

Commonsense Reasoning for Legged Robot Adaptation with Vision-Language Models

Annie S. Chen^{*1}, Alec M. Lessing^{*1}, Andy Tang^{*1}, Govind Chada^{*1}, Laura Smith², Sergey Levine², Chelsea Finn¹
Stanford University¹, UC Berkeley²

Abstract—Legged robots are physically capable of navigating a diverse variety of environments and overcoming a wide range of obstructions. For example, in a search and rescue mission, a legged robot could climb over debris, crawl through gaps, and navigate out of dead ends. However, the robot’s controller needs to respond intelligently to such varied obstacles, and this requires handling unexpected and unusual scenarios successfully. This presents an open challenge to current learning methods, which often struggle with generalization to the long tail of unexpected situations without heavy human supervision. To address this issue, we investigate how to leverage the broad knowledge about the structure of the world and commonsense reasoning capabilities of vision-language models (VLMs) to aid legged robots in handling difficult, ambiguous situations. We propose a system, VLM-Predictive Control (VLM-PC), combining two key components that we find to be crucial for eliciting on-the-fly, adaptive behavior selection with VLMs: (1) in-context adaptation over previous robot interactions and (2) planning multiple skills into the future and replanning. We evaluate VLM-PC on several challenging real-world obstacle courses, involving dead ends and climbing and crawling, on a Go1 quadruped robot. Our experiments show that by reasoning over the history of interactions and future plans, VLMs enable the robot to autonomously perceive, navigate, and act in a wide range of complex scenarios that would otherwise require environment-specific engineering or human guidance.

I. INTRODUCTION

Robots deployed in open-world environments must be able to handle highly unstructured and complicated environments. This is particularly the case for legged robots, which may need to operate in an extremely diverse range of circumstances. Consider a quadruped robot tasked with performing search and rescue in a collapsed building. This robot faces a long tail of different possible environments and obstacles, which might require climbing over debris, crawling through gaps, and backtracking and navigating out of dead ends without a map. Handling these diverse real-world scenarios autonomously, without detailed human guidance and specific skill directives, remains a significant challenge. Prior work in locomotion has endowed legged robots with agile skills like running, climbing, and crawling [1], [2], but possessing these skills alone does not solve the problem of fully autonomous deployment. To handle complex, unstructured scenarios, a robot must be able to decide how to deploy its repertoire of skills with a nuanced understanding of its situation. Consider the example of clearing a novel obstacle like debris in a collapsed building. We expect an intelligent robot to perceive and try a skill that

is likely to succeed, e.g., climbing. If the robot’s attempt was unsuccessful, e.g. the debris is too slippery to climb over, the robot should recognize this and try another strategy given the information it has gathered, e.g. backtrack and try a new strategy like finding a path around the log instead.

Foundation models such as vision-language models (VLMs) have the potential to help robots handle real-world, unstructured scenarios, since they possess commonsense knowledge acquired from diverse internet-scale image and language data. Indeed, multiple prior works have shown how robots can leverage knowledge in large language models (LLMs) and VLMs for high-level planning [3]–[5] in robotic manipulation. In principle, VLMs can provide high-level semantic knowledge for legged robots as well, e.g. by identifying obstacles or selecting high-level behaviors. However, the use of LLMs and VLMs has been far more limited in legged robots, compared to robotic manipulation. Furthermore, in diverse, complex environments with obstacles where the robot may get stuck and need to try multiple strategies to overcome, naively prompting a VLM to output a skill may often fail due to inaccuracies in the model’s interpretation of the scene and subsequent inability to adapt with the robot’s environment interactions.

In this work, we investigate how legged robots can leverage VLMs and their general knowledge about the structure of the world and commonsense reasoning capabilities to suggest contextually informed behaviors based on visual inputs. We find the following two key insights crucial for facilitating adaptive behavior selection in complex, unstructured settings with VLMs: (1) The robustness of VLMs in novel situations can be greatly improved by taking into account the robot’s interaction history, leveraging chain-of-thought reasoning [6], [7], and (2) Prompting the model to plan multiple skills ahead and optionally replan at each timestep is essential for foreseeing potential failures.

Combining these insights, we propose VLM Predictive Control (VLM-PC), which can be seen as a history-conditioned high-level analogue of visual model predictive control [8]–[11] in skill space. With an image of the robot’s view along with the history of interactions as input, the VLM is prompted to generate a multi-step plan of skills. In order to choose what plan to follow, and ultimately what skill to execute next, the model is prompted to reason through the robot’s current state and whether the previous existing plan made progress on the desired task and re-plan if needed.

In our experiments, we find that leveraging VLMs in

* equal contribution. Correspondence to asc8@stanford.edu. See <https://anniesch.github.io/vlm-pc/> for videos and full paper.

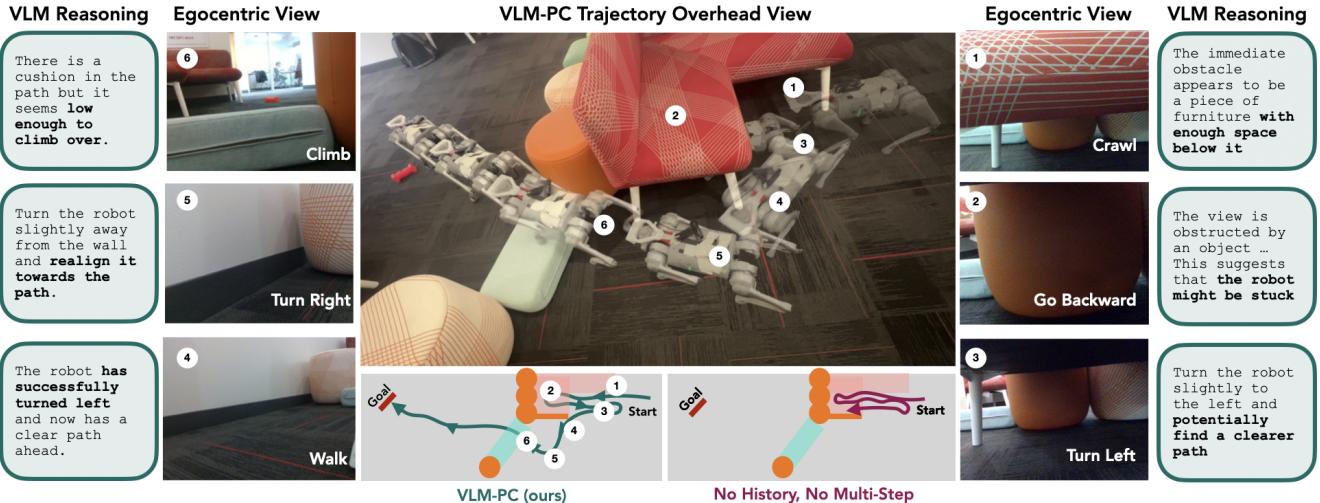


Fig. 1: Vision-Language Model Predictive Control (VLM-PC) enables real-world locomotion adaptation. By leveraging the commonsense reasoning abilities of pre-trained VLMs to adaptively select behaviors, VLM-PC allows legged robots to quickly adjust strategies when encountering a wide range of situations, even backtracking when appropriate. **Center:** An example trajectory of the robot tasked with finding the red chew toy amid obstacles using VLM-PC—it first crawls under a couch, then backs out of it when it finds it is a dead end, turns to walk around the couch, climbs over a sizeable cushion, and finally locates the toy. **Bottom left:** An overhead view of the trajectory with VLM-PC. **Bottom right:** An example trajectory of the robot’s behavior using a VLM naively, where the robot gets stuck and cannot adapt. **Left and right:** Visualization of the robot’s egocentric POV that is provided to the VLM at different points along the trial along with excerpts of reasoning with VLM-PC at those points.

this way allows a Go1 robot to handle a range of real-world situations that have not been tackled by prior work in a fully autonomous manner. Across five challenging real-world settings, one of which is shown in Figure 1, our approach completes the target task around 30% more successfully by leveraging in-context adaptation and multi-step planning. Our results show that with in-context adaptation and multi-step planning, VLM-PC can aid robot grounding for locomotion, and that even without detailed robot grounding, which is challenging to provide in locomotion, our approach enables the robot to understand its environment enough to autonomously traverse complex environments.

II. RELATED WORK

Our work tackles the issue of enabling legged robots to perform robustly in unstructured, unknown test-time conditions. Traditional model-based control approaches have achieved impressive agile locomotion [1], [12]–[17] but are not well-equipped to navigate arbitrary, open-world environments. Learning-based approaches hold the promise of greater generalization capabilities, and training a single policy with reinforcement learning (RL) has also demonstrated successful low-level locomotion capabilities from robust walking to jumping and bipedal walking [18]–[31]. Behind a majority of these successes is the use of domain randomization [19], [21], [32]–[40], which involves training the robot under a variety of different dynamics to robustify the policy. Our work tackles an orthogonal, complementary problem: enabling legged robots to autonomously solve complex, partially observed tasks given a repertoire of low-level skills (which can be acquired through either traditional model-based approaches or RL training). Using these skills to solve a long-horizon

task requires understanding the scene and reasoning over the information gathered in the environment, trying different low-level strategies, and adapting high-level plans on-the-fly accordingly.

Prior work has also explored utilizing a repertoire of skills to help legged robots navigate that require a combination of distinct behaviors. For example, [41] train a policy that uses human input via remote control to select skills, while others have explored using learned models to choose appropriate behaviors on-the-fly, e.g., using search in latent space [21], [25], [42], direct inference using proprioceptive history [22], [43], [44], prediction based on egocentric depth [24], [30], [45], [46], or using value functions [47]. These works rely on human supervision or domain-specific information required to train model-based behavior selection. In contrast, our approach represents the robot’s range of skills in language and studies how to leverage this representation with pre-trained VLMs using in-context reasoning to adapt on-the-fly in complex scenarios.

Outside legged locomotion, extensive research has explored combining prior behaviors to address long-horizon tasks, often by training high-level policies that orchestrate learned skills into complex behaviors [48]–[57]. Natural language provides a simple abstraction to index these behaviors, and using language as an abstraction for behaviors provides an interpretable space for a high-level planner to select strategies to try [3]–[5], [58]–[62] or to generate robot code [63], [64]. In-context reasoning with LLMs has refined low-level behaviors [65]–[68], improved planning with feedback [58] and facilitated learning from human feedback [69], [70], but these do not incorporate VLMs, which can offer rich

multimodal understanding. Recent works have begun going beyond LLMs and incorporating VLMs for manipulation [71]–[73] and navigation [74], [75]. Unlike these works, we focus particularly on equipping the robot to handle unpredictable situations where it might get stuck and need to explore different strategies to make progress. Enabling this in a diverse array of environments requires robust commonsense reasoning abilities, and we study the extent to which VLMs can provide these for legged robots.

While high-level planning in language grounding has been studied for manipulation or navigation tasks, it has explored far less for legged locomotion. Key works have interfaced through foot contact patterns [76] or code [77] with LLM planning. Our work implements a straightforward language-skill interface for locomotion and is the first to explore how legged robots can utilize the commonsense reasoning capabilities of pre-trained VLMs to autonomously guide adaptive behavior selection. In particular, the vast majority of prior works apply LLMs zero-shot based on the *current instruction or observation*; in contrast, we demonstrate that in-context adaptation over the *history of observations* plays a crucial role in compensating for the robot’s lack of grounding in locomotion.

III. PROBLEM STATEMENT

We assume the robot has access to a set of n skills, which are sufficient to allow the robot to traverse the environment. Given the recent development of highly robust low-level quadrupedal locomotion controllers via RL [22], [31], [41], we believe this assumption to be reasonable for a wide variety of real-world scenarios. For example, if the robot has the ability to climb, crawl, walk forwards and backwards, and turn in various directions, we expect that it could sequentially apply these skills to handle a variety of situations – the challenge then is to determine when to deploy each skill to navigate an unseen, unstructured situation. Each skill corresponds to a policy π_i , which takes in a state $s \in \mathcal{S}$ and outputs a low-level action u . At test time, the robot interacts in a partially observed environment, where it receives images $\{I\}$ and must process them and output a skill and time duration δ that executes policy π_i for an amount of time δ .

We frame our problem setting as an instantiation of single-life deployment [78], where the agent has prior behaviors and is evaluated on a task during a “single-life” trial without any human intervention. This setting is meant to be representative of real settings in which a robot is autonomously deployed without prior knowledge of the environment or any human guidance available. In our experimental settings with legged locomotion, this corresponds to completing a task (e.g. finding an object) by moving in a desired direction while successfully overcoming any obstacles in the terrain.

IV. VISION-LANGUAGE MODEL PREDICTIVE CONTROL (VLM-PC)

Our goal is to enable legged robots to make informed decisions that lead to effective, context-aware adaptation to help navigate these situations autonomously and successfully. Our central hypothesis is that many real-world situations

demand complex reasoning due to unexpected circumstances that may be difficult to generalize to. In this section, we first describe how we represent the robot’s skills via language to be used then by a pre-trained VLM. We then detail how we prompt the VLM to reason through the robot’s current state and history of interactions to select the next skill to execute to solve a task in unstructured environments.

A. Interfacing Robotic Locomotion Skills with VLMs

We consider generative VLMs, also known as multimodal language models, which take as input $\{I, x\}$, including images $\{I\}$ and prompt text x and outputs text y from a distribution over textual completions $P(\cdot | \{I\}, x)$. We label each of the robot’s prior behaviors $\pi_i \in \Pi$ with a command l_i , a textual description of the corresponding behavior. We also define levels of magnitude m for each that define the duration $\delta_{l_i, m}$ that π_i should be executed. While there are many ways to acquire locomotion policies, e.g., via traditional model-based techniques or learning-based approaches, we use the built-in controller provided by the Go1 robot. To make these policies amenable to being used by a VLM, we choose to represent the policies as skills (as opposed to less interpretable, low-level actions such as joint angles or foot contact patterns), the details of which can be found in the Appendix on our website. For example, l_i could be “Climb” or “Crawl”, and m is “Small”, “Medium”, or “Large”. At timestep T , the VLM is prompted to output high-level action $a_T = (l_i, m)$, which leads to the robot executing low-level actions $u_t = \pi_i(s_t; m_T)$ for the number of seconds dictated by m_T . Through prompt engineering, we ensure that the VLM outputs the skill and magnitude in a specified format that allows us to extract the high-level skill command for the robot to execute.

B. Using VLMs for Adaptive Behavior Selection

We propose a system, Vision-Language Model Predictive Control (VLM-PC), that uses a VLM to account for these errors and successively refine strategies, so that the robot can autonomously adjust from strategies that fail and try others. Summarized in Figure 2, VLM-PC combines two key insights to effectively enable VLMs to serve as an effective high-level policy: (1) reasoning about information gathered by the robot in its environment and (2) selecting actions by planning ahead and iteratively replanning during execution. The VLM we use in all of our experiments is GPT-4o. We tuned the prompts to take into account the setting of legged locomotion and the limited view from the robot’s camera. Full prompts and an example log of the VLM’s chats are shown in the Appendix on our project website: <https://anniesch.github.io/vlm-pc/>.

a) Using in-context reasoning to adapt on-the-fly.

We want the VLM to reason through the history in the environment and the progress made with the commanded skills before deciding on the next skill, in order to determine if the robot should try a new strategy. As such, we include as input to the VLM an image representing the robot’s current view along with the full history of interactions (including the robot’s previous images and the previous outputs of the VLM) and a prompt, i.e. the input at timestep t is $(I_1, x_1, y_1, I_{\delta_{m_1}}, x_{\delta_{m_1}}, y_{\delta_{m_1}}, \dots, x_{t-1}, I_t, p_t)$, which contains for each previous query step i , each previous image I_i and prompt x_i along with VLM output y_i . We then prompt the VLM to first

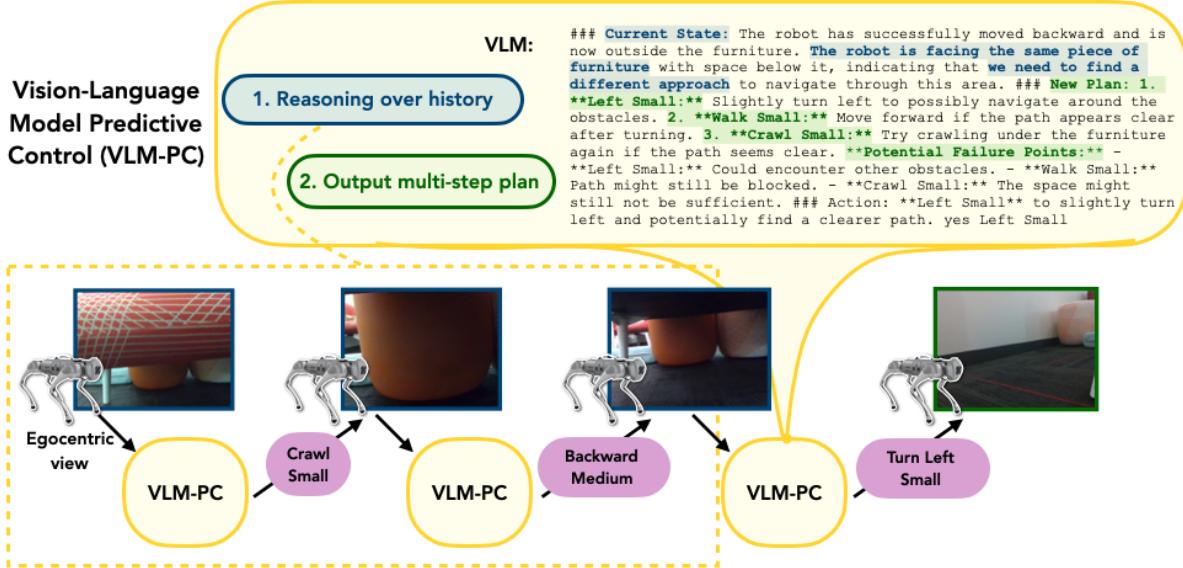


Fig. 2: **Vision-Language Model Predictive Control (VLM-PC)**. Our method uses a pre-trained VLM to provide high-level skill commands for a legged robot to execute. Given the robot’s current view and history of interactions, the VLM is first prompted to reason through the robot’s current state and progress with the history of commanded skills, and is then prompted to make a new multi-step plan, compare it to the prior plan, and adjust if needed. The robot executes the first skill in the plan, and the VLM is queried again.

reason through what progress the robot has made using the history of commands selected and the current position and orientation of the robot.

b) Multi-step planning and execution.

Due to partial observability, there is often no clear answer as to which skill is most appropriate for a given situation, so planning multiple skills ahead and optionally replanning at each timestep may help with foreseeing and recovering from unexpected outcomes/failures. So, we use an approach akin to model predictive control [8]–[11], wherein we prompt the VLM to produce the immediate skill to execute by planning multiple steps $l_t, l_{t+\delta_m}, \dots, l_{t+k}$, into the future and reasoning about the consequences of the actions. This allows the VLM to foresee different possible strategies that might be applicable to the current situation, so it may better adjust in the future if the next chosen skill does not make progress. To implement this, we specifically prompt the VLM to make a multi-step plan taking into account the latest visual observation I_t , compare the new plan to the prior existing plan, and use the one that seems more applicable.

V. EXPERIMENTAL RESULTS

In this section, we study whether VLM-PC can enable a Go1 quadruped robot to tackle five challenging real-world situations in a fully autonomous manner. Concretely, we aim to answer the following empirical questions: (1) Can VLM-PC enable the robot to autonomously adapt in unseen, partially observed environments and effectively complete tasks that require reasoning over what strategies the robot has tried in the past? (2) How much do in-context reasoning over the robot’s experience and multi-step planning affect the robot’s ability to complete these test settings? (3) Does including additional in-context examples improve the robot’s ability to handle the given setting? We first describe simulated results before the main real world results of our experiments. For qualitative video results, see our project webpage: <https://anniesch.github.io/vlm-pc/>.

A. Real-World Experimental Setup

We use a Go1 quadruped robot from Unitree. The robot is equipped with an Intel Realsense D435 camera mounted on its

head, which provides an egocentric view of the environment, which is the only source of information the robot has about its surroundings. We configure the default controller to correspond to a set of prior behaviors: walking forward, crawling (at a low height), climbing (which can overcome stair-height obstacles), walking backward, turning left, and turning right. This same set of behaviors is used for all experiments, and details of the skills are in Appendix C. In each setting, we report the average and median wall clock time in seconds (where lower is better) needed to complete the task along with the success rate across five trials for each method. If the robot does not complete the task within 100 seconds of executing actions, we consider it a failure. For each method in each setting, we report these metrics across five trials.

a) Evaluation Settings.

To evaluate each method, we conduct trials in five real-world indoor and outdoor settings. The settings test the robot’s ability to adapt to varying terrain conditions, requiring agile skills and dynamic strategy adjustments based on new information. The goal in each setting is to reach the “red chew toy”. The robot only receives information from its camera and does not have access to a map of the environment. The tasks are shown in Figure 3, annotated with the goal and an example path through the course, and described as follows:

Indoor 1: The robot first must crawl under a couch, determine that it is a dead end, back up and turn to walk around the couch, climb a cushion it cannot pass without climbing, and finally locate the toy.

Indoor 2: The robot first faces a couch that it must crawl under to the opposite side, then faces several stools blocking its path with a narrow gap between them, and must determine that it cannot fit through the gap and must turn and go around to locate the red chew toy.

Outdoor 1: The robot first faces bushes that it must turn from and go around, then faces a series of small logs that it must climb over, and finally locates the red chew toy.

Outdoor 2: The robot first faces a curb that it must climb over, a dirt hill that it must walk up, a wooden plank that it must climb over, and finally locate the red chew toy between the bushes.

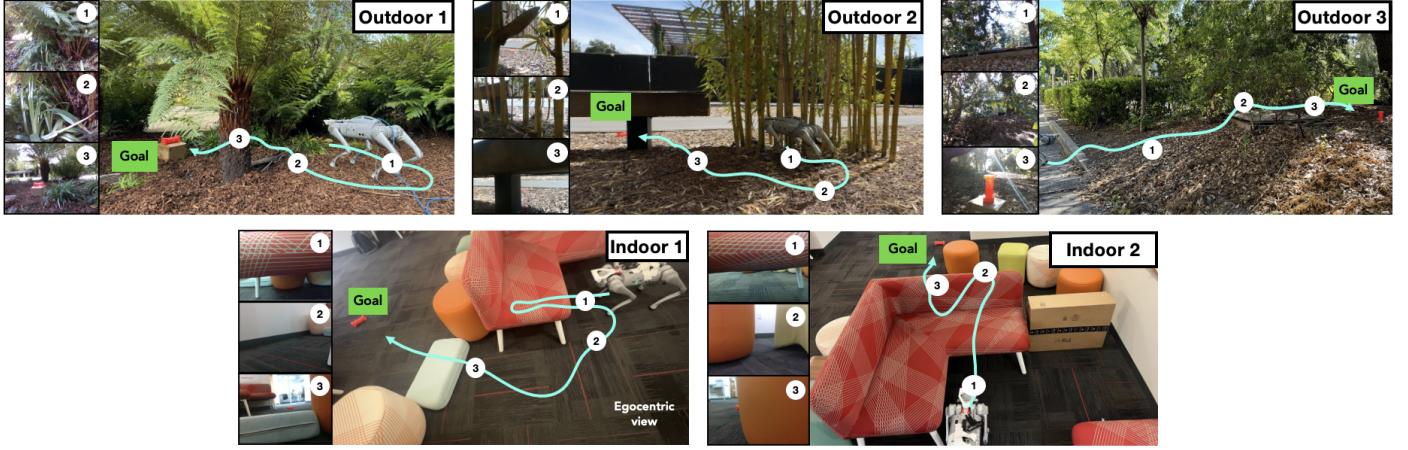


Fig. 3: **Deployment Environments.** We evaluate VLM-PC on five challenging real-world settings, each of which presents unseen obstacles designed for the robot to get stuck, and requires commonsense reasoning to solve. For each setting, we give a third-person view of the obstacle course as well as an example path through the course, with three different egocentric views (labeled 1, 2, 3) at different points to show the diversity of scenes the robot encounters from its viewpoint.

Method	Outdoor 1			Outdoor 2			Outdoor 3		
	Avg (s) ↓	Median (s) ↓	Success (%)	Avg (s) ↓	Median (s) ↓	Success (%)	Avg (s) ↓	Median (s) ↓	Success (%)
Random	84.2	100	20	92	100	20	100	100	0
No History	82	100	20	100	100	0	49.6	17.1	60
No Multi-Step	81.5	100	20	100	100	0	57.4	42	60
VLM-PC	49.4	17	60	68.8	65.5	60	61.7	50.5	60

Method	Indoor 1			Indoor 2		
	Avg (s) ↓	Median (s) ↓	Success (%)	Avg (s) ↓	Median (s) ↓	Success (%)
Random	100	100	0	93.9	100	20
No History	100	100	0	87.9	100	20
No Multi-Step	57.2	34	60	82.9	100	40
VLM-PC	66.7	46.7	60	37.1	35.3	80

TABLE I: **Results on Each Setting.** We report the average and median time to complete the task (where lower is better) and success rate, across five trials for each method in each of the five settings. VLM-PC far outperforms the comparisons across all metrics in three of the scenes (Outdoor 1, Outdoor 2, and Indoor 2) and is comparable to the best other method in the other two scenes. Furthermore, VLM-PC is the only method that succeeds a majority of the time in each setting.

b) Comparisons.

We compare VLM-PC to several variants that differ in the amount of context provided to the VLM and the amount the VLM is prompted to plan: (1) **No History**: The VLM is prompted with only the current image and the prompt, and is not provided with any history of interactions but is still prompted to output a multi-step plan of skills at each timestep. (2) **No Multi-Step**: The VLM is prompted with the full history of interactions, including the robot’s previous images and the previous outputs of the VLM, but is only prompted to plan a single skill at each timestep. (3) **VLM-PC**: The VLM is prompted with the full history of interactions, including the robot’s previous images and the previous outputs of the VLM, and is prompted to make a multi-step plan of skills at each timestep. As a baseline, we additionally compare to (4) **Random**, which randomly selects a skill and magnitude to execute at each timestep.

B. Main Results

As shown in Figure 8, on average across all five settings, VLM-PC successfully completes the task 64% of the time, almost 30% more than the second best method (No Multi-Step), which succeeds on average 36% of the time. VLM-PC is also on average over 20% faster at completing the target task as the next best method, showing

that including both history and multi-step planning are important for improving the use of these VLMs in providing high-level commands in a variety of settings. In Figure 3, we find that particularly on Indoor 2, Outdoor 1, and Outdoor 2, VLM-PC is more than twice as successful as the next best method. No Multi-Step is the second best method, and does comparably to VLM-PC (which does multi-step planning) on Indoor 1 and Outdoor 3, indicating that in some situations, multi-step planning does not significantly help, although it does not hurt performance. No History fails in almost every setting except Outdoor 3, as it often gets stuck behind obstacles that require trying multiple different strategies. Random fails in every setting, showing that each setting requires nontrivial reasoning for the robot to succeed. One key insight from these results is the finding that in-context adaptation aids robot grounding for locomotion, allowing the robot to ground sufficiently well to achieve a high success rate at the given tasks, even without detailed understanding of each skill.

C. Adding Labeled In-Context Examples Can Improve Performance

As large foundation models trained on Internet-scale data are used as these high level planners, they can leverage in-context learning, where examples or instructions are included as context in

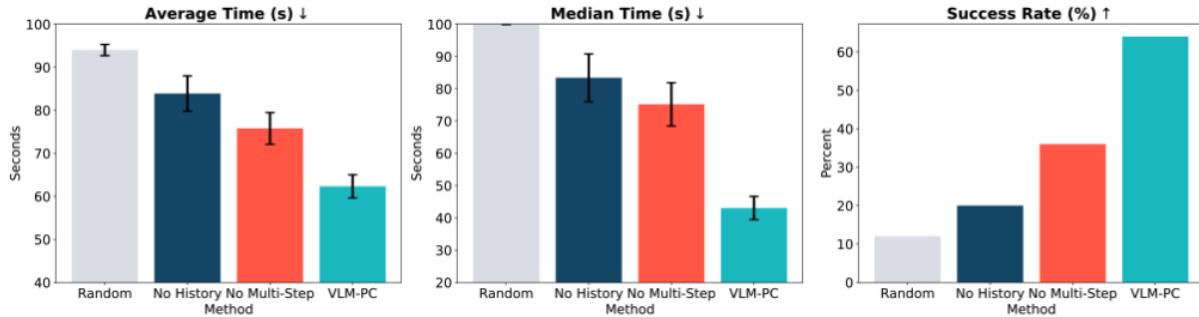


Fig. 4: **Main Results Averaged Across Settings.** Across all five settings on average, VLM-PC significantly outperforms Random, No History, and No Multi-Step on average and median time to complete the task and success rate, performing roughly 30% more successfully than the next best method.

Course	Method	Avg time (s)	Median time (s)	Success Rate (%)
Indoor 2	VLM-PC	37.1	35.3	80
	VLM-PC + IC	13.5	13.7	100
Outdoor 1	VLM-PC	49.4	17.0	60
	VLM-PC + IC	10.0	10.0	100

Fig. 5: **VLM-PC with Labeled In-Context Examples.** We find that in two of the obstacle courses, leveraging the VLM’s in-context learning capabilities by providing additional images labeled with the best command can significantly improve performance.

the input to the model [79], [80]. We provide an extension of our method including in-context examples, called VLM-PC+IC, where we include in the first prompt several additional images, taken from the egocentric view at different points in the environment, as well as a label for each of them with the best command to take. This provides the VLM with more context about the environment and the best strategies to take at key points. As shown in Table 5, we find that in two of the obstacle courses, this can significantly improve performance. While inexpensive to obtain, this does require human labeling of several images from the deployment environment with the best command to take, which may not be feasible in all deployment settings. Nonetheless, this extension further reinforces the importance of providing useful context to the VLM and having it use this context to make informed decisions, and shows that these labeled examples can be useful context on top of the history of experiences in the environment.

D. Results in a Controlled Simulated Setting

In order to more easily compare against more methods, we additionally evaluate in a controlled simulated setting, built off of the legged_gym repo by [81], where the robot finds itself in a dead end, must back up out of it, turn and walk up stairs, shown in Figure 8. We give the robot the same skills as in the real world setting. In addition to comparing to No History and No Multi-Plan, we compare to two additional methods: (1) High-Level Classifier (HLC), which trains a classifier on the oracle skill to use in different obstacles and uses it to select which skill to use at a given state, similar to work by [82], and (2) Distillation [30], [31] from oracle actions in training environments. Note that both of these methods require extra human labeling on oracle skills at different states during training and are challenging to implement effectively on the real robot.

In Table IV, we find that similar to our conclusions in our real-world experiments, VLM-PC significantly outperforms both existing methods. In particular, we find that VLM-PC is able to reason through to solve the task much more effectively than HLC and Distillation, despite requiring less human supervision, as it is able to reason over previous attempts and efficiently recover from mistakes. For more details on this setting, see the Appendix on our website.

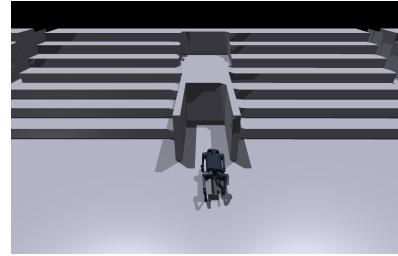


Fig. 6: **Simulated Evaluation Environment.** We evaluate VLM-PC along with several comparisons from prior work in a challenging simulated setting, where the robot must back out or avoid the dead end, turn accordingly, and climb the stairs.

Method	Random	No-Hist	No-Plan	VLM-PC	HLC	Distill
Success (%)	7.1	28.6	50	64.3	28.6	7.0

TABLE II: **Results in Simulation.** VLM-PC outperforms all comparisons in our controlled simulated setting.

VI. DISCUSSION AND LIMITATIONS

We introduced Vision-Language Model Predictive Control (VLM-PC), which enables legged robots to rapidly adapt to changing, unseen circumstances during deployment. On a Go1 quadruped robot, we find that VLM-PC can autonomously handle a range of complex real-world tasks involving climbing over obstacles, crawling under furniture, and navigating around dead ends and through cluttered environments. While VLM-PC is promising solution for enabling legged robots to handle new tasks, there remains much left to explore regarding how to best leverage VLMs for adaptive behavior for legged robots, especially as core VLM capabilities continue to improve. First, improving language grounding for locomotion to better capture the nuances of the robot’s capabilities could lead to more effective decision-making. It would also be interesting to explore if fine-tuning these VLMs, perhaps with techniques like reinforcement learning from human feedback, can lead to more efficient reasoning. Incorporating more sensors or scene reconstruction could provide a more comprehensive view of the environment, allowing the VLM to reason more effectively. Finally, it would be interesting to explore how to use VLMs to combine high-level planning for locomotion with that for manipulation, to enable robots to handle a wider range of tasks.

ACKNOWLEDGMENTS

We thank Zipeng Fu and Yunhao Cao for help with hardware, and Jonathan Yang, Anikait Singh, and other members of the IRIS lab for helpful discussions on this project. This work was supported by an NSF CAREER award, ONR grant N00014-21-1-2685 and N00014-22-1-2621, Volkswagen, and the AI Institute, and an NSF graduate research fellowship.

REFERENCES

- [1] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, “Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot,” *Autonomous robots*, vol. 40, pp. 429–455, 2016.
- [2] 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.
- [3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” *arXiv preprint arXiv:2204.01691*, 2022.
- [4] 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.
- [5] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, “Palm-e: An embodied multimodal language model,” *arXiv preprint arXiv:2303.03378*, 2023.
- [6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.
- [7] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large language models are zero-shot reasoners,” *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.
- [8] C. E. Garcia, D. M. Prett, and M. Morari, “Model predictive control: Theory and practice—a survey,” *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
- [9] M. Morari and J. H. Lee, “Model predictive control: past, present and future,” *Computers & chemical engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.
- [10] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.
- [11] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, “Visual foresight: Model-based deep reinforcement learning for vision-based robotic control,” *arXiv preprint arXiv:1812.00568*, 2018.
- [12] H. Dai, A. Valenzuela, and R. Tedrake, “Whole-body motion planning with centroidal dynamics and full kinematics,” in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.
- [13] M. Hutter, C. Gehring, D. Jud, A. Lauber, D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Höpflinger, “Anymal - a highly mobile and dynamic quadrupedal robot,” pp. 38–44, 2016.
- [14] H.-W. Park, P. M. Wensing, and S. Kim, “High-speed bounding with the mit cheetah 2: Control design and experiments,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017. [Online]. Available: <https://doi.org/10.1177/0278364917694244>
- [15] D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, “Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots,” *IEEE Robotics and Automation Letters*, vol. 3, pp. 2261–2268, 2018.
- [16] G. Bledt, M. J. Powell, B. Katz, J. Carlo, P. Wensing, and S. Kim, “Mit cheetah 3: Design and control of a robust, dynamic quadruped robot,” pp. 2245–2252, 2018.
- [17] B. Katz, J. Carlo, and S. Kim, “Mini cheetah: A platform for pushing the limits of dynamic quadruped control,” *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6295–6301, 2019.
- [18] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, “Learning to walk via deep reinforcement learning,” *arXiv preprint arXiv:1812.11103*, 2018.
- [19] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, “Sim-to-real: Learning agile locomotion for quadruped robots,” *arXiv preprint arXiv:1804.10332*, 2018.
- [20] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, “Data efficient reinforcement learning for legged robots,” *Conference on Robot Learning*, vol. abs/1907.03613, 2019.
- [21] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, “Sim-to-real transfer for biped locomotion,” in *2019 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2019, pp. 3503–3510.
- [22] 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.
- [23] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang, “Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers,” *arXiv preprint arXiv:2107.03996*, 2021.
- [24] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, “Legged locomotion in challenging terrains using egocentric vision,” in *Conference on Robot Learning*, 2022.
- [25] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, “Learning agile robotic locomotion skills by imitating animals,” *arXiv preprint arXiv:2004.00784*, 2020.
- [26] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, “Advanced skills by learning locomotion and local navigation end-to-end,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.
- [27] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, “Legged robots that keep on learning: Fine-tuning locomotion policies in the real world,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1593–1599.
- [28] K. Caluwaerts, A. Iscen, J. C. Kew, W. Yu, T. Zhang, D. Freeman, K.-H. Lee, L. Lee, S. Saliceti, V. Zhuang *et al.*, “Barkour: Benchmarking animal-level agility with quadruped robots,” *arXiv preprint arXiv:2305.14654*, 2023.
- [29] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, “Agile but safe: Learning collision-free high-speed legged locomotion,” *arXiv preprint arXiv:2401.17583*, 2024.
- [30] Z. Zhuang, Z. Fu, J. Wang, C. G. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, “Robot parkour learning,” in *7th Annual Conference on Robot Learning*, 2023.
- [31] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, “Extreme parkour with legged robots,” *arXiv preprint arXiv:2309.14341*, 2023.
- [32] M. Cutler, T. J. Walsh, and J. P. How, “Reinforcement learning with multi-fidelity simulators,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3888–3895.
- [33] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, “Epopt: Learning robust neural network policies using model ensembles,” *arXiv preprint arXiv:1610.01283*, 2016.
- [34] F. Sadeghi and S. Levine, “Cad2rl: Real single-image flight without a single real image,” *arXiv preprint arXiv:1611.04201*, 2016.
- [35] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.
- [36] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.
- [37] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, “Solving rubik’s cube with a robot hand,” *arXiv preprint arXiv:1910.07113*, 2019.
- [38] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg, “Dynamics randomization revisited: A case study for quadrupedal locomotion,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4955–4961.
- [39] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, “Rapid locomotion via reinforcement learning,” *arXiv preprint arXiv:2205.02824*, 2022.
- [40] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner *et al.*, “Learning agile soccer skills for a bipedal robot with deep reinforcement learning,” *arXiv preprint arXiv:2304.13653*, 2023.
- [41] 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.
- [42] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, “Learning fast adaptation with meta strategy optimization,” *IEEE Robotics and Automation Letters*, 2020.
- [43] A. Kumar, Z. Fu, D. Pathak, and J. Malik, “Rma: Rapid motor adaptation for legged robots,” *arXiv preprint arXiv:2107.04034*, 2021.
- [44] Z. Fu, X. Cheng, and D. Pathak, “Deep whole-body control: learning a unified policy for manipulation and locomotion,” in *Conference on Robot Learning*, 2022.
- [45] 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*, 2022.

- [46] R. Yang, G. Yang, and X. Wang, “Neural volumetric memory for visual locomotion control,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [47] A. S. Chen, G. Chada, L. Smith, A. Sharma, Z. Fu, S. Levine, and C. Finn, “Adapt on-the-go: Behavior modulation for single-life robot deployment,” *arXiv preprint arXiv:2311.01059*, 2023.
- [48] P.-L. Bacon, J. Harb, and D. Precup, “The option-critic architecture,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.
- [49] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine, “Mcp: Learning composable hierarchical control with multiplicative compositional policies,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [50] Y. Lee, J. Yang, and J. J. Lim, “Learning to coordinate manipulation skills via skill behavior diversification,” in *International conference on learning representations*, 2019.
- [51] M. Sharma, J. Liang, J. Zhao, A. LaGrassa, and O. Kroemer, “Learning to compose hierarchical object-centric controllers for robotic manipulation,” *arXiv preprint arXiv:2011.04627*, 2020.
- [52] R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, and C. Schmid, “Learning to combine primitive skills: A step towards versatile robotic manipulation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4637–4643.
- [53] O. Nachum, S. S. Gu, H. Lee, and S. Levine, “Data-efficient hierarchical reinforcement learning,” *Advances in neural information processing systems*, vol. 31, 2018.
- [54] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta, “Efficient bimanual manipulation using learned task schemas,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1149–1155.
- [55] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, “Guided reinforcement learning with learned skills,” *arXiv preprint arXiv:2107.10253*, 2021.
- [56] M. Dalal, D. Pathak, and R. R. Salakhutdinov, “Accelerating robotic reinforcement learning via parameterized action primitives,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 847–21 859, 2021.
- [57] S. Nasiriany, H. Liu, and Y. Zhu, “Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7477–7484.
- [58] 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.
- [59] M. MacMahon, B. Stankiewicz, and B. Kuipers, “Walk the talk: Connecting language, knowledge, and action in route instructions,” *Def.*, vol. 2, no. 6, p. 4, 2006.
- [60] D. K. Misra, J. Sung, K. Lee, and A. Saxena, “Tell me dave: Context-sensitive grounding of natural language to manipulation instructions,” *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 281–300, 2016.
- [61] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor, “Language-conditioned imitation learning for robot manipulation tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 139–13 150, 2020.
- [62] T. Kollar, S. Tellex, D. Roy, and N. Roy, “Toward understanding natural language directions,” in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2010, pp. 259–266.
- [63] 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.
- [64] 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.
- [65] 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,” *arXiv preprint arXiv:2306.08647*, 2023.
- [66] H. Sha, Y. Mu, Y. Jiang, L. Chen, C. Xu, P. Luo, S. E. Li, M. Tomizuka, W. Zhan, and M. Ding, “Languagempc: Large language models as decision makers for autonomous driving,” *arXiv preprint arXiv:2310.03026*, 2023.
- [67] 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,” *arXiv preprint arXiv:2307.04721*, 2023.
- [68] M. G. Arenas, T. Xiao, S. Singh, V. Jain, A. Z. Ren, Q. Vuong, J. Varley, A. Herzog, I. Leal, S. Kirmani *et al.*, “How to prompt your robot: A promptbook for manipulation skills with code as policies,” in *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.
- [69] L. Zha, Y. Cui, L.-H. Lin, M. Kwon, M. G. Arenas, A. Zeng, F. Xia, and D. Sadigh, “Distilling and retrieving generalizable knowledge for robot manipulation via language corrections,” *arXiv preprint arXiv:2311.10678*, 2023.
- [70] J. Liang, F. Xia, W. Yu, A. Zeng, M. G. Arenas, M. Attarian, M. Bauza, M. Bennice, A. Bewley, A. Dostmohamed *et al.*, “Learning to learn faster from human feedback with language model predictive control,” *arXiv preprint arXiv:2402.11450*, 2024.
- [71] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, “Voxposer: Composable 3d value maps for robotic manipulation with language models,” *arXiv preprint arXiv:2307.05973*, 2023.
- [72] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu *et al.*, “Pivot: Iterative visual prompting elicits actionable knowledge for vlm’s,” *arXiv preprint arXiv:2402.07872*, 2024.
- [73] 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,” *arXiv preprint arXiv:2403.01823*, 2024.
- [74] D. Shah, M. R. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine, “Navigation with large language models: Semantic guesswork as a heuristic for planning,” in *Conference on Robot Learning*. PMLR, 2023, pp. 2683–2699.
- [75] D. Shah, B. Osiński, S. Levine *et al.*, “Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action,” in *Conference on robot learning*. PMLR, 2023, pp. 492–504.
- [76] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada, “Saytap: Language to quadrupedal locomotion,” *arXiv preprint arXiv:2306.07580*, 2023.
- [77] 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.
- [78] A. Chen, A. Sharma, S. Levine, and C. Finn, “You only live once: Single-life reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 784–14 797, 2022.
- [79] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, “Rethinking the role of demonstrations: What makes in-context learning work?” *arXiv preprint arXiv:2202.12837*, 2022.
- [80] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui, “A survey on in-context learning,” *arXiv preprint arXiv:2301.00234*, 2022.
- [81] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, “Learning to walk in minutes using massively parallel deep reinforcement learning,” 2022. [Online]. Available: <https://arxiv.org/abs/2109.11978>
- [82] L. Han, Q. Zhu, J. Sheng, C. Zhang, T. Li, Y. Zhang, H. Zhang, Y. Liu, C. Zhou, R. Zhao *et al.*, “Lifelike agility and play on quadrupedal robots using reinforcement learning and generative pre-trained models,” *arXiv preprint arXiv:2308.15143*, 2023.

APPENDIX

We provide examples of typical interactions with the VLM in Figure 7 without history, without multi-step planning, compared to VLM-PC, and we find that with ours, the VLM is able to both reason through effectiveness of prior strategies and plan ahead to try coherent new strategies to overcome the current obstacle.

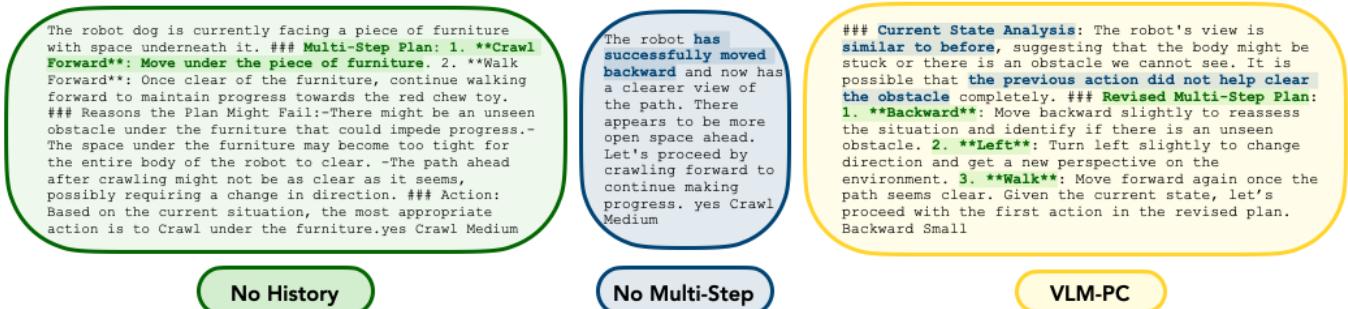


Fig. 7: **Typical VLM Interactions.** With VLM-PC (Right), the VLM can both analyze the efficacy of previous commands and prepare new, coherent plans to tackle the current obstacle, by combining benefits from multi-step planning from No History (Left) and reasoning over history from No Multi-Step (Center).

A. Additional Experimental Results in a Controlled Simulated Setting

In addition to the main results using VLM controllers, we also test our VLM controllers against other common non-VLM based baselines in a simulated environment requiring commonsense decisions on what low-level policy to use.

The test environment, a forked version of the legged_gym repo by [81], requires the robot to use all of its abilities: first moving forward, seeing no way out, then backing up and turning out of the dead end, and finally climbing stairs to get to the goal location. We record the success rate of the robot getting out of the dead end (henceforth “first success rate”) as well as the success rate of the robot in completing the obstacle course (henceforth “final success rate”). To evaluate the effectiveness of various prompting methods, we also compare our vanilla VLM-PC prompt to one which is substantially longer to include more detail about all possible skills (VLM-PC Detail).

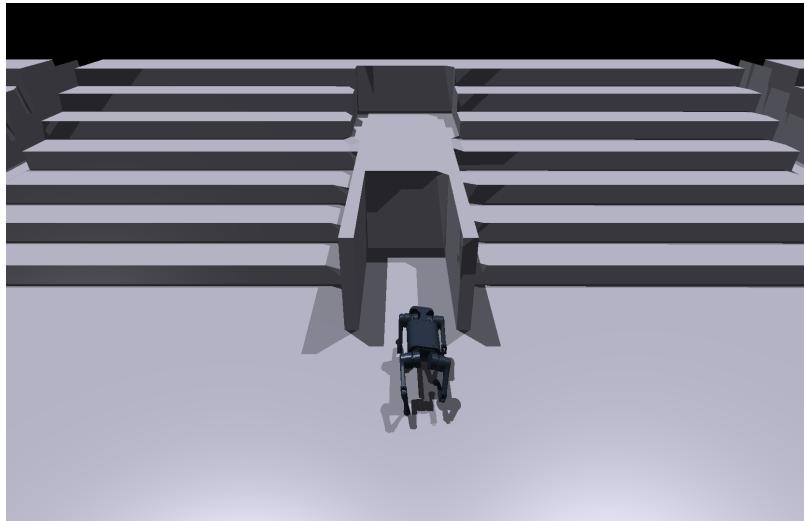


Fig. 8: **Simulated Evaluation Environment.**

We added comparisons to two prior methods that do not use VLMs: a learned high-level classifier and distillation [30], [31] from oracle actions in training environments. The training environments have the robot initialized in an environment with just one obstacle, rather than the full environment, with oracle actions to move through the environment. The training environments include a stairs environment to climb, a gap environment to jump over, a dead end to walk into to explore, go right or left to avoid, or back out of, and a wall to go right or left to get around or back up away from. The high-level classifier is henceforth referred to as HLC. Over 1200 episodes of data were used to train the HLC. At test time, the HLC was queried every 40 timesteps.

The distilled policy was trained with new episodes being continuously generated. At each timestep, a random choice to execute the actions from the expert policy or the student policy to step the simulation was made. Initially, the teacher policy was used to generate actions 99% of the time. Over the course of training, this was exponentially decayed to 2% (this factor was reduced by .1% each batch of 32 timestep). No matter which policy was used to generate the trajectory at each timestep, the expert policy’s actions in each state were generated to train the student policy.

For the high-level classifier, we used a CNN architecture with three convolutional layers with 3x3 kernel size, separated by max-pool followed by two convolutional layers, which is optimized via cross-entropy loss on the correct high-level action for the given image. For our distilled policy, we use the same CNN as an image encoder, and concatenate the image to an embedding (generated by a 2-layer MLP) of

other robot observations such as proprioceptive data. A final 2-layer MLP takes in the concatenated embeddings and outputs the appropriate actions. This policy is trained to minimize MSE loss with the exact actions taken by the low-level policy. Model hyperparameters are provided below:

TABLE III: Layer Sizes for HLC/Distillation

Image Embedding (Both)	
Image Input	128x128, 4 channel
CNN Kernel	3x3
CNN Stride	[2, 2, 1]
CNN Pooling	MaxPool
MLP Output Size	[512, 6]
Observation Embedding (Distill Only)	
Observation Input	235
MLP Output Size	[256, 128]
Final MLP (Distill Only)	
MLP Output Size	[256, 12]

We run 14 trials with each method up to 1500 simulated steps, which is more than enough for the robot to succeed or fail. We exclude trials where the policy fails due to problems with the simulation or low-level issues (e.g. if the robot trips itself), to purely evaluate how well the high-level policy can choose a best course of action. In Table IV, we find that similar to our conclusions in our real-world experiments, VLM-PC significantly outperforms both existing methods. In particular, we find that VLM-PC is able to reason through to solve the task much more effectively than HLC and distillation, as it is able to reason over previous attempts and efficiently recover from mistakes.

Method	Random	VLM No-Hist	VLM No-Plan	VLM-PC	VLM-PC Detail	HLC	Distill
First Success (%)	21.4	28.6	100	92.9	92.9	42.9	14
Final Success (%)	7.1	28.6	50	64.3	35.7	28.6	7.0

TABLE IV: Comparison of Simulated Methods.

B. Comparison to human oracle

Here, we ran the outdoor experiments using the controller to pilot the robot’s default policy to explore each environment for the end objective. We have added an oracle expert demonstration, where a human operator replaces the VLM but receives the same inputs and commands the same outputs, on our three outdoor environments. As expected, the robot is able to complete the task more quickly and successfully. In Table V, we find that a human controlling the robot with only egocentric data can easily solve the setting in the time allotted. However, unlike VLM-PC, this requires heavy human supervision throughout deployment time.

Method	Outdoor 1			Outdoor 2			Outdoor 3		
	Avg (s) ↓	Median (s) ↓	Success (%)	Avg (s) ↓	Median (s) ↓	Success (%)	Avg (s) ↓	Median (s) ↓	Success (%)
VLM-PC	49.4	17	60	68.8	65.5	60	61.7	50.5	60
Human Oracle	17.8	18.2	100	21.8	21.5	100	36.4	34.5	100

TABLE V: VLM-PC Comparison to Human Oracle.

C. Skill Details and Hyperparameters

We obtain different behaviors from the default controller by modulating the parameters passed in. Specifically, we control x- and y-velocity in the robot frame, gait type, body height, yaw speed, and duration to achieve different skills. The parameters used for each skill are described in the tables below, along with the duration corresponding to each magnitude. After the action is done executing, the robot will stay frozen in the position it was left in at the end of the last action, e.g. if the last action was to crawl, the robot will stay low to the ground. We additionally provide our GPT-4o query hyperparameters.

D. Prompts and Logs

In the following, we include the prompts used for VLM-PC, where the text highlighted in grey indicates text that is used for all comparison methods (including No History and No Multi-Step Plan). The text in green corresponds to the prompting for reasoning over history and is included in VLM-PC and the No Multi-Step Plan prompts. The text highlighted in blue corresponds to prompting for multi-step planning, which is included in VLM-PC and No History. The text in yellow corresponds to reasoning over the historical multi-step plan and is included only in the full VLM-PC prompt. When included, the ICL prompt went immediately after the first paragraph of the initial prompt and consisted of a short explanation followed by example egocentric views and one or two actions the robot might take when it each view. For VLM-PC and No Multi-Step Plan methods, the “Initial Prompt” below was given at the start and repeated after every six responses. Otherwise the “Successive Prompt” was given in all queries after the first. Note that we used GPT-4o as the VLM for all of our experiments, and additional prompt tuning may be necessary for other VLMs. Anecdotally, we tried using the Gemini Flash model and found that it did not reason as effectively with these prompts.

In the pages afterward, we display a full example log with VLM-PC on the Outdoor 2 obstacle course, where the prompts and input images are included in blue and the output of the VLM at each step is included in green. See our anonymous website for videos of our results.

Initial Prompt:

You are controlling a robot dog to get through obstacles in this scene. The robot dog is the size of a golden retriever. The goal is to make forward progress in this starting direction and arrive at the location of a red chew toy on the other side of the obstacles in front of you. After each action you command the robot to take, you will be given a new image taken from the head of the robot dog. Each time you receive an image, you must command one of the following six actions: Walk, Crawl, Left, Right, Backward, Climb. You should command Crawl when it should go under an obstacle, command Climb when it should go over the obstacle. Examples of obstacles the robot can safely Climb over include stairs, small suitcases, cushions, small logs, and small boxes. Do not try to Crawl under these. Examples of obstacles the robot can safely Crawl under include furniture (if there is sufficient space) and suspended tree branches. Do not try to Climb furniture and suspended tree branches. Command Left when it should go left to get around an obstacle, and Right when it should go right to get around the obstacle. Right/Left command inplace turns, meaning that the robot will change the direction it is facing but not its position. This means that if you command a turn, you need to move forwards in order to move in the new direction. Do not try to turn Left or Right when you confidently believe you are underneath or above an obstacle. You should command the robot Backward if the camera is obscured and it is not clear what other action to take. If there is space to walk forward in front of you, you should command Walk. If you ever command Backward, do not try to do the same action you did before you commanded Backward again immediately (you may do so later once you are in a different scenario). If it is possible to go under the obstacle, try to Crawl, and if it is possible to go over an obstacle, try to Climb, but only if it seems like a clear path after the obstacle. If not, it might be a dead end and it might be better to move Backward. Keep in mind that the camera is on the head of the robot, and the whole body of the robot is 3 feet behind and also needs to clear the obstacle. Even if the camera view is clear, the body of the robot might be stuck: in these cases, consider Backward or repeating the same action as before. The task can be completed with the right sequence of actions, so if the robot is not succeeding, you should use the history of actions and experiences to figure out why and command different actions. When you chose Walk, Crawl, or Climb, you will move directly forward, so you may want to slightly turn to make sure you will move in the right direction before doing so.

In your first response, describe a multi-step plan using the available actions specified above for completing the task and give reasons that plan might fail. At each subsequent step, the beginning of your response should use the state of the robot, including its current position and orientation, to reason through what progress the robot has made with the plan. You should reconsider your multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan seems to be working, keep following actions from that plan. If an action from the plan does not seem to be working, either revise the plan or try a making a new multi-step plan with different actions. For example, if you have had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle.

Each time you give a response, the third to last word of your response should be a yes or no indicating whether progress was made on the prior state with the previous command. The second to last word of your output should be the action to take. The last word of your response should be a magnitude

(Small, Medium, Large) that specifies how much of that action to take. For Right or Left commands, magnitude Small corresponds to turning that direction a full camera view amount. For other commands, magnitude Small corresponds to taking that action for around one second. Once again, the last three* words of your response must be in the format yes/no Walk/Crawl/Left/Right/Backward/Climb Small/Medium/Large.

Successive Prompt:

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You should reconsider your multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan** is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan** is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.

*For No History: "two"

**For No Multi-Step Plan: "your old sequence of actions"

You are controlling a robot dog to get through obstacles in this scene. The robot dog is the size of a golden retriever. The goal is to make forward progress in this starting direction and arrive at the location of a red chew toy on the other side of the obstacles in front of you. After each action you command the robot to take, you will be given a new image taken from the head of the robot dog. Each time you receive an image, you must command one of the following six actions: Walk, Crawl, Left, Right, Backward, Climb. You should command Crawl when it should go under an obstacle, command Climb when it should go over the obstacle.

Examples of obstacles the robot can safely Climb over include stairs, small suitcases, cushions, small logs, and small boxes. Do not try to Crawl under these. Examples of obstacles the robot can safely Crawl under include furniture (if there is sufficient space) and suspended tree branches. Do not try to Climb furniture and suspended tree branches. Command Left when it should go left to get around an obstacle, and Right when it should go right to get around the obstacle.

Right/Left command in place turns, meaning that the robot will change the direction it is facing but not its position. This means that if you command a turn, you need to move forwards in order to move in the new direction. Do not try to turn Left or Right when you confidently believe you are underneath or above an obstacle. You should command the robot Backward if the camera is obscured and it is not clear what other action to take. If there is space to walk forward in front of you, you should command Walk. If you ever command Backward, do not try to do the same action you did before you commanded Backward again immediately (you may do so later once you are in a different scenario). If it is possible to go under the obstacle, try to Crawl, and if it is possible to go over an obstacle, try to Climb, but only if it seems like a clear path after the obstacle. If

not, it might be a dead end and it might be better to move Backward. Keep in mind that the camera is on the head of the robot, and the whole body of the robot is 3 feet behind and also needs to clear the obstacle. Even if the camera view is clear, the body of the robot might be stuck: in these cases, consider Backward or repeating the same action as before. The task can be completed with the right sequence of actions, so if the robot is not succeeding, you should use the history of actions and experiences to figure out why and command different actions.

When you chose Walk, Crawl, or Climb, you will move directly forward, so you may want to slightly turn to make sure you will move in the right direction before doing so. In your first response, describe a multi-step plan using the available actions specified above for completing the task and give reasons that plan might fail. At each subsequent step, the beginning of your response should use the state of the robot, including its current position and orientation, to reason through what progress the robot has made with the plan. You should reconsider your multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan seems to be working, keep following actions from that plan. If an action from the plan does not seem to be working, either revise the plan or try a making a new multi-step plan with different actions. For example, if you have had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. Each time you give a response, the third to last word of your response should be a yes or no indicating whether progress was made on the prior state with the previous command. The second to last word of your output should be the action to take. The last word of your response should be a magnitude (Small, Medium, Large) that specifies how much of that action to take. For Right or Left commands, magnitude Small corresponds to turning that direction a full camera view amount. For other commands, magnitude Small corresponds to taking that action for around one second. Once again, the last three words of your response must be in the format yes/no Walk/Crawl/Left/Right/Backward/Climb Small/Medium/Large.



Multi-Step Plan:

1. **Crawl (Small):** The robot dog should crawl under the bamboo stems in front of it. This will allow it to make progress without getting stuck or needing to navigate around the bamboo.
2. **Walk (Medium):** After clearing the bamboo, the robot dog should walk forward to cover more ground towards the red chew toy.
3. **Left (Small):** If there are obstacles directly ahead, the robot dog should turn left to see if there is a clearer path.
4. **Walk (Medium):** Continue walking forward if the path is clear.
5. **Right (Small):** If there are still obstacles, the robot dog should turn right to check for another clear path.

Reasons the Plan Might Fail:

- The bamboo stems might be too low or too close together for the robot dog to crawl under.
- There may be additional obstacles not visible in the initial image that block the robot dog's path.
- The robot dog might get stuck on uneven ground or hidden objects while crawling or walking.

Current Action:

Let's begin with the first step of the plan.

yes Crawl Small

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You reconsider your new multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.



Analysis:

The robot has successfully crawled under the bamboo stems, as evidenced by the new view. The red chew toy is now visible in the distance, indicating forward progress. However, there is still an obstacle in the form of a low platform or bench ahead. The space underneath this platform appears large enough for the robot to crawl under.

Revised Multi-Step Plan:

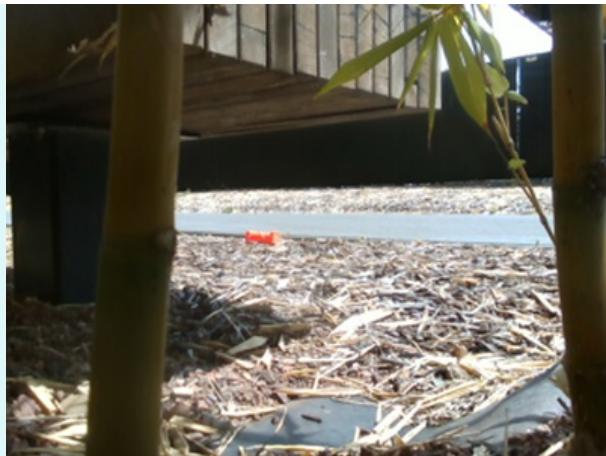
1. **Crawl (Small):** Continue crawling forward to move under the low platform or bench.
2. **Walk (Medium):** Once clear of the platform, the robot should walk forward to close the remaining distance to the chew toy.
3. **Left (Small):** If there are obstacles directly ahead, turn left to find a clear path.
4. **Walk (Medium):** Continue walking forward if the path is clear.
5. **Right (Small):** If there are still obstacles, turn right to check for another clear path.

Next Action:

Let's proceed with the first step of the revised plan.

yes Crawl Small

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You reconsider your new multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.



Analysis:

The view has not changed, indicating that the robot might be stuck. The body of the robot may not have cleared the bamboo stems completely or there could be another unseen obstacle behind or beneath the platform.

Revised Multi-Step Plan:

1. **Backward (Small):** Move backward to ensure the robot is completely clear of any obstacles it might be stuck on.
2. **Crawl (Medium):** After moving backward, attempt to crawl under the platform again to ensure the robot can clear the obstacle.

Next Action:

Let's proceed with the first step of the revised plan.

no Backward Small

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You reconsider your new multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.



Analysis:

The robot has successfully moved backward, revealing more of the bamboo stems and the low platform. The red chew toy is still visible in the distance. The robot is now in a better position to reattempt crawling under the platform.

Revised Multi-Step Plan:

1. **Left (Small)**: Turn left to better align the robot with a clearer path through the bamboo stems.
2. **Crawl (Medium)**: After turning, crawl forward to move under the low platform.
3. **Walk (Medium)**: Once clear of the platform, walk forward to close the remaining distance to the chew toy.
4. **Left (Small)**: If there are obstacles directly ahead, turn left to find a clear path.
5. **Walk (Medium)**: Continue walking forward if the path is clear.

Next Action:

Let's proceed with the first step of the revised plan.

yes Left Small

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You reconsider your new multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.



Analysis:

The robot has successfully turned left, revealing a clearer path ahead. There is a new obstacle, a cart, which the robot can likely climb over. The red chew toy is still visible in the distance, indicating the direction to progress.

Revised Multi-Step Plan:

1. **Climb (Medium)**: Climb over the cart to make progress towards the red chew toy.
2. **Walk (Medium)**: Once clear of the cart, walk forward to close the remaining distance to the chew toy.
3. **Left (Small)**: If there are obstacles directly ahead, turn left to find a clear path.
4. **Walk (Medium)**: Continue walking forward if the path is clear.
5. **Right (Small)**: If there are still obstacles, turn right to check for another clear path.

Next Action:

Let's proceed with the first step of the revised plan.

yes Climb Medium

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You reconsider your new multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.



Analysis:

The robot has moved forward and is now partially under the cart, indicating that the climb was not entirely successful. The body of the robot may be stuck under the cart.

Revised Multi-Step Plan:

1. **Backward (Small)**: Move backward to clear the robot from being stuck under the cart.
2. **Right (Small)**: Turn right to align with a path around the cart.
3. **Walk (Medium)**: Walk forward to bypass the cart and continue towards the red chew toy.
4. **Left (Small)**: If there are obstacles directly ahead, turn left to find a clear path.
5. **Walk (Medium)**: Continue walking forward if the path is clear.

Next Action:

Let's proceed with the first step of the revised plan.

no Backward Small

As a reminder, you are controlling a robot dog to get through obstacles in this scene. The robot dog is the size of a golden retriever. The goal is to make forward progress in this starting direction and arrive at the location of a red chew toy on the other side of the obstacles in front of you. After each action you command the robot to take, you will be given a new image taken from the head of the robot dog. Each time you receive an image, you must command one of the following six actions: Walk, Crawl, Left, Right, Backward, Climb. You should command Crawl when it should go under an obstacle, command Climb when it should go over the obstacle. Examples of obstacles the robot can safely Climb over include stairs, small suitcases, cushions, small logs, and small boxes. Do not try to Crawl under these. Examples of obstacles the robot can safely Crawl under include furniture (if there is sufficient space) and suspended tree branches. Do not try to Climb furniture and suspended tree branches. Command Left when it should go left to get around an obstacle, and Right when it should go right to get around the obstacle. Right/Left command in place turns, meaning that the robot will change the direction it is facing but not its position. This means that if you command a turn, you need to move forwards in order to move in the new direction. Do not try to turn Left or Right when you confidently believe you are underneath or above an obstacle. You should command the robot Backward if the camera is obscured and it is not clear what other action to take. If there is space to walk forward in front of you, you should command Walk. If you ever command Backward, do not try to do the same action you did before you commanded Backward again immediately (you may do so later once you are in a different scenario). If it is possible to go under the obstacle, try to Crawl, and if it is possible to go over an obstacle, try to Climb, but only if it seems like a clear path after the obstacle. If not, it might be a dead end and it might be better to move Backward.

Keep in mind that the camera is on the head of the robot, and the whole body of the robot is 3 feet behind and also needs to clear the obstacle. Even if the camera view is clear, the body of the robot might be stuck: in these cases, consider Backward or repeating the same action as before. The task can be completed with the right sequence of actions, so if the robot is not succeeding, you should use the history of actions and experiences to figure out why and command different actions. When you chose Walk, Crawl, or Climb, you will move directly forward, so you may want to slightly turn to make sure you will move in the right direction before doing so. In your first response, describe a multi-step plan using the available actions specified above for completing the task and give reasons that plan might fail. At each subsequent step, the beginning of your response should use the state of the robot, including its current position and orientation, to reason through what progress the robot has made with the plan. You should reconsider your multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan seems to be working, keep following actions from that plan. If an action from the plan does not seem to be working, either revise the plan or try a making a new multi-step plan with different actions. For example, if you have had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. Each time you give a response, the third to last word of your response should be a yes or no indicating whether progress was made on the prior state with the previous command. The second to last word of your output should be the action to take. The last word of your response should be a magnitude (Small, Medium, Large) that specifies how much of that action to take. For Right or Left commands, magnitude Small corresponds to turning that direction a full camera view amount. For other commands, magnitude Small corresponds to taking that action for around one second. Once again, the last three words of your response must be in the format yes/no Walk/Crawl/Left/Right/Backward/Climb Small/Medium/Large.



Analysis:

The robot has successfully moved backward and now has a clearer view of the cart and the path around it. The path to the right seems to provide more space to navigate around the cart.

Revised Multi-Step Plan:

1. **Right (Small):** Turn right to align with the clearer path around the cart.
2. **Walk (Medium):** Walk forward to bypass the cart and continue towards the red chew toy.
3. **Left (Small):** If there are obstacles directly ahead, turn left to find a clear path.
4. **Walk (Medium):** Continue walking forward if the path is clear.
5. **Right (Small):** If there are still obstacles, turn right to check for another clear path.

Next Action:

Let's proceed with the first step of the revised plan.

yes Right Small

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You reconsider your new multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.



Analysis:

The robot has successfully turned right, providing a clearer view under the platform. The red chew toy is still visible in the distance. The space under the platform appears to be large enough for the robot to crawl under it.

Revised Multi-Step Plan:

1. **Crawl (Medium)**: Crawl forward to move under the platform.
2. **Walk (Medium)**: Once clear of the platform, walk forward to close the remaining distance to the chew toy.
3. **Left (Small)**: If there are obstacles directly ahead, turn left to find a clear path.
4. **Walk (Medium)**: Continue walking forward if the path is clear.
5. **Right (Small)**: If there are still obstacles, turn right to check for another clear path.

Next Action:

Let's proceed with the first step of the revised plan.

yes Crawl Medium

Based on your last action, the robot has moved. Here is what the robot currently sees. If the view is exactly the same as before, this is likely because your last action was not appropriate for this situation and the body of the robot is probably

stuck (maybe because of an unobserved obstacle). Keep in mind that this is the view from the head of the robot. Even if the camera view is clear, the body of the robot might be stuck. In these cases, consider Backward. Remember your ultimate objective of making progress in the direction the robot initially started in so that you eventually arrive at the red chew toy. Use the state of the robot including its current position and orientation to reason through what progress the robot has made with the plan. You reconsider your new multi-step plan of actions to take, keeping in mind the history of what actions you have selected and the progress in the scene. If the plan is working well, keep following actions from that plan, making slight adjustments if necessary. If the plan is not working, try a new multi-step plan with different actions. For example, if you had many tries of Climbing over an obstacle and the robot is unable to make reliable forward progress, you might want to try going Backward or Crawling or going Left or Right around the obstacle. End your response with the first action in the chosen plan and a magnitude in the format specified earlier.



Analysis:

The robot has successfully crawled under the platform, and the view is now clear with the red chew toy very close ahead. There are no visible obstacles between the robot and the toy.

Revised Multi-Step Plan:

1. **Walk (Medium):** Walk forward to reach the red chew toy.

Next Action:

Let's proceed with the first step of the revised plan.

yes Walk Medium

TABLE VI: Behavior Parameters

Parameter	Walk	Climb	Crawl	Left Turn	Right Turn	Backward
X-Velocity (Small)	0.4 m/s	0.6 m/s	0.3 m/s	0 m/s	0 m/s	-0.3 m/s
X-Velocity (Medium)	0.6 m/s	0.6 m/s	0.3 m/s	0 m/s	0 m/s	-0.3 m/s
X-Velocity (Large)	0.6 m/s	0.6 m/s	0.3 m/s	0 m/s	0 m/s	-0.3 m/s
Y-Velocity	0 m/s	0 m/s	0 m/s	0 m/s	0 m/s	0 m/s
Gait Type	1	3	1	1	1	1
Body Height	0 m	0 m	-0.3 m	0 m	0 m	0 m
Yaw Speed	0 rad/s	0 rad/s	0 rad/s	0.3 rad/s	-0.3 rad/s	0 rad/s
Duration (Small)	3 s	6 s	2 s	2.5 s	2.5 s	1.5 s
Duration (Medium)	5 s	9 s	3 s	3.5 s	3.5 s	2.5 s
Duration (Large)	7 s	12 s	4 s	4.5 s	4.5 s	5 s

TABLE VII: GPT-4o Query Hyperparameters

Parameter	Value
Temperature	0.7
Top P	0.95
Max Tokens	800