LLM-driven delta-postion control for legged robots

Mohitvishnu Gadde Ashutosh Gupta Hojjat Goudarzi
Oregon State University
{gaddem, guptaash, torabigh}@oregonstate.edu

Abstract

Large Language Models (LLMs), trained on internet-scaled data, have demonstrated comparable capabilities in high-level reasoning, semantic understanding, and contextual decision-making. In this project, we explore their use as a planner for dynamic legged locomotion by introducing an interface that maps natural language commands to delta-position commands. This paper focuses on the temporal and spatial reasoning strengths of the LLMs to generate meaningful pose offsets to guide a quadruped to locomote. The main objective of this project is to investigate how well LLMs can generalize in generating these relative commands and identify scenarios where their performance degrades. These insights would inform the development of a more robust and scalable framework to enable more complex behaviors for whole-body humanoid manipulation, which potentially would require fine-tuning of the LLM to better align with the task at hand. – Demo Video

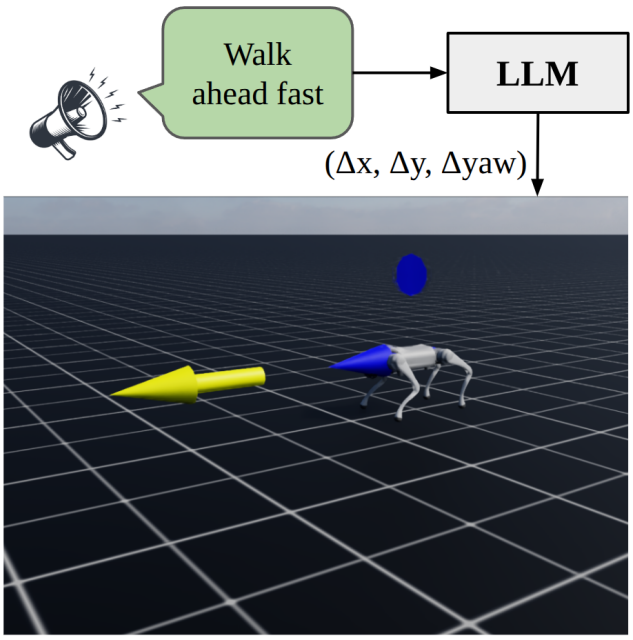

Figure 1. Illustration of our proposed system, where natural language instructions are mapped to delta-position commands that guide the robot's locomotion.

1. Introduction

Natural language is one of the most intuitive forms of communication that human beings have, which allows them to convey ideas, emotions, and instructions to one another. It also facilitates seamless contexts in various forms, like conversations to complex technical discussions. Therefore, there is a need for a framework that enables simple and effective interaction between humans and robots, to enable more natural and flexible control over robotic systems (cite). Recent advancements in Large Language Models (LLMs) have unlocked a wide range of applications deemed challenging to solve a decade ago. They are used for human assistance and language understanding, such as question-answering [14]. Recent works have also explored the use of LLMs in robotic applications, such as using them for robot control [3], scene reasoning [19], and long-horizon task planning [5].

Though LLMs have proven to be successful in various applications, it is still challenging for LLMs to comprehend low-level robotics commands such are motor torques, target joint angles, and high-frequency control signals. In this project, we investigate the use of delta-position commands as a bridge between low-level control and natural language instructions for dynamic legged robots in order to overcome this difficulty. We provide an architecture where LLMs produce a delta-pose command, which is then used by the learned locomotion controller, as opposed to translating language to motor actions. This abstraction allows LLMs to reason over commands that are grounded in time and space without having to comprehend hardware-specific actuation or operate at high control-loop frequency.

This problem remains unsolved primarily due to the difficulty of aligning the high-level semantic structure of language with the continuous, low-level demands of robotic control. Prior work has demonstrated some success using discrete contact patterns [25] or pre-trained behavior prim-

itives, but these approaches often lack the flexibility or expressiveness needed for whole-body control in dynamic environments.

The proposed approach leverages the reasoning capabilities of Large Language Models (LLMs) while delegating fine-grained motor control to task-specific controllers trained to interpret delta-position commands. We begin by developing a delta-position-based hierarchical locomotion controller for the Unitree Go2 robot. This controller executes the desired behavior based on a target pose command generated by the LLM. Specifically, we use a pretrained velocity controller capable of tracking velocity and turn-rate commands, which serves as the low-level policy. To bridge natural language and control, we design a prompt template that enables the LLM to generate delta-position outputs, comprising target positions in x, y, and target heading by leveraging its spatial and temporal reasoning abilities. The main contributions of this project are:

- A language-conditioned control framework that uses delta-position commands as an interpretable interface between LLM outputs and robot execution.
- A hierarchical control pipeline that integrates an LLM with a pretrained velocity-tracking controller on the Unitree Go2 robot.
- A prompt engineering strategy for enabling LLMs to reason over spatially grounded instructions and generate continuous motion goals.
- A set of experiments evaluating the feasibility of using LLMs for quadruped locomotion, along with analysis of failure cases to inform future fine-tuning and generalization.

The rest of the paper is organized as follows: Section 2 discusses the relevant related works, Section 3 describes the methodology of our approach, Section 4 presents the experimental setup and the results, and finally Section 5 ends with concluding remarks.

2. Related Work

Language to control robots: Recent advances in Large Language Models (LLMs) have enabled new paradigms for robot control that leverage the models' capabilities in high-level reasoning and natural language understanding. Authors of [1] introduce a modular framework that presents a modular system for executing language commands in real-world manipulation tasks that combines value-based affordance reasoning with LLMs. In this work, the instructions are broken down into smaller objectives, and actions are chosen according to their projected likelihood of achievement. [7] extends this work by incorporating sensor inputs, thus enabling multi-modal scene-aware decision making.

RT-1 [2] and RT-2 [3] introduce a transformer-based control architecture which are trained on large-scale robot data. Thus allowing LLMs to directly interface with robot observation and output tokens for required actions. On the other hand, [24] demonstrates the utility of LLMs in open-world navigation by combining GPT-3 [8], CLIP [10], and ViNG [23] to convert language into location-based navigation commands.

These systems demonstrate various contributions, including the breakdown of control into low-level actuation and high-level planning. This is done by integration of multi-modal data (vision + language), and the use of pre-trained LLMs for symbolic task reasoning. The majority of the above-described methods are limited in their applicability to dynamic, continuous domains such as legged locomotion because they either rely on discrete action spaces, pre-trained affordance primitives, or symbolic planners.

Some works have explored direct language-conditioned behaviors [15] introduce a concept of command interruptions as a critical challenge in language-guided dynamic locomotion and develop a learning-based method to detect and mitigate harmful interruptions to enable robust control policies. For quadruped locomotion, [25] presents a real-time language-to-motion module that converts natural language into continuous walking patterns.

Despite these developments, it is still difficult to get LLMs to generate low-level or mid-level control signals for dynamic robots. To address this, our work introduces a continuous delta-position command interface $(\Delta x, \Delta y, \Delta yaw)$ as a mid-level abstraction that connects natural language with low-level velocity controllers. This approach is inspired by [25], which introduces the idea of using foot contact patterns as an LLM-compatible interface for locomotion. We leverage the spatial and temporal reasoning of the LLMs to offload low-level actuation to a pre-trained controller. This design choice enables our system for real-world deployment on dynamic legged platforms and lays down path to extend it to humanoid manipulation tasks.

Locomotion controllers for Legged robotics: Legged robots have increasingly become popular across various applications such as payload transport, disaster response, agriculture, and construction [26]. However, legged locomotion presents a complex control problem, requiring both precision and robustness to manage the nonlinear dynamics of the system effectively [11].

Much of the existing research in legged locomotion has focused on developing optimization-based control strategies. Among these, Model Predictive Control (MPC) has emerged as a leading approach due to its stable, safe, and robust locomotion capabilities [29]. Recent works have demonstrated dynamic and adaptive whole-body control for

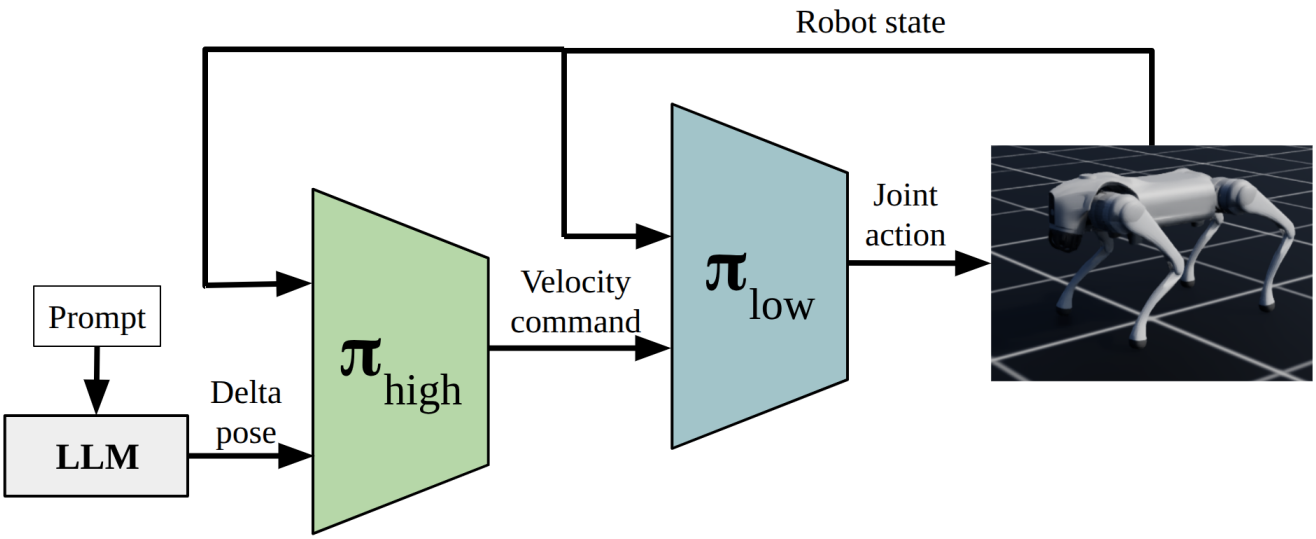

Figure 2. Overall system architecture.

quadrupeds that can effectively handle both system and environmental uncertainties [6, 20, 27]. Although MPC has made significant progress in legged locomotion, it still faces key limitations, such as sensitivity to model mismatch, slow online optimization, and difficulty in managing high-dimensional models.

On the other hand, Reinforcement Learning (RL) approaches have demonstrated impressive legged locomotion capabilities across diverse, complex, and natural environments [4,13,18,28]. RL-based approaches do not require an explicit system model. Instead, they learn control policies in simulation by interacting with the environment to maximize a reward function that describes the locomotion task objectives [16]. The resulting policies can directly map raw robot observations to joint actions in a closed-loop manner [12,17]. However, RL methods also come with their own set of challenges, such as the simulation-to-reality (sim-to-real) gap, high data sampling requirements during training [21], and limited safety guarantees in the learned policies.

To address these challenges, we propose a hierarchical control architecture that would generate intermediate target velocities and turn rates to the pre-trained low-level policy. Our work proposes a delta-position command as a mid-level abstraction that generates commands relative to the robot's local frame, which are represented as $(\Delta x, \Delta y, \Delta yaw)$ goals to the robot.

3. Methodology

The goal of this work is to enable natural language-driven control of a dynamic legged robot using delta-position commands as an intermediate interface between high-level reasoning and low-level execution. Our methodology is composed of three main components: (1) a pre-trained velocity controller, (2) a delta-position interface layer, and (3) language to delta-position.

3.1. Low-level locomotion policy

We formulate the low-level velocity controller for the quadruped as an RL policy trained to track target velocity commands. Specifically, we adopt an actor-critic framework and train the policy using the Proximal Policy Optimization (PPO) algorithm [22]. Both the actor and critic networks are implemented as multi-layer perceptrons (MLPs) with hidden layer sizes of $[512, 256, 128]$. The policy receives as input the current robot state and the target velocity command, which includes desired linear velocities in the x and y directions and angular velocity around the z-axis. The robot state consists of base linear and angular velocities, the projected gravity vector, joint positions, joint velocities, and the previous action output by the policy. The policy outputs the action in the form of delta joint positions.

To learn a stable and efficient locomotion, we adapt the reward function from prior work on quadruped locomotion [16]. The locomotion policy is controlled at $50\,Hz$, and the simulator is operated at $200\,Hz$. Training is conducted in simulation using Isaac Sim with 4096 parallel environments for a total of 10,000 PPO iterations, corresponding to approximately 1 billion environment interactions. This policy is then used as the pre-trained low-level velocity controller in our setup, as shown in Figure 2.

3.2. Delta-Position Poilcy

We formulate the high-level controller as an RL policy that maps a target delta pose to velocity commands for the low-level controller to execute. This controller is trained using the same actor-critic framework and PPO algorithm as the low-level policy [22], but with smaller network architectures. Both the actor and critic are implemented as multi-layer perceptrons (MLPs) with hidden layer sizes of $[128, 64]$.

The policy receives as input the robot's current state and a delta pose command. The state includes the robot's base position and orientation in the world frame, base angular velocity, and the gravity vector projected into the body frame. The command specifies the desired relative displacement and orientation change as $(\Delta x, \Delta y, \Delta yaw)$, which is sampled uniformly within predefined bounds as shown in Equation 1. At each policy step, the remaining delta command is updated based on the robot's current position, enabling progress tracking toward the goal.

$$\begin{aligned} \Delta x &\sim \mathcal{U}(-3.0, 3.0)\,\text{m} \\ \Delta y &\sim \mathcal{U}(-3.0, 3.0)\,\text{m} \\ \Delta\text{yaw} &\sim \mathcal{U}(-180^\circ, 180^\circ) \end{aligned} \tag{1}$$

The policy outputs target linear and angular velocities in the body frame, which the pre-trained locomotion policy then tracks. The reward function used to train this high-level controller consists of two components: (i) task rewards that encourage accurate tracking of the desired displacement and heading, and (ii) regularization terms that penalize high action rates and joint torques to discourage abrupt or unstable motions.

This high-level policy runs at the same control frequency as the low-level controller ($50\,Hz$) and is trained in Isaac Sim with 4096 parallel environments for 1,000 PPO iterations, totaling approximately 30 million environment interactions. Once trained, this policy acts as the interface between the language model and the robot's control stack, converting natural language-derived goals into actionable motion commands, as illustrated in Figure 2.

To improve generalization beyond the training distribution, we implement a simple truncation mechanism for out-of-range commands. If the input delta pose exceeds the training bounds, it is decomposed into a sequence of sub-commands, each clipped to lie within the original training range. For instance, a 6-meter forward command in Δx is split into two consecutive 3-meter steps, allowing the policy to iteratively complete the full motion while operating within its learned limits.

3.3. Language to Delta-position commands

LLMs have the capability of learning spatial and behavioral patterns from large-scale text data. But their effectiveness diminishes when queried for robot control. They require clear and well-structured prompts to guide them to generate optimal outputs. Carefully engineered prompts can be useful to unlock LLM's reasoning abilities. Such an example can be seen in Figure 3. In this work, we design a prompt system that enables the LLM to map input natural language commands to output delta-position commands suitable for the quadruped's local position. The prompt is composed of the following parts:

1. **General Instruction Block:** This block explains the function of the LLM as a robot locomotion planner and outlines the desired behavior, which is to provide relative delta-position commands $(\Delta x, \Delta y, \Delta yaw)$ based on the robot's instantaneous position and a natural language instruction. Additionally, it ensures that all outputs are machine-readable and interpretable by enforcing format restrictions and output consistency.

2. **Delta Motion Definition Block:** This block gives the LLM the semantics of each output dimension in terms of rotation and spatial displacement. The forward or backward movement is represented by Δx, lateral motion Δy, and final pose around the vertical axis by Δyaw. All these outputs are given in the local body frame of the robot, and the values are clearly constrained to reflect physically feasible constraints for short-horizon legged motion.

3. **Input and Output Definition Blocks:** This block is to specify a structured schema with three scalar values $(\Delta x, \Delta y, \Delta yaw)$ as outputs and inputs that contain the robot's global pose and a free-form command.

4. **Example Block:** This section provides few-shot demonstrations of input-output pairs. This allows to cover a range of movement commands, including translations, rotations, and composite behaviors. Despite offering only a handful of examples, the LLM generalizes effectively to unseen and imprecise commands, demonstrating robust interpolation and extrapolation abilities.

4. Results

We evaluate the proposed language to the delta-position framework across four task categories. Each task is designed to asses the model's ability to generalize to the natural language command into meaningful delta-positon commands. For this project, we use **GPT-4o** as our primary inference model. All the experiments were conducted in simulations using the Unitree Go2 quadruped robot. We utilize the aforementioned delta-position hierarchical policy.

<General Instruction block>
You are an expert quadruped robot locomotion planner.
Your job is to predict the relative delta motion of the robot based on its current position and a natural language command.
You will always output the relative motion as **delta x**, **delta y**, and **delta yaw**, in that order.
You must ensure your response always follows the correct format, regardless of the input phrasing.

<Delta motion definition block>
The robot's relative motion is defined as follows:
1. Δx is the forward-backward displacement in meters (positive is forward).
2. Δy is the lateral (sideways) displacement in meters (positive is left).
3. Δyaw is the final change in heading (in degrees, positive is counter-clockwise).

- All displacements are measured relative to the robot's current body frame.
- Only the final relative pose is output, not intermediate steps.
- For turning in place or pivoting actions, Δx and Δy may be 0.
- Typical value ranges:
  -- Δx: [-3.0, 3.0] meters
  -- Δy: [-3.0, 3.0] meters
  -- Δyaw: [-180, 180] degrees

<Input format definition block>
- Each input will be in this format:
Current position: (x: <float>, y: <float>, yaw: <float>)
Instruction: <natural language command>

Where:
- x and y are the robot's global coordinates in meters
- yaw is the robot's current heading in degrees (0° means facing global +x)
- The instruction may describe movement goals, turns, or combination behaviors

<Output format definition block>
You must format your response exactly as follows:
- x: <float>
- y: <float>
- yaw: <float>

- All values must be floats (you may round to 2 decimal places).
- Use no units in the output, only numeric values.

<Examples block>
Input:
Current position: (x: 2.0, y: 3.0, yaw: 90.0)
Instruction: sidestep right quickly

Output:
x: 0.0
y: -1.0
yaw: 0.0

Input:
Current position: (x: -1.5, y: 0.0, yaw: 0.0)
Instruction: walk forward and turn left

Output:
x: 1.0
y: 0.0
yaw: 45.0

Input:
Current position: (x: 0.0, y: 0.0, yaw: 180.0)
Instruction: reverse slowly

Output:
x: -0.5
y: 0.0
yaw: 0.0

<User input block>
Generate a response based on the input provided below:
Input:
Current position: (x:-0.02 , y:-0.0 , yaw:-1.26)
Instruction: walk ahead fast

Output:

Figure 3. Complete prompt structure used in all experiments. The final 'User Input' block is modified per command, while the rest remains fixed.

4.1. Task Categories

We categorize our evaluation into the following four groups:

4.1.1 Minimal Prompt Context

The LLM is provided only with the general instruction and a natural language command, without delta-motion definitions or examples. In this under-specified setting, GPT-4o frequently produced invalid or unstructured outputs, failing to infer meaningful positional changes. This confirms that prompt completeness is critical for structured motion generation.

4.1.2 Full Prompt with Context Blocks

This setting includes the full structured prompt, including delta-motion semantics, input/output format, and example blocks. Under this configuration, GPT-4o performed best, generating correct $(\Delta x, \Delta y, \Delta\text{yaw})$ tuples for a wide range of commands, including "sidestep right quickly" or "walk forward and turn left." These results validate the efficacy of our prompt design.

4.1.3 Temporal Commands

In this setting, the instructions involved the duration and speed for the agent to follow (e.g., "walk forward for 5 seconds at 2 m/s"). GPT-4o partially understands these commands, but often struggles to compute precise displacements (e.g., outputting $\Delta x = 2.0$ instead of 10.0). Despite this, performance exceeds Task 1 and shows partial success in temporal grounding.

4.1.4 Vague or Ambiguous Commands

Finally, we test vague phrases such as "move a bit" or "go there quickly." Outputs in this category are less reliable and less consistent across trials, showing the LLM's tendency to hallucinate or over-interpret under-specified inputs. This

highlights the need for grounding mechanisms or language fine-tuning for safety-critical deployment.

4.2. LLM Comparison and Onboard Performance

The original goal of this project was to use the quantized FP16 version of the LLaMA 3.1–8B model [9] as the primary inference model, which would run locally on the robot's NVIDIA Jetson Orin Nano computer. The core idea was to fine-tune this model using supervised data generated from GPT-4o outputs. Which would enable a compact, locally deployable LLM that could replicate GPT-4o's instruction-following behavior without relying on external inference.

To begin this process, we manually collected approximately 100 high-quality instruction-response pairs by querying GPT-4o in order to avoid the costs associated with large-scale API usage. We performed a preliminary supervised fine-tuning run using this dataset on LLaMA 3.1–8B. However, the limited size of the dataset proved insufficient to meaningfully adapt the model, resulting in poor performance compared to GPT-4o.

Due to the time constraints and the resource-intensive nature of generating a sufficiently large and diverse dataset, we were unable to complete the fine-tuning pipeline. Hence, we reverted back to using GPT-4o as the primary inference model during evaluation. Despite this, the system architecture remains designed for onboard deployment, and future work will focus on scaling up data generation and completing the fine-tuning process to support complex real-time tasks such as humanoid manipulation.

5. Conclusion

In this project, we propose the use of Large Language Models (LLMs) to generate delta-position commands for quadruped locomotion based on natural language input. Our method leverages structured prompt design to enable models like GPT-4o to reason about spatial and temporal goals, providing a flexible and interpretable interface for controlling legged robots.

While our original goal was to fine-tune a locally deployable LLaMA 3.1–8B model using supervision from GPT-4o outputs, we were only able to test this approach on a small, manually curated dataset. We were unable to report quantitative metrics, as the plan relied on using GPT-4o as ground truth for performance comparison. Despite this, our framework lays the groundwork for future onboard deployment and extension to more complex manipulation tasks using language-driven control.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 2

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 2

[3] Arthur Bucker, Luis Figueredo, Sami Haddadin, Ashish Kapoor, Shuang Ma, Sai Vemprala, and Rogerio Bonatti. Latte: Language trajectory transformer. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 7287–7294. IEEE, 2023. 1, 2

[4] Elliot Chane-Sane, Joseph Amigo, Thomas Flayols, Ludovic Righetti, and Nicolas Mansard. SoloParkour: Constrained Reinforcement Learning for Visual Locomotion from Privileged Experience. In *Proceedings of The 8th Conference on Robot Learning*, pages 4268–4285. PMLR, Jan. 2025. ISSN: 2640-3498. 3

[5] Kun Chu, Xufeng Zhao, Cornelius Weber, and Stefan Wermter. LLM+MAP: Bimanual Robot Task Planning using Large Language Models and Planning Domain Definition Language, Mar. 2025. arXiv:2503.17309 [cs]. 1

[6] Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Oct. 2018. ISSN: 2153-0866. 3

[7] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023. 2

[8] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020. 2

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 6

[10] Markus Hafner, Maria Katsantoni, Tino Köster, James Marks, Joyita Mukherjee, Dorothee Staiger, Jernej Ule, and Mihaela Zavolan. Clip and complementary methods. *Nature Reviews Methods Primers*, 1(1):20, 2021. 2

[11] Fabian Jenelten, Junzhe He, Farbod Farshidian, and Marco Hutter. DTC: Deep Tracking Control. *Science Robotics*, 9(86):eadh5401, Jan. 2024. arXiv:2309.15462 [cs]. 2

[12] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots, July 2021. arXiv:2107.04034 [cs]. 3

[13] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, Oct. 2020. 3

[14] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang

Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal LLM Agents: Insights and Survey about the Capability, Efficiency and Security, May 2024. arXiv:2401.05459 [cs]. 1

[15] Ashish Malik, Stefan Lee, and Alan Fern. Interruptive language control of bipedal locomotion. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12394–12399. IEEE, 2024. 2

[16] Gabriel B. Margolis and Pulkit Agrawal. Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior. In *Proceedings of The 6th Conference on Robot Learning*, pages 22–31. PMLR, Mar. 2023. ISSN: 2640-3498. 3

[17] Gabriel B. Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, Apr. 2024. Publisher: SAGE Publications Ltd STM. 3

[18] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, Jan. 2022. Publisher: American Association for the Advancement of Science. 3

[19] Ruaridh Mon-Williams, Gen Li, Ran Long, Wenqian Du, and Christopher G. Lucas. Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, 7(4):592–601, Apr. 2025. Publisher: Nature Publishing Group. 1

[20] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, July 2018. Conference Name: IEEE Robotics and Automation Letters. 3

[21] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. In *Proceedings of the 5th Conference on Robot Learning*, pages 91–100. PMLR, Jan. 2022. ISSN: 2640-3498. 3

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, Aug. 2017. arXiv:1707.06347 [cs]. 3, 4

[23] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Ving: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021. 2

[24] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023. 2

[25] Yujin Tang, Wenhao Yu, Jie Tan, Heiga Zen, Aleksandra Faust, and Tatsuya Harada. Saytap: Language to quadrupedal locomotion. In *Conference on Robot Learning*, pages 3556–3570. PMLR, 2023. 1, 2

[26] Zhenyu Wu, Kun Zheng, Zhiyang Ding, and Hongbo Gao. A survey on legged robots: Advances, technologies and applications. *Engineering Applications of Artificial Intelligence*, 138:109418, Dec. 2024. 2

[27] Shaohang Xu, Lijun Zhu, Hai-Tao Zhang, and Chin Pang Ho. Robust Convex Model Predictive Control for Quadruped Locomotion Under Uncertainties. *IEEE Transactions on Robotics*, 39(6):4837–4854, Dec. 2023. 3

[28] Zifan Xu, Amir Hossain Raj, Xuesu Xiao, and Peter Stone. Dexterous Legged Locomotion in Confined 3D Spaces with Reinforcement Learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11474–11480, May 2024. 3

[29] Zhitong Zhang, Honglei An, Qing Wei, and Hongxu Ma. Learning-Based Model Predictive Control for Quadruped Locomotion on Slippery Ground. In *2022 4th International Conference on Control and Robotics (ICCR)*, pages 47–52, Dec. 2022. 2