

Commonsense Reasoning for Legged Robot Adaptation with Vision-Language Models

Anonymous Author(s)

Affiliation

Address

email

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

22 **Keywords:** vision-language models, on-the-fly adaptation, legged locomotion

23 1 Introduction

24 Robots deployed in open-world environments must be able to handle highly unstructured and
25 complicated environments. This is particularly the case for legged robots, which may need to
26 operate in an extremely diverse range of circumstances. Consider a quadruped robot tasked with
27 performing search and rescue in a collapsed building. This robot faces a long tail of different possible
28 environments and obstacles, which might require climbing over debris, crawling through gaps, and
29 backtracking and navigating out of dead ends without a map. Handling these diverse real-world
30 scenarios autonomously, without detailed human guidance and specific skill directives, remains a
31 significant challenge. Prior work in locomotion has endowed legged robots with agile skills like
32 running, climbing, and crawling [1, 2], but possessing these skills alone does not solve the problem
33 of fully autonomous deployment. To handle complex, unstructured scenarios, a robot must be able to
34 decide how to deploy its repertoire of skills with a nuanced understanding of its situation. Consider
35 the example of clearing a novel obstacle like debris in a collapsed building. We expect an intelligent
36 robot to perceive and try a skill that is likely to succeed, e.g., climbing. If the robot’s attempt was
37 unsuccessful, e.g. the debris is too slippery to climb over, the robot should recognize this and try
38 another strategy given the information it has gathered, e.g. backtrack and try a new strategy like
39 finding a path around the log instead.

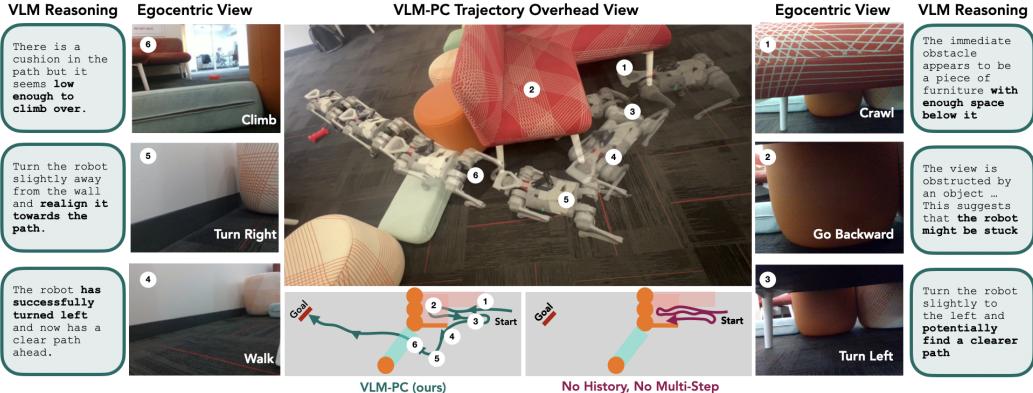


Figure 1: Vision-Language Model Predictive Control (VLM-PC) enables real-world locomotion adaptation. By leveraging the commonsense reasoning abilities of pre-trained VLMs to adaptively select behaviors, VLM-PC allows legged robots to quickly adjust strategies when encountering a wide range of situations, 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.

- 40 Foundation models such as vision-language models (VLMs) have the potential to help robots handle
 41 real-world, unstructured scenarios, since they possess commonsense knowledge acquired from diverse
 42 internet-scale image and language data. Indeed, multiple prior works have shown how robots can
 43 leverage knowledge in large language models (LLMs) and VLMs for high-level planning [3, 4, 5]
 44 in robotic manipulation. In principle, VLMs can provide high-level semantic knowledge for legged
 45 robots as well, e.g. by identifying obstacles or selecting high-level behaviors. However, the use of
 46 LLMs and VLMs has been far more limited in legged robots, compared to robotic manipulation.
 47 Furthermore, in diverse, complex environments with obstacles where the robot may get stuck and
 48 need to try multiple strategies to overcome, naively prompting a VLM to output a skill may often fail
 49 due to inaccuracies in the model’s interpretation of the scene and subsequent inability to adapt with
 50 the robot’s environment interactions.
- 51 In this work, we investigate how legged robots can leverage VLMs and their general knowledge about
 52 the structure of the world and commonsense reasoning capabilities to suggest contextually informed
 53 behaviors based on visual inputs. We find the following two key insights crucial for facilitating
 54 adaptive behavior selection in complex, unstructured settings with VLMs: (1) The robustness of
 55 VLMs in novel situations can be greatly improved by taking into account the robot’s interaction
 56 history, leveraging chain-of-thought reasoning [6, 7], and (2) Prompting the model to plan multiple
 57 skills ahead and optionally replan at each timestep is essential for foreseeing potential failures.
- 58 Combining these insights, we propose VLM Predictive Control (VLM-PC), which can be seen as
 59 a history-conditioned high-level analogue of visual model predictive control [8, 9, 10, 11] in skill
 60 space. With an image of the robot’s view along with the history of interactions as input, the VLM is
 61 prompted to generate a multi-step plan of skills. In order to choose what plan to follow, and ultimately
 62 what skill to execute next, the model is prompted to reason through the robot’s current state and
 63 whether the previous existing plan made progress on the desired task and re-plan if needed.
- 64 In our experiments, we find that leveraging VLMs in this way allows a Go1 robot to handle a range of
 65 real-world situations that have not been tackled by prior work in a fully autonomous manner. Across
 66 five challenging real-world settings, one of which is shown in Figure 1, our approach completes
 67 the target task around 30% more successfully by leveraging in-context adaptation and multi-step
 68 planning. Our results illustrate the potential of pre-trained VLMs, even without training on interaction
 69 data, to navigate situations autonomously from the view of a physical embodiment.

70 **2 Related Work**

71 Our work tackles the issue of enabling legged robots to perform robustly in unstructured, unknown
72 test-time conditions. Traditional model-based control approaches have achieved impressive agile
73 locomotion [12, 1, 13, 14, 15, 16, 17] but are not well-equipped to navigate arbitrary, open-world
74 environments. Learning-based approaches hold the promise of greater generalization capabilities,
75 and training a single policy with reinforcement learning (RL) has also demonstrated successful
76 low-level locomotion capabilities from robust walking to jumping and bipedal walking [18, 19, 20,
77 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]. Behind a majority of these successes is the use of domain
78 randomization [32, 33, 34, 35, 36, 19, 21, 37, 38, 39, 40], which involves training the robot under a
79 variety of different dynamics to robustify the policy. Our work tackles an orthogonal, complementary
80 problem: enabling legged robots to autonomously solve complex, partially observed tasks given a
81 repertoire of low-level skills (which can be acquired through either traditional model-based approaches
82 or RL training). Using these skills to solve a long-horizon task requires understanding the scene and
83 reasoning over the information gathered in the environment, trying different low-level strategies, and
84 adapting high-level plans on-the-fly accordingly.

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

94 Outside legged locomotion, extensive research has explored combining prior behaviors to address
95 long-horizon tasks, often by training high-level policies that orchestrate learned skills into complex
96 behaviors [48, 49, 50, 51, 52, 53, 54, 55, 56, 57]. Natural language provides a simple abstraction to
97 index these behaviors, and using language as an abstraction for behaviors provides an interpretable
98 space for a high-level planner to select strategies to try [3, 4, 58, 59, 5, 60, 61, 62] or to generate
99 robot code [63, 64]. In-context reasoning with LLMs has refined low-level behaviors [65, 66, 67, 68],
100 improved planning with feedback [58] and facilitated learning from human feedback [69, 70], but
101 these do not incorporate VLMs, which can offer rich multimodal understanding. Recent works have
102 begun going beyond LLMs and incorporating VLMs for manipulation [71, 72, 73] and navigation [74,
103 75]. Unlike these works, we focus particularly on equipping the robot to handle unpredictable
104 situations where it might get stuck and need to explore different strategies to make progress. Enabling
105 this in a diverse array of environments requires robust commonsense reasoning abilities, and we study
106 the extent to which VLMs can provide these for legged robots.

107 While high-level planning in language grounding has been studied for manipulation or navigation
108 tasks, it has explored far less for legged locomotion. Key works have interfaced through foot contact
109 patterns [76] or code [77] with LLM planning. Our work implements a straightforward language-skill
110 interface for locomotion and is the first to explore how legged robots can utilize the commonsense
111 reasoning capabilities of pre-trained VLMs to autonomously guide adaptive behavior selection. In
112 particular, the vast majority of prior works apply LLMs zero-shot based on the current instruction or
113 observation; in this work, we study how the in-context adaptation ability of VLMs can help robots
114 adapt to different scenarios.

115 **3 Problem Statement**

116 We assume the robot has access to a set of n skills, which are sufficient to allow the robot to traverse
117 the environment. Given the recent development of highly robust low-level quadrupedal locomotion
118 controllers via RL [22, 41, 31], we believe this assumption to be reasonable for a wide variety of
119 real-world scenarios. For example, if the robot has the ability to climb, crawl, walk forwards and
120 backwards, and turn in various directions, we expect that it could sequentially apply these skills

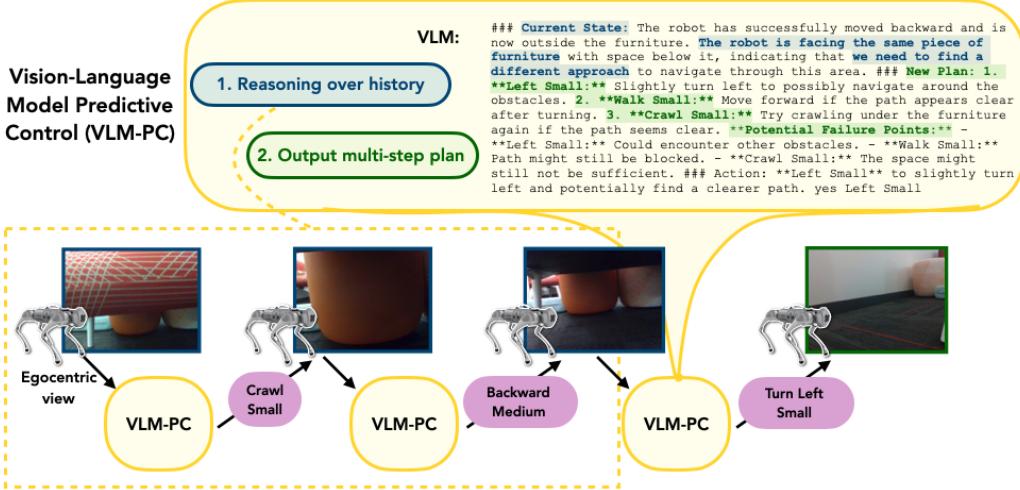


Figure 2: **Vision-Language Model Predictive Control (VLM-PC)**. Our method uses a pre-trained VLM to provide high-level skill commands for a legged robot to execute. Given the robot’s current view and history of interactions, the VLM is first prompted to reason through the robot’s current state and 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.

121 to handle a variety of situations – the challenge then is to determine when to deploy each skill to
 122 navigate an unseen, unstructured situation. Each skill corresponds to a policy π_i , which takes in a
 123 state $s \in \mathcal{S}$ and outputs a low-level action u . At test time, the robot interacts in a partially observed
 124 environment, where it receives images $\{I\}$ and must process them and output a skill and time duration
 125 δ that executes policy π_i for an amount of time δ .

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

132 4 Vision-Language Model Predictive Control (VLM-PC)

133 Our goal is to enable legged robots to make informed decisions that lead to effective, context-aware
 134 adaptation to help navigate these situations autonomously and successfully. Our central hypothesis is
 135 that many real-world situations demand complex reasoning due to unexpected circumstances that
 136 may be difficult to generalize to. In this section, we first describe how we represent the robot’s skills
 137 via language to be used then by a pre-trained VLM. We then detail how we prompt the VLM to
 138 reason through the robot’s current state and history of interactions to select the next skill to execute to
 139 solve a task in unstructured environments.

140 4.1 Interfacing Robotic Locomotion Skills with VLMs

141 We consider generative VLMs, also known as multimodal language models, which take as input
 142 $\{I, x\}$, including images $\{I\}$ and prompt text x and outputs text y from a distribution over textual
 143 completions $P(\cdot | \{I\}, x)$. We label each of the robot’s prior behaviors $\pi_i \in \Pi$ with a command l_i , a
 144 textual description of the corresponding behavior. We also define levels of magnitude m for each
 145 that define the duration $\delta_{l_i, m}$ that π_i should be executed. While there are many ways to acquire
 146 locomotion policies, e.g., via traditional model-based techniques or learning-based approaches, we
 147 use the built-in controller provided by the Go1 robot. To make these policies amenable to being used
 148 by a VLM, we choose to represent the policies as skills (as opposed to less interpretable, low-level
 149 actions such as joint angles or foot contact patterns) that we acquire by varying several parameters

150 (x- and y-velocity in the robot frame, gait type, body height, yaw speed, and duration), the details
151 of which can be found in Appendix A.1. For example, l_i could be “Climb” or “Crawl”, and m is
152 “Small”, “Medium”, or “Large”. At timestep T , the VLM is prompted to output high-level action
153 $a_T = (l_i, m)$, which leads to the robot executing low-level actions $u_t = \pi_i(s_t; m_T)$ for the number
154 of seconds dictated by m_T . Through prompt engineering, we ensure that the VLM outputs the skill
155 and magnitude in a specified format that allows us to extract the high-level skill command for the
156 robot to execute.

157 4.2 Using VLMs for Adaptive Behavior Selection

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

166 **Using in-context reasoning to adapt on-the-fly.** In large foundation models, techniques like
167 chain-of-thought [6, 7, 79], where the model is prompted to output intermediate reasoning steps, have
168 been shown to significantly improve the model’s ability to perform complex reasoning. We aim to
169 leverage such techniques to equip the VLM to better understand and reason through the environment
170 and provide more effective high-level commands to the robot. In particular, we want the VLM to
171 reason through the history in the environment and the progress made with the commanded skills
172 before deciding on the next skill, in order to determine if the robot should try a new strategy. As
173 such, we include as input to the VLM an image representing the robot’s current view along with
174 the full history of interactions (including the robot’s previous images and the previous outputs of
175 the VLM) and a prompt, i.e. the input at timestep t is $(I_1, x_1, y_1, I_{\delta_m}, x_{\delta_m}, y_{\delta_m}, \dots, x_{t-1}, I_t, p_t)$,
176 which contains for each previous query step i , each previous image I_i and prompt x_i along with
177 VLM output y_i . We then prompt the VLM to first reason through what progress the robot has made
178 using the history of commands selected and the current position and orientation of the robot.

179 **Multi-step planning and execution.** Due to partial observability, there is often no clear answer as
180 to which skill is most appropriate for a given situation and without physical experience, the VLM is
181 not grounded in the robot’s low-level capabilities (i.e. the VLM understands that the robot can crawl,
182 but it does not know exactly how it crawls and whether this crawling skill will actually be successful
183 in the current situation). So, we use an approach akin to model predictive control [8, 9, 10, 11],
184 wherein we prompt the VLM to produce the immediate skill to execute by planning multiple steps l_t ,
185 $l_{t+\delta_m}, \dots, l_{t+k}$, into the future and reasoning about the consequences of the actions. This allows the
186 VLM to foresee different possible strategies that might be applicable to the current situation, so it
187 may better adjust in the future if the next chosen skill does not make progress. Then after the robot
188 executes the skill corresponding to l_t , the VLM repeats this planning for each step during deployment.
189 To implement this, we specifically prompt the VLM to make a multi-step plan taking into account
190 the latest visual observation I_t , compare the new plan to the prior existing plan, and use the one that
191 seems more applicable.

192 5 Experimental Results

193 In this section, we study whether VLM-PC can enable a Go1 quadruped robot to tackle five challenging
194 real-world situations in a fully autonomous manner. Concretely, we aim to answer the following
195 empirical questions: (1) Can VLM-PC enable the robot to autonomously adapt in unseen, partially
196 observed environments and effectively complete tasks that require reasoning over what strategies
197 the robot has tried in the past? (2) How much do in-context reasoning over the robot’s experience
198 and multi-step planning affect the robot’s ability to complete these test settings? (3) Does including

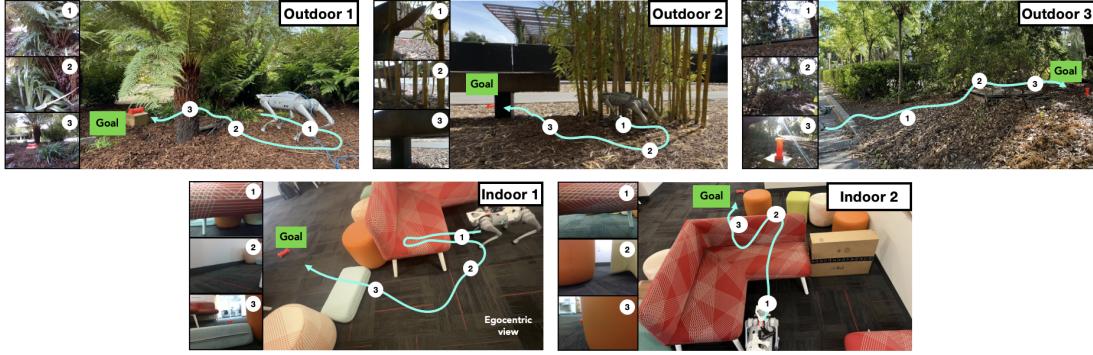


Figure 3: Deployment Environments. We evaluate VLM-PC on five challenging real-world settings, each of which presents unseen obstacles designed for the robot to get stuck, and requires commonsense reasoning to solve. For each setting, we give a third-person view of the obstacle course as well as an example 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.

199 additional in-context examples improve the robot’s ability to handle the given setting? We first
 200 describe our experimental setup before presenting the results of our experiments. For qualitative
 201 video results, see our anonymous project webpage: <https://vlm-adapt.vercel.app/>

202 5.1 Experimental Setup

203 We use a Go1 quadruped robot from Unitree. The robot is equipped with an Intel Realsense D435
 204 camera mounted on its head, which provides an egocentric view of the environment, which is the only
 205 source of information the robot has about its surroundings. We configure the default controller to
 206 correspond to a set of prior behaviors: walking forward, crawling (at a low height), climbing (which
 207 can overcome stair-height obstacles), walking backward, turning left, and turning right. This same
 208 set of behaviors is used for all experiments, and details of the skills are in Appendix A.1. In each
 209 setting, we report the average and median wall clock time in seconds (where lower is better) needed
 210 to complete the task along with the success rate across five trials for each method. If the robot does
 211 not complete the task within 100 seconds of executing actions, we consider it a failure. For each
 212 method in each setting, we report these metrics across five trials.

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

219 **Indoor 1:** The robot first must crawl under a couch, determine that it is a dead end, back up and turn
 220 to walk around the couch, climb a cushion it cannot pass without climbing, and finally locate the toy.
 221 **Indoor 2:** The robot first faces a couch that it must crawl under to the opposite side, then faces several
 222 stools blocking its path with a narrow gap between them, and must determine that it cannot fit through
 223 the gap and must turn and go around to locate the red chew toy.

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

226 **Outdoor 2:** The robot first faces a series of bamboo plants that it must turn from and go around, then
 227 a bench that it must crawl under, and then find the red chew toy.

228 **Outdoor 3:** The robot first faces a curb that it must climb over, a dirt hill that it must walk up, a
 229 wooden plank that it must climb over, and finally locate the red chew toy between the bushes.

230 **Comparisons.** We compare VLM-PC to several variants that differ in the amount of context
 231 provided to the VLM and the amount the VLM is prompted to plan: (1) **No History:** The VLM
 232 is prompted with only the current image and the prompt, and is not provided with any history of

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 1: **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.

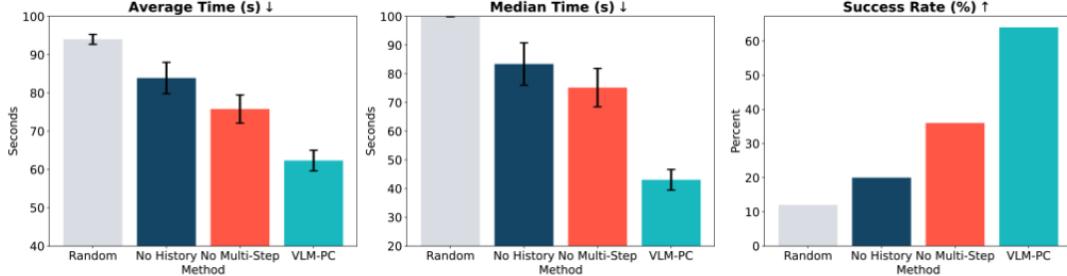


Figure 4: **Main Results Averaged Across Settings.** Across all five settings on average, VLM-PC significantly outperforms Random, No History, and No Multi-Step on average and median time to complete the task and success rate, performing roughly 30% more successfully than the next best method.

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.

5.2 Main Results

As shown in Figure 4, 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. We provide examples of typical interactions with the VLM in Figure 5 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.

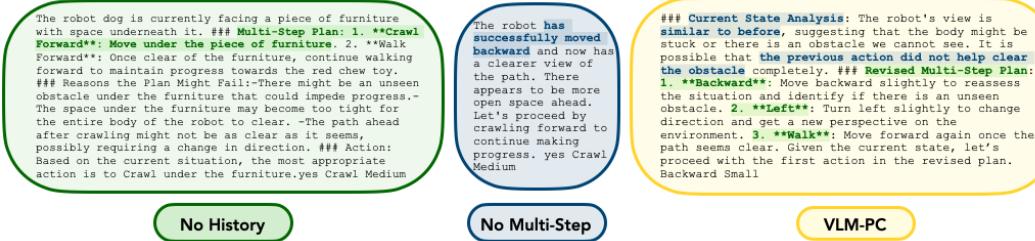


Figure 5: **Typical VLM Interactions.** With VLM-PC (Right), the VLM can both analyze the efficacy of previous commands and prepare new, coherent plans to tackle the current obstacle, by combining benefits from multi-step planning from No History (Left) and reasoning over history from No Multi-Step (Center).

256 5.3 Adding Labeled In-Context Examples Can Improve Performance

257 As large foundation models trained on
 258 Internet-scale data are used as these
 259 high level planners, they can leverage
 260 in-context learning, where examples
 261 or instructions are included as context
 262 in the input to the model [80, 81]. We
 263 provide an extension of our method
 264 including in-context examples, called
 265 VLM-PC+IC, where we include in the
 266 first prompt several additional images,
 267 taken from the egocentric view at different points in the environment, as well as a label for each
 268 of them with the best command to take. This provides the VLM with more context about the
 269 environment and the best strategies to take at key points. As shown in Table 6, we find that in two
 270 of the obstacle courses, this can significantly improve performance. While inexpensive to obtain,
 271 this does require human labeling of several images from the deployment environment with the best
 272 command to take, which may not be feasible in all deployment settings. Nonetheless, this extension
 273 further reinforces the importance of providing useful context to the VLM and having it use this
 274 context to make informed decisions, and shows that these labeled examples can be useful context on
 275 top of the history of experiences in the environment.

276 6 Discussion and Limitations

277 We introduced Vision-Language Model Predictive Control (VLM-PC), which enables legged robots
 278 to rapidly adapt to changing, unseen circumstances during deployment. On a Go1 quadruped robot,
 279 we find that VLM-PC can autonomously handle a range of complex real-world tasks involving
 280 climbing over obstacles, crawling under furniture, and navigating around dead ends and through
 281 cluttered environments. While VLM-PC is promising solution for enabling legged robots to handle
 282 new tasks, there remains much left to explore regarding how to best leverage VLMs for adaptive
 283 behavior for legged robots, especially as core VLM capabilities continue to improve. First, improving
 284 language grounding for locomotion to better capture the nuances of the robot’s capabilities could
 285 lead to more effective decision-making. It would also be interesting to explore if fine-tuning these
 286 VLMs, perhaps with techniques like reinforcement learning from human feedback, can lead to more
 287 efficient reasoning. In addition, the camera is currently mounted on the head of the robot and provides
 288 a limited field of view, making it challenging for the agent to understand the full context of its
 289 environment. This limitation sometimes causes the VLM to struggle with scenarios where the robot’s
 290 body may be stuck on an obstacle even if the head has cleared it. Incorporating more sensors or scene
 291 reconstruction could provide a more comprehensive view of the environment, allowing the VLM to
 292 reason more effectively. Finally, it would be interesting to explore how to use VLMs to combine
 293 high-level planning for locomotion with that for manipulation, to enable robots to handle a wider
 294 range of tasks.

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

Figure 6: **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.

276 As large foundation models trained on
 277 Internet-scale data are used as these
 278 high level planners, they can leverage
 279 in-context learning, where examples
 280 or instructions are included as context
 281 in the input to the model [80, 81]. We
 282 provide an extension of our method
 283 including in-context examples, called
 284 VLM-PC+IC, where we include in the
 285 first prompt several additional images,
 286 taken from the egocentric view at different points in the environment, as well as a label for each
 287 of them with the best command to take. This provides the VLM with more context about the
 288 environment and the best strategies to take at key points. As shown in Table 6, we find that in two
 289 of the obstacle courses, this can significantly improve performance. While inexpensive to obtain,
 290 this does require human labeling of several images from the deployment environment with the best
 291 command to take, which may not be feasible in all deployment settings. Nonetheless, this extension
 292 further reinforces the importance of providing useful context to the VLM and having it use this
 293 context to make informed decisions, and shows that these labeled examples can be useful context on
 294 top of the history of experiences in the environment.

295 **References**

- 296 [1] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion,
297 and R. Tedrake. Optimization-based locomotion planning, estimation, and control design for
298 the atlas humanoid robot. *Autonomous robots*, 40:429–455, 2016.
- 299 [2] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning
300 agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.
- 301 [3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan,
302 K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances.
303 *arXiv preprint arXiv:2204.01691*, 2022.
- 304 [4] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners:
305 Extracting actionable knowledge for embodied agents. In *International Conference on Machine
306 Learning*, pages 9118–9147. PMLR, 2022.
- 307 [5] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson,
308 Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint
309 arXiv:2303.03378*, 2023.
- 310 [6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-
311 thought prompting elicits reasoning in large language models. *Advances in neural information
312 processing systems*, 35:24824–24837, 2022.
- 313 [7] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot
314 reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- 315 [8] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice—a
316 survey. *Automatica*, 25(3):335–348, 1989.
- 317 [9] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers &
318 chemical engineering*, 23(4-5):667–682, 1999.
- 319 [10] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *2017 IEEE
320 International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.
- 321 [11] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based
322 deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*,
323 2018.
- 324 [12] H. Dai, A. Valenzuela, and R. Tedrake. Whole-body motion planning with centroidal dynamics
325 and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages
326 295–302. IEEE, 2014.
- 327 [13] M. Hutter, C. Gehring, D. Jud, A. Lauber, D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie,
328 P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Höpflinger. Anymal
329 - a highly mobile and dynamic quadrupedal robot. pages 38–44, 2016.
- 330 [14] H.-W. Park, P. M. Wensing, and S. Kim. High-speed bounding with the mit cheetah 2: Control
331 design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017.
332 doi:10.1177/0278364917694244. URL <https://doi.org/10.1177/0278364917694244>.
- 333 [15] D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter. Dynamic locomotion through online
334 nonlinear motion optimization for quadrupedal robots. *IEEE Robotics and Automation Letters*,
335 3:2261–2268, 2018.
- 336 [16] G. Bledt, M. J. Powell, B. Katz, J. Carlo, P. Wensing, and S. Kim. Mit cheetah 3: Design and
337 control of a robust, dynamic quadruped robot. pages 2245–2252, 2018.

- 338 [17] B. Katz, J. Carlo, and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic
339 quadruped control. *2019 International Conference on Robotics and Automation (ICRA)*, pages
340 6295–6301, 2019.
- 341 [18] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep
342 reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.
- 343 [19] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke.
344 Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*,
345 2018.
- 346 [20] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforce-
347 ment learning for legged robots. *Conference on Robot Learning*, abs/1907.03613, 2019.
- 348 [21] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. In *2019*
349 *ieee/rsj international conference on intelligent robots and systems (iros)*, pages 3503–3510.
350 IEEE, 2019.
- 351 [22] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion
352 over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- 353 [23] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang. Learning vision-guided quadrupedal
354 locomotion end-to-end with cross-modal transformers. *arXiv preprint arXiv:2107.03996*, 2021.
- 355 [24] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains
356 using egocentric vision. In *Conference on Robot Learning*, 2022.
- 357 [25] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic
358 locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.
- 359 [26] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced skills by learning locomotion and
360 local navigation end-to-end. In *2022 IEEE/RSJ International Conference on Intelligent Robots
361 and Systems (IROS)*, pages 2497–2503. IEEE, 2022.
- 362 [27] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on
363 learning: Fine-tuning locomotion policies in the real world. In *2022 International Conference
364 on Robotics and Automation (ICRA)*, pages 1593–1599. IEEE, 2022.
- 365 [28] K. Caluwaerts, A. Iscen, J. C. Kew, W. Yu, T. Zhang, D. Freeman, K.-H. Lee, L. Lee, S. Saliceti,
366 V. Zhuang, et al. Barkour: Benchmarking animal-level agility with quadruped robots. *arXiv
367 preprint arXiv:2305.14654*, 2023.
- 368 [29] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi. Agile but safe: Learning collision-free
369 high-speed legged locomotion. *arXiv preprint arXiv:2401.17583*, 2024.
- 370 [30] Z. Zhuang, Z. Fu, J. Wang, C. G. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot
371 parkour learning. In *7th Annual Conference on Robot Learning*, 2023.
- 372 [31] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. *arXiv
373 preprint arXiv:2309.14341*, 2023.
- 374 [32] M. Cutler, T. J. Walsh, and J. P. How. Reinforcement learning with multi-fidelity simulators. In
375 *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3888–3895.
376 IEEE, 2014.
- 377 [33] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine. Epopt: Learning robust neural network
378 policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.
- 379 [34] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv
380 preprint arXiv:1611.04201*, 2016.

- 381 [35] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization
382 for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ*
383 *international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.
- 384 [36] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic
385 control with dynamics randomization. In *2018 IEEE international conference on robotics and*
386 *automation (ICRA)*, pages 3803–3810. IEEE, 2018.
- 387 [37] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino,
388 M. Plappert, G. Powell, R. Ribas, et al. Solving rubik’s cube with a robot hand. *arXiv preprint*
389 *arXiv:1910.07113*, 2019.
- 390 [38] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg. Dynamics randomization revisited: A
391 case study for quadrupedal locomotion. In *2021 IEEE International Conference on Robotics*
392 *and Automation (ICRA)*, pages 4955–4961. IEEE, 2021.
- 393 [39] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforce-
394 ment learning. *arXiv preprint arXiv:2205.02824*, 2022.
- 395 [40] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik,
396 S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, et al. Learning agile soccer skills for a bipedal
397 robot with deep reinforcement learning. *arXiv preprint arXiv:2304.13653*, 2023.
- 398 [41] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with
399 multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.
- 400 [42] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha. Learning fast adaptation with meta strategy
401 optimization. *IEEE Robotics and Automation Letters*, 2020.
- 402 [43] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots.
403 *arXiv preprint arXiv:2107.04034*, 2021.
- 404 [44] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: learning a unified policy for
405 manipulation and locomotion. In *Conference on Robot Learning*, 2022.
- 406 [45] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust
407 perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 2022.
- 408 [46] R. Yang, G. Yang, and X. Wang. Neural volumetric memory for visual locomotion control. In
409 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- 410 [47] A. S. Chen, G. Chada, L. Smith, A. Sharma, Z. Fu, S. Levine, and C. Finn. Adapt on-the-go:
411 Behavior modulation for single-life robot deployment. *arXiv preprint arXiv:2311.01059*, 2023.
- 412 [48] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI*
413 *conference on artificial intelligence*, volume 31, 2017.
- 414 [49] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. Mcp: Learning composable
415 hierarchical control with multiplicative compositional policies. *Advances in Neural Information*
416 *Processing Systems*, 32, 2019.
- 417 [50] Y. Lee, J. Yang, and J. J. Lim. Learning to coordinate manipulation skills via skill behavior
418 diversification. In *International conference on learning representations*, 2019.
- 419 [51] M. Sharma, J. Liang, J. Zhao, A. LaGrassa, and O. Kroemer. Learning to compose hierarchical
420 object-centric controllers for robotic manipulation. *arXiv preprint arXiv:2011.04627*, 2020.
- 421 [52] R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, and C. Schmid. Learning to combine
422 primitive skills: A step towards versatile robotic manipulation. In *2020 IEEE International*
423 *Conference on Robotics and Automation (ICRA)*, pages 4637–4643. IEEE, 2020.

- 424 [53] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning.
425 *Advances in neural information processing systems*, 31, 2018.
- 426 [54] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta. Efficient bimanual manipulation using learned
427 task schemas. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*,
428 pages 1149–1155. IEEE, 2020.
- 429 [55] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim. Guided reinforcement learning with learned skills.
430 *arXiv preprint arXiv:2107.10253*, 2021.
- 431 [56] M. Dalal, D. Pathak, and R. R. Salakhutdinov. Accelerating robotic reinforcement learning
432 via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34:
433 21847–21859, 2021.
- 434 [57] S. Nasiriany, H. Liu, and Y. Zhu. Augmenting reinforcement learning with behavior primitives
435 for diverse manipulation tasks. In *2022 International Conference on Robotics and Automation
436 (ICRA)*, pages 7477–7484. IEEE, 2022.
- 437 [58] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch,
438 Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language
439 models. *arXiv preprint arXiv:2207.05608*, 2022.
- 440 [59] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge,
441 and action in route instructions. *Def*, 2(6):4, 2006.
- 442 [60] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of
443 natural language to manipulation instructions. *The International Journal of Robotics Research*,
444 35(1-3):281–300, 2016.
- 445 [61] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. Language-
446 conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information
447 Processing Systems*, 33:13139–13150, 2020.
- 448 [62] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions.
449 In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages
450 259–266. IEEE, 2010.
- 451 [63] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code
452 as policies: Language model programs for embodied control. In *2023 IEEE International
453 Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.
- 454 [64] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and
455 A. Garg. Progprompt: Generating situated robot task plans using large language models. In
456 *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530.
457 IEEE, 2023.
- 458 [65] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez,
459 L. Hasenclever, J. Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint
460 arXiv:2306.08647*, 2023.
- 461 [66] H. Sha, Y. Mu, Y. Jiang, L. Chen, C. Xu, P. Luo, S. E. Li, M. Tomizuka, W. Zhan, and M. Ding.
462 Languagempc: Large language models as decision makers for autonomous driving. *arXiv
463 preprint arXiv:2310.03026*, 2023.
- 464 [67] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and
465 A. Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*,
466 2023.

- 467 [68] M. G. Arenas, T. Xiao, S. Singh, V. Jain, A. Z. Ren, Q. Vuong, J. Varley, A. Herzog, I. Leal,
 468 S. Kirmani, et al. How to prompt your robot: A promptbook for manipulation skills with code
 469 as policies. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@*
 470 *CoRL2023*, 2023.
- 471 [69] L. Zha, Y. Cui, L.-H. Lin, M. Kwon, M. G. Arenas, A. Zeng, F. Xia, and D. Sadigh. Distilling
 472 and retrieving generalizable knowledge for robot manipulation via language corrections. *arXiv*
 473 preprint *arXiv:2311.10678*, 2023.
- 474 [70] J. Liang, F. Xia, W. Yu, A. Zeng, M. G. Arenas, M. Attarian, M. Bauza, M. Bennice, A. Bewley,
 475 A. Dostmohamed, et al. Learning to learn faster from human feedback with language model
 476 predictive control. *arXiv preprint arXiv:2402.11450*, 2024.
- 477 [71] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value
 478 maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.
- 479 [72] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu,
 480 et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint*
 481 *arXiv:2402.07872*, 2024.
- 482 [73] S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi,
 483 and D. Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*,
 484 2024.
- 485 [74] D. Shah, M. R. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine. Navigation with large language
 486 models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*,
 487 pages 2683–2699. PMLR, 2023.
- 488 [75] D. Shah, B. Osiński, S. Levine, et al. Lm-nav: Robotic navigation with large pre-trained models
 489 of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023.
- 490 [76] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada. Saytap: Language to quadrupedal
 491 locomotion. *arXiv preprint arXiv:2306.07580*, 2023.
- 492 [77] Y. Ouyang, J. Li, Y. Li, Z. Li, C. Yu, K. Sreenath, and Y. Wu. Long-horizon locomotion and ma-
 493 nipulation on a quadrupedal robot with large language models. *arXiv preprint arXiv:2404.05291*,
 494 2024.
- 495 [78] A. Chen, A. Sharma, S. Levine, and C. Finn. You only live once: Single-life reinforcement
 496 learning. *Advances in Neural Information Processing Systems*, 35:14784–14797, 2022.
- 497 [79] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit,
 498 M. Ryoo, V. Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning
 499 with language. *arXiv preprint arXiv:2204.00598*, 2022.
- 500 [80] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer.
 501 Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint*
 502 *arXiv:2202.12837*, 2022.
- 503 [81] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui. A survey on
 504 in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

505 **A Appendix**

506 **A.1 Skill Details and Hyperparameters**

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

Table 2: Behavior: Walk

Parameter	Value
X-Velocity (Small)	0.4 m/s
X-Velocity (Medium)	0.6 m/s
X-Velocity (Large)	0.6 m/s
Y-Velocity	0 m/s
Gait Type	1
Body Height	0 m
Yaw Speed	0 rad/s
Duration (Small)	3 s
Duration (Medium)	5 s
Duration (Large)	7 s

Table 3: Behavior: Climb

Parameter	Value
X-Velocity	0.6 m/s
Y-Velocity	0 m/s
Gait Type	3
Body Height	0 m
Yaw Speed	0 rad/s
Duration (Small)	6 s
Duration (Medium)	9 s
Duration (Large)	12 s

Table 4: Behavior: Crawl

Parameter	Value
X-Velocity	0.3 m/s
Y-Velocity	0 m/s
Gait Type	1
Body Height	-0.3 m
Yaw Speed	0 rad/s
Duration (Small)	2 s
Duration (Medium)	3 s
Duration (Large)	4 s

Table 5: Behavior: Left Turn

Parameter	Value
X-Velocity	0 m/s
Y-Velocity	0 m/s
Gait Type	1
Body Height	0 m
Yaw Speed	0.3 rad/s
Duration (Small)	2.5 s
Duration (Medium)	3.5 s
Duration (Large)	4.5 s

Table 6: Behavior: Right Turn

Parameter	Value
X-Velocity	0 m/s
Y-Velocity	0 m/s
Gait Type	1
Body Height	0 m
Yaw Speed	-0.3 rad/s
Duration (Small)	2.5 s
Duration (Medium)	3.5 s
Duration (Large)	4.5 s

Table 7: Behavior: Backward

Parameter	Value
X-Velocity	-0.3 m/s
Y-Velocity	0 m/s
Gait Type	1
Body Height	0 m
Yaw Speed	0 rad/s
Duration (Small)	1.5 s
Duration (Medium)	2.5 s
Duration (Large)	5 s

Table 8: GPT-4o Query Hyperparameters

Parameter	Value
Temperature	0.7
Top P	0.95
Max Tokens	800

514 **A.2 Prompts and Logs**

515 In the following, we include the verbatim prompts used for VLM-PC as well as an example full chat
 516 log for one trial.

517 **A.2.1 VLM-PC Initial Prompt:**

518 You are controlling a robot dog to get through obstacles in this scene. The robot
 519 dog is the size of a golden retriever. The goal is to make forward progress in
 520 this starting direction and arrive at the location of a red chew toy on the
 521 other side of the obstacles in front of you. After each action you command the
 522 robot to take, you will be given a new image taken from the head of the robot
 523 dog. Each time you receive an image, you must command one of the following six
 524 actions: Walk, Crawl, Left, Right, Backward, Climb. You should command Crawl
 525 when it should go under an obstacle, command Climb when it should go over the
 526

527 obstacle. Examples of obstacles the robot can safely Climb over include stairs,
 528 small suitcases, cushions, small logs, and small boxes. Do not try to Crawl
 529 under these. Examples of obstacles the robot can safely Crawl under include
 530 furniture (if there is sufficient space) and suspended tree branches. Do not
 531 try to Climb furniture and suspended tree branches. Command Left when it should
 532 go left to get around an obstacle, and Right when it should go right to get
 533 around the obstacle. Right/Left command inplace turns, meaning that the robot
 534 will change the direction it is facing but not its position. This means that if
 535 you command a turn, you need to move forwards in order to move in the new
 536 direction. Do not try to turn Left or Right when you confidently believe you
 537 are underneath or above an obstacle. You should command the robot Backward if
 538 the camera is obscured and it is not clear what other action to take. If there
 539 is space to walk forward in front of you, you should command Walk. If you ever
 540 command Backward, do not try to do the same action you did before you commanded
 541 Backward again immediately (you may do so later once you are in a different
 542 scenario). If its possible to go under the obstacle, try to Crawl, and if its
 543 possible to go over an obstacle, try to Climb, but only if it seems like a
 544 clear path after the obstacle. If not, it might be a dead end and it might be
 545 better to move Backward. Keep in mind that the camera is on the head of the
 546 robot, and the whole body of the robot is 3 feet behind and also needs to clear
 547 the obstacle. Even if the camera view is clear, the body of the robot might be
 548 stuck: in these cases, consider Backward or repeating the same action as
 549 before. The task can be completed with the right sequence of actions, so if the
 550 robot is not succeeding, you should use the history of actions and experiences
 551 to figure out why and command different actions. When you chose Walk, Crawl,
 552 or Climb, you will move directly forward, so you may want to slightly turn to
 553 make sure you will move in the right direction before doing so.\nIn your first
 554 response, describe a multi-step plan using the available actions specified
 555 above for completing the task and give reasons that plan might fail. At each
 556 subsequent step, the beginning of your response should use the state of the
 557 robot, including its current position and orientation, to reason through what
 558 progress the robot has made with the plan. You should reconsider your multi-
 559 step plan of actions to take, keeping in mind the history of what actions you
 560 have selected and the progress in the scene. If the plan seems to be working,
 561 keep following actions from that plan. If an action from the plan does not seem
 562 to be working, either revise the plan or try a making a new multi-step plan
 563 with different actions. For example, if you have had many tries of Climbing
 564 over an obstacle and the robot is unable to make reliable forward progress, you
 565 might want to try going Backward or Crawling or going Left or Right around the
 566 obstacle. \nEach time you give a response, the third to last word of your
 567 response should be a yes or no indicating whether progress was made on the
 568 prior state with the previous command. The second to last word of your output
 569 should be the action to take. The last word of your response should be a
 570 magnitude (Small, Medium, Large) that specifies how much of that action to take.
 571 For Right or Left commands, magnitude Small corresponds to turning that
 572 direction a full camera view amount. For other commands, magnitude Small
 573 corresponds to taking that action for around one second. Once again, the last
 574 three words of your response must be in the format yes/no Walk/Crawl/Left/Right
 575 /Backward/Climb Small/Medium/Large.

577 A.2.2 VLM-PC Successive Chat Prompt:

578 "Based on your last action, the robot has moved. Here is what the robot
 579 currently sees. If the view is the same as before, this is likely because your
 580 last action was not appropriate for this situation and the body of the robot
 581 is probably stuck (maybe because of an unobserved obstacle). Keep in mind that
 582 this is the view from the head of the robot. Even if the camera view is clear,
 583 the body of the robot might be stuck. In these cases, consider Backward.
 584 Remember your ultimate objective of making progress in the direction the robot
 585 initially started in so that you eventually arrive at the red chew toy. Use
 586 the state of the robot including its current position and orientation to
 587 reason through what progress the robot has made with the plan. You reconsider
 588 your new multi-step plan of actions to take, keeping in mind the history of
 589 what actions you have selected and the progress in the scene. If the plan is
 590

591 working well, keep following actions from that plan, making slight adjustments
592 if necessary. If the plan is not working, try a new multi-step plan with
593 different actions. For example, if you had many tries of Climbing over an
594 obstacle and the robot is unable to make reliable forward progress, you might
595 want to try going Backward or Crawling or going Left or Right around the
596 obstacle. End your response with the first action in the chosen plan and a
597 magnitude in the format specified earlier."

599 Full example log with VLM-PC on the Outdoor 2 obstacle course:

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's possible to go under the obstacle, try to Crawl, and if it's 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

605

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's possible to go under the obstacle, try to Crawl, and if it's 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