval performance of the VLM, we manually annotated 100 instructions and their corresponding motion descriptors based on our understanding of the instructions due to the lack of publicly-available groundtruth.

We hypothesize that retrieval performance degrades as the instruction database scales, which is supported by the results in Table II. With the text as string input, cosine similarity correctly retrieved only 21% of the instructions. The top-K similarity improves this to 27% by narrowing the retrieval scope to the most similar candidates. Re-ranking with the BLIP-2 ITM head increases accuracy to 51%, but the improvement remains marginal.

Our proposed mixed-precision retrieval significantly improves performance, retrieving 72% of the instructions, suggesting greater robustness and accuracy. The improvement is

Fig. 6. LocoVLM succesfully interprets the scenes in the form of RGB images and provides the robot with motion descriptors that are suitable for the observed scenes.

attributed to combining two levels of similarity: top-K similarity captures low-level textual cues, while the ITM metric captures higher-level semantics. Their combination enables more accurate retrieval than either metric alone. Additionally, incorporating text-as-image representation further improves the accuracy up to 87%.

2) Interpretation of Queries Out of the Database.: One important feature of LocoVLM is its ability to interpret instructions outside of the database. This enables users to interact with the robot more naturally and intuitively, without being constrained by the queries in the database. For instance, the VLM interprets the query "`you are a kangaroo`" as "`let's jump like a rabbit`", even though the query is absent from the database. Similarly, it responds to the query "`this is a library!`" by suggesting "`move quietly`", which aligns with common behavior in a library. These results highlight that the semantic reasoning capabilities of the VLM are sufficient to retrieve feasible motion commands, even for out-of-database queries.

3) Interpretation of Robot-Centric Image Queries.: We evaluated LocoVLM's performance on image-based queries captured by the robot's onboard camera without additional processing as. The experiment took place outdoors in a campus environment, where the robot transitioned from pavement to snow-covered terrain as shown in Fig. 6.

On the pavement, LocoVLM interpreted the scene as "`traipse lightly like a deer`", prompting a trot with a moderate velocity limit of 0.6 m/s. In snowy areas, it produced interpretations like "`a field of ice, walk light-footed`" or "`skulk with stealth like a lynx`", resulting in slower gaits with lower velocity limits (0.2-0.3 m/s) and longer gait cycle periods (0.6-0.7 s). The lynx analogy is especially notable given the lynx's natural habitat in cold, snowy environments.

These results demonstrate LocoVLM's ability to process robot-centric image queries and generate context-aware locomotion commands. This is particularly useful when environments are visually distinct but geometrically similar, as in our pavement-to-snow scenario.

D. Zero-Shot Generalization Across Embodiments

We demonstrate the feasibility of utilizing LocoVLM to generalize across different legged robot embodiment, specifically on a humanoid robot. We trained a style-conditioned locomotion policy for a Unitree H1 robot with minimum modifications, i.e., by changing the gait phase offsets only for two legs. Therefore, the policy is trained only on the `trot` and `pronk` gaits.

We directly utilized the skill database generated for quadrupedal robots by using only the first two gait offsets for each instruction. Due to resource limitations, we experimented only in a MuJoCo simulation environment [43], [44]. The results in Fig. 7 show successful skill retrieval and execution on the humanoid robot. This experiment demonstrates the generalizability of LocoVLM that is attributed to the generalized motion parameterization and the instruction-grounded skill database.

V. Conclusion

In this paper, we presented LocoVLM, a novel hierarchical locomotion adaptation framework that leverages pre-trained foundation models as a high-level motion advisor for a versatile style-conditioned locomotion policy. Unlike recent works that utilize LLMs for in-the-loop control, LocoVLM does not require in-the-loop LLM access, thereby enabling deployment in real-time systems without the need for continuous connectivity to the LLM. Specifically, we proposed a two-stage data generation process to efficiently scale up data generation and distill the knowledge from the LLM into an efficient offline VLM. Our framework was validated through extensive experiments, demonstrating its skill expressiveness and generalizability across different legged robot embodiments and tasks.

References

[1] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick *et al.*, "Autonomous Spot: Long-range autonomous exploration of extreme environments with legged locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020, pp. 2518–2525.

[2] J. Lee, M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter, "Learning robust autonomous navigation and locomotion for wheeled-legged robots," *Sci. Robot.*, vol. 9, no. 89, p. eadi9641, 2024.

[3] A. Jacoff, J. Jeon, O. Huke, D. Kanoulas, S. Ha, D. Kim, and H. Moon, "Taking the First Step Toward Autonomous Quadruped Robots: The quadruped robot challenge at ICRA 2023 in London [Competitions]," *IEEE Robot. Autom. Mag.*, vol. 30, no. 3, pp. 154–158, 2023.

[4] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Robot. Sci. Syst.*, 2021.

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

[6] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. PMLR Conf. Robot. Learn.*, 2022, pp. 91–100.

[7] 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 Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4630–4637, 2022.

[8] I. M. A. Nahrendra, B. Yu, and H. Myung, "DreamWaQ: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023.

[9] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2022, pp. 2497–2503.

[10] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, p. eabk2822, 2022.

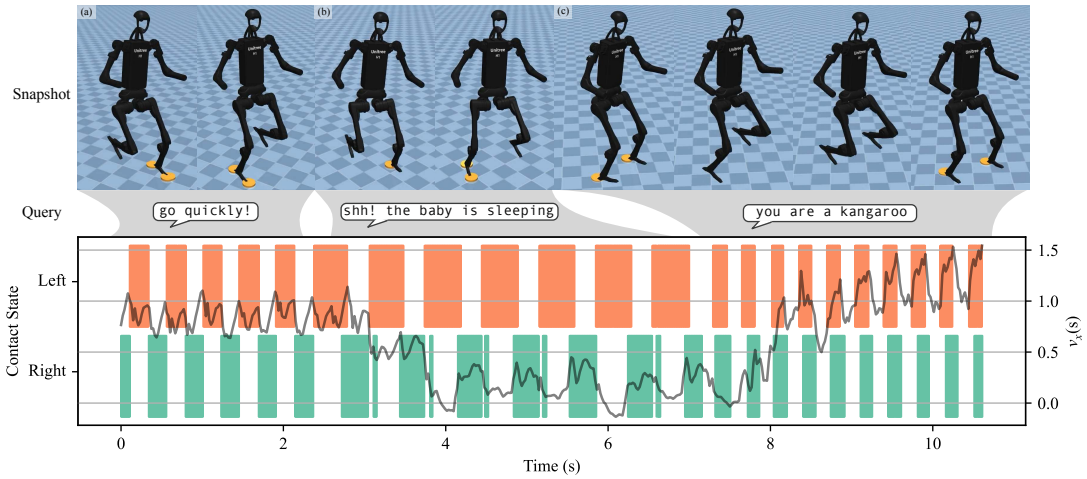

Fig. 7. Although LocoVLM is configured for a quadrupedal robot, it can generalize to a humanoid robot in a zero-shot manner by utilizing only a subset of the gait phase offsets. The humanoid robot successfully adapts its locomotion style to follow the given text query.

[11] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "ANYmal parkour: Learning agile navigation for quadrupedal robots," *Sci. Robot.*, vol. 9, no. 88, p. eadi7566, 2024.

[12] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 11 443–11 450.

[13] I. M. A. Nahrendra, B. Yu, M. Oh, D. Lee, S. Lee, H. Lee, H. Lim, and H. Myung, "Obstacle-aware quadrupedal locomotion with resilient multi-modal reinforcement learning," *arXiv preprint arXiv:2409.19709*, 2024.

[14] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada, "SayTap: Language to quadrupedal locomotion," in *Proc. PMLR Conf. Robot. Learn.*. PMLR, 2023, pp. 3556–3570.

[15] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning," in *Proc. PMLR Conf. Robot. Learn.*, 2023, pp. 23–72.

[16] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do As I Can and Not As I Say: Grounding language in robotic affordances," in *Proc. PMLR Conf. Robot. Learn.*, 2022.

[17] M. F. Ginting, D.-K. Kim, S.-K. Kim, B. J. Krishna, M. J. Kochenderfer, S. Omidshafiei, and A.-a. Agha-mohammadi, "SayComply: Grounding field robotic tasks in operational compliance through retrieval-based language models," *arXiv preprint arXiv:2411.11323*, 2024.

[18] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "RT-1: Robotics transformer for real-world control at scale," *Robot. Sci. Syst.*, 2023.

[19] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proc. PMLR Conf. Robot. Learn.*, 2023, pp. 2165–2183.

[20] S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi, and D. Sadigh, "RT-H: Action hierarchies using language," *Robot. Sci. Syst.*, 2024.

[21] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and A. Zeng, "Large language models as general pattern machines," in *Proc. PMLR Conf. Robot. Learn.*, 2023, pp. 2498–2518.

[22] Y. Kim, D. Kim, J. Choi, J. Park, N. Oh, and D. Park, "A survey on integration of large language models with intelligent robots," *Intelligent Service Robotics*, vol. 17, no. 5, pp. 1091–1107, 2024.

[23] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3699–3706, 2020.

[24] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Proc. PMLR Conf. Robot. Learn.*, 2023, pp. 22–31.

[25] G. Kim, Y.-H. Lee, and H.-W. Park, "A learning framework for diverse legged robot locomotion using barrier-based style rewards," *arXiv preprint arXiv:2409.15780*, 2024.

[26] G. Bellegarda, M. Shafiee, and A. Ijspeert, "AllGaits: Learning all quadruped gaits and transitions," *arXiv preprint arXiv:2411.04787*, 2024.

[27] J. Humphreys and C. Zhou, "Learning to Adapt: Bio-inspired gait strategies for versatile quadruped locomotion," *arXiv preprint arXiv:2412.09440*, 2024.

[28] C. Zhang, W. Xiao, T. He, and G. Shi, "WoCoCo: Learning whole-body humanoid control with sequential contacts," in *Proc. PMLR Conf. Robot. Learn.*, 2024.

[29] 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 *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9493–9500.

[30] Z. Mandi, S. Jain, and S. Song, "RoCo: Dialectic multi-robot collaboration with large language models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 286–299.

[31] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik *et al.*, "Language to rewards for robotic skill synthesis," in *Proc. PMLR Conf. Robot. Learn.*, 2023, pp. 374–404.

[32] A. Jiao, T. P. Patel, S. Khurana, A.-M. Korol, L. Brunke, V. K. Adajania, U. Culha, S. Zhou, and A. P. Schoellig, "Swarm-GPT: Combining large language models with safe motion planning for robot choreography design," *arXiv preprint arXiv:2312.01059*, 2023.

[33] A.-C. Cheng, Y. Ji, Z. Yang, X. Zou, J. Kautz, E. Bıyık, H. Yin, S. Liu, and X. Wang, "NaVILA: Legged robot vision-language-action model for navigation," *arXiv preprint arXiv:2412.04453*, 2024.

[34] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Proc. PMLR Conf. Robot. Learn.*, 2023, pp. 3766–3777.

[35] A. Yu, G. Yang, R. Choi, Y. Ravan, J. Leonard, and P. Isola, "Learning visual parkour from generated images," in *Proc. PMLR Conf. Robot. Learn.*, 2024.

[36] 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," in *Proc. Int. Conf. Learn. Represent.*, 2024.

[37] Y. J. Ma, W. Liang, H. Wang, S. Wang, Y. Zhu, L. Fan, O. Bastani, and D. Jayaraman, "DrEureka: Language model guided sim-to-real transfer," in *Robot. Sci. Syst.*, 2024.

[38] W. Liang, S. Wang, H.-J. Wang, O. Bastani, D. Jayaraman, and Y. J. Ma, "Environment curriculum generation via large language models," in *Proc. PMLR Conf. Robot. Learn.*, 2024.

[39] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proc. PMLR Int. Conf. Mach. Learn.*, 2023, pp. 19 730–19 742.

[40] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. PMLR Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[41] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *Proc. PMLR Int. Conf. Mach. Learn.*. PMLR, 2022, pp. 12 888–12 900.

[42] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[43] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2012, pp. 5026–5033.

[44] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," 2022. [Online]. Available: http://github.com/google-deepmind/mujoco_menagerie

[45] 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.

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

[47] 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," *Advances in Neural Information Processing Systems (NeurIPS), Track on Datasets and Benchmarks*, 2021.

APPENDIX

I. DESCRIPTION GENERATION DETAILS

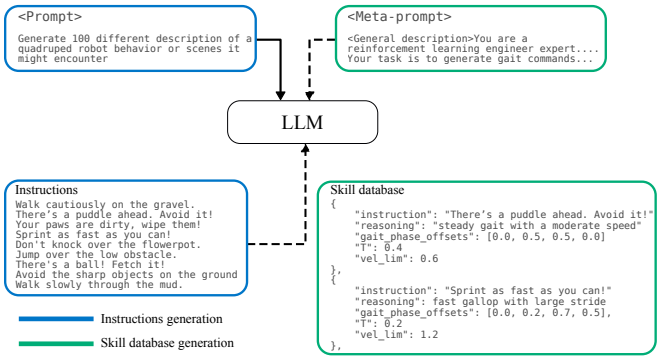

Fig. 8. Offline skill database generation pipeline. The LLM firstly generates instructions, which are categorized into mimicking behaviors, scene responses, and direct instructions. These instructions are then passed to a meta-prompt to generate contents for the skill database.

The two-stage description generation as shown in Fig. 8 offers two primary benefits: (1) it mitigates the maximum token length constraint of the LLM, and (2) it enables the LLM to generate diverse and structured motion descriptors by passing the generated instructions to a meta-prompt that produces the motion descriptors. We used the GPT-4o [45] model as the LLM.

Rather than generating all three types of instructions in a single prompt, we observed that the LLM produces more diverse yet structured instructions when prompted separately for each category. A crucial key implementation detail was prompting the LLM to explicitly generate n instructions for each category instead of generating all instructions simultaneously. This categorical prompting approach prevents the LLM from hallucinating false instructions and ensures that the generated instructions are relevant to the category. To comply with the maximum token length constraint of the LLM, we set $n = 100$ for each generation process.

II. INSTRUCTION PROMPT

The instruction prompt is used to categorically generate three types of instructions: (1) mimicking behaviors, (2) scene descriptions, and (3) direct instructions. A template prompt is used to give general instructions to the LLM about its basic tasks, as shown in Listing 1.

Afterward, the instructions for each category are generated by substituting categorical prompts into the template prompt. We attached each categorical prompts in Listings 2, 3, and 4.

We used the GPT-4o [45] model as the LLM to generate the instructions. The meta and category prompts are given to the LLM application programming interface (API) with a `system` role, and the number of instructions to generate is given as a `user` role. The LLM generates the instructions in the form of listed texts.

Listing 1: Template prompt for generating short instructions.

```
<General instructions>
#You are an expert in biomechanics, especially in
     quadrupedal biomechanics.
#You are familiar with various gait pattern of a
     quadrupedal animal.
#Given an input <N> that indicates the number for
     descriptions, you should output <N> different
     descriptions
{DESCRIPTION_TYPE}
```

Listing 2: Categorical prompt for generating instructions to mimic certain behaviors.

```
Your task now is to tell a quadrupedal animal about how to
     :
- mimic an animal
- sometimes define the appropriate speed: slow, fast,
     moderate
- your output should be brief and concise for a short
     description

<Input-Output format and examples>
Input: 8
Output:
- let's hop like a rabbit
- run beautifully as a horse
- trot slowly, resembling a dog
- frogs jump by lifting all its foot!
- show me a hamster
- show me how a dog run quickly
- mimic a cat walk slowly
- do a galloping zebra!
```

Listing 3: Categorical prompt for generating instructions for giving directive commands.

```
Your task now is to tell a quadrupedal animal about how to
     :
- do some particular actions, especially related to:
* gait pattern
* foot stepping frequency
* speed
* loudness of the food stepping
* carefulness of the foot stepping

<Input-Output format and examples>
Input: 11
Output:
- trot with high frequency
- pace slowly and quietly
- trot slowly
- bound quickly!
- bound very slowly please
- tell me what a pace looks like
- oh no! catch that thief running!
- stop where you are!
- dance for me!
- reduce your noise
- let's be loud!
```

Listing 4: Categorical prompt for generating descriptions of particular scenes

```
Your task now is to:
- Decribe a scene that can be encountered in daily lives
- The scene requires the animal gait to be changed, e.g.
1. slowing down in a library
2. careful on ice
3. stopping at dangerous edge
- The description should be diverse but conceise

<Input-Output format and examples>
Input: 8
Output:
- be careful of the slippery tiles
- this is a library
- someone is sleeping. be quiet
- ouch! the floor is too hot!
- carefully approach the stairs
- on no! its a dead end!
- there is a meeting, silent
- the snow is slippery
```

III. SKILL PROMPT

The skill prompt is used to translate instructions into motion descriptors. The meta-prompt for the skill prompt is shown in Listing 5. This prompt tells the LLM to generate motion descriptors based on the input instructions. Before generating the motion descriptors, the LLM is first required to generate a reasoning prompt explaining the rationale behind the motion descriptors (Section III-B.3). The motion descriptors are generated in the form of a structured `.json` file content.

We also shuffled the input instructions to the LLM to prevent the model from memorizing the input-output pairs. This shuffling process is crucial to prevent the LLM from generating repetitive motion descriptors.

Listing 5: Meta-prompt for generating motion descriptors given a list of instructions.

```
<General description>
You are a reinforcement learning engineer expert,
     specializing in quadrupedal robots.
Your task it to provide gait style commands of a
     quadrupedal robot based on the user input that
     describes the robot's behavior or surroundings.
You are required to ALWAYS give the output in a correct
     and standardized form.

<What is gait styles?>
We  parameterize gait styles using 2 parameterized values
1. T: float
 - T is the gait cycle duration of the robot
 - T is proportional to 1/f, where f is the number of foot
      contact per 1 s
 - 0.2 < T <= 1
 - Lower T means more step per second and vice versa
 - Higher T is required for longer walking stride similar
     to how cheetah runs.

2. gait_phase_offsets: List[float, float, float, float]
 - The sequence of the variable in the list is [FL, FR, RL
     , RR]
 - FL=front left foot
 - FR=froont right foot
 - RL=rear left foot
 - RR=rear right foot
 - each offsets ranges between 0 to 1

3. vel_lim: float
 - The robot can ideally move up to 1.5m/s
 - The robot is small, its body length is around 40cm
 - Setting low vel_lim will make the robot move slower
 - Setting high vel_lim allows the robot to move faster
 - Set vel_lim to 0 to make the robot completely stop


<Some basic gait_phase_offsets dict>
You can use this dict as your reference. But I expect you
     to be creative, not confined to this example only.
gait_dict = {
            'trot': [0.0, 0.5, 0.5, 0.0],
            'rotary_gallop': [0.0, 0.2, 0.7, 0.5],
            'pace': [0.0, 0.5, 0.0, 0.5],
            'pronk': [0.0, 0.0, 0.0, 0.0],
            'bound': [0.0, 0.0, 0.5, 0.5],
        }
A brief contextual description:
1. Trot is a diagonal foot contact, known to be one of the
      stable gait, but not so creative
2. Rotary gallop shows rotary contact for each foot.
3. Pace shows synchronous contacts between each left/right
      hemisphere
4. Pronk lifts all feet at the same time and land at the
     same time
5. Bound lifts all front or all rear legs at the same time
6. Gallop, pronk, and bound are typically more efficient
     with higher T (>0.5) on higher speed because of more
     leg flying phase.

<Important things to note>
1. Your output MUST NOT biased or overfit to those in the
     gait dict. The gait dict is only an example gaits,
     where the controller can perform well. You are free
     to create new gait lists and be creative!
2. You can select T between 0 to 1, but in general, 0.3<=T
     <=0.6 is preferrable for stability. There could be
     cases where you need T>0.6, but it is not desired.
3. NEVER give T<0.2

<Reasoning>
You should output a brief reasoning of the gait that you
     recommend.

<Example input-output format>
Input:
-trot slowly
-trot quickly
-pace quickly
-save your energy
-bound like a rabbit
-walk slowly
-you are a frog
-gallop swiftly
-be quiet!
Output:
{
    "instruction": "trot slowly",
    "reason": "trot with low vel_lim",
    "T": 0.6,
    "gait_phase_offsets": [0.0, 0.5, 0.5, 0.0],
    "vel_lim": 0.4
},
{
    "instruction": "trot quickly",
    "reason": "trot with high vel_lim",
    "T": 0.3,
    "gait_phase_offsets": [0.0, 0.5, 0.5, 0.0],
    "vel_lim": 1.2
},
{
    "instruction": "pace quickly",
    "reason": "pace gait, high vel_lim, low T for high
     frequency",
    "T": 0.35,
    "gait_phase_offsets": [0.0, 0.5, 0.0, 0.5],
    "vel_lim": 1.0
},
{
    "instruction": "save your energy",
    "reason": "pace gait energy-efficient, reduce to
     moderate vel_lim and T",
    "T": 0.4,
    "gait_phase_offsets": [0.0, 0.5, 0.0, 0.5],
    "vel_lim": 0.5
},
{
    "instruction": "bound like a rabbit",
    "reason": "bound with moderate vel_lim, increase T for
      longer swing",
    "T": 0.6,
    "gait_phase_offsets": [0.0, 0.0, 0.5, 0.5],
    "vel_lim: 1.0

},
{
    "instruction": "walk slowly",
    "reason": "make a slow gallop gait, reduce vel_limit
     and increase T for lower step noise",
    "T": 0.6,
    "gait_phase_offsets": [0.0, 0.2, 0.7, 0.5],
    "vel_lim: 0.3


},
{
    "instruction": "you are a frog",
    "reason": "frog jump with all legs flying, set
     moderate vel_lim, set moderate T for moderate swing
     time",
    "T": 0.45,
    "gait_phase_offsets": [0.0, 0.0, 0.0, 0.0],
    "vel_lim": 0.6
},
{
```

```
    "instruction": "gallop swiftly",
    "reason": "make a gallop gait, increase T for flying
     phase, increase vel_lim for running",
    "T": 0.6,
    "gait_phase_offsets": [0.0, 0.2, 0.7, 0.5],
    "vel_lim: 1.2

},
{
    "instruction": "be quiet!",
    "reason": "gallop slowly with high T for lower noise",
    "T": 0.65,
    "gait_phase_offsets": [0.0, 0.2, 0.7, 0.5],
    "vel_lim: 0.25
}
```

IV. System Setup

A. Locomotion Controller

For all experiments, we utilized the Unitree Go1 robot as the platform. The locomotion controller was trained using the proximal policy optimization (PPO) algorithm [46] with an asymmetric actor-critic implementation and state estimation, following the methodologies of [7] and [8]. The policy was trained in a custom implementation of the training environment built using the Isaac Gym simulator [47]. Several physical parameters were randomized to ensure sim-to-real robustness, including robot mass, center of mass, motor stiffness, damping, and terrain friction. Additionally, system delays were randomized to emulate real-world latency.

The locomotion policy was deployed on the Unitree Go1's onboard Jetson Xavier NX board. The policy was operated at 50 Hz, sending joint angle commands to the robot's motor controllers that convert the commands to motor torques at 200 Hz.

B. VLM Inference

The VLM module, based on a pre-trained BLIP-2 model, requires a GPU for inference due to the lack of CPU compatibility for certain built-in functions. For this purpose, we utilized a separate laptop equipped with an NVIDIA GeForce RTX 3070 Ti GPU. All input instructions, whether in text or image format, were transmitted to the VLM module via a robot operating system (ROS) network. The retrieved motion descriptor was subsequently sent to the robot through the same ROS network. We observed no performance degradation due to network latency, as the VLM advisor and locomotion controller operate asynchronously. Furthermore, the data transmission and VLM inference time were negligible, amounting to less than 100 ms.