A Pragmatist Robot: Learning to Plan Tasks by Experiencing the Real World

Kaixian Qu, Guowei Lan, René Zurbrügg, Changan Chen,
Christopher E. Mower, Haitham Bou-Ammar, and Marco Hutter

***Abstract*—Large language models (LLMs) have emerged as the dominant paradigm for robotic task planning using natural language instructions. However, trained on general internet data, LLMs are not inherently aligned with the embodiment, skill sets, and limitations of real-world robotic systems. Inspired by the emerging paradigm of verbal reinforcement learning—where LLM agents improve through self-reflection and few-shot learning without parameter updates—we introduce PRAGMABOT, a framework that enables robots to learn task planning through real-world experience. PRAGMABOT employs a vision-language model (VLM) as the robot's "brain" and "eye", allowing it to visually evaluate action outcomes and self-reflect on failures. These reflections are stored in a short-term memory (STM), enabling the robot to quickly adapt its behavior during ongoing tasks. Upon task completion, the robot summarizes the lessons learned into its long-term memory (LTM). When facing new tasks, it can leverage retrieval-augmented generation (RAG) to plan more grounded action sequences by drawing on relevant past experiences and knowledge. Experiments on four challenging robotic tasks show that STM-based self-reflection increases task success rates from 35% to 84%, with emergent intelligent object interactions. In 12 real-world scenarios (including eight previously unseen tasks), the robot effectively learns from the LTM and improves single-trial success rates from 22% to 80%, with RAG outperforming naive prompting. These results highlight the effectiveness and generalizability of PRAGMABOT. Project webpage: `https://pragmabot.github.io/`**

***Index Terms*—Learning from experience, task planning, AI-enabled robotics, embodied AI.**

I. INTRODUCTION

RECENTLY, large language models (LLMs) have demonstrated near-human performance across a range of reasoning tasks, showcasing emergent capabilities in diverse domains such as coding and law [1], [2], [3], [4]. These broad competencies have enabled LLMs to move beyond traditional language tasks and sparked interest in the field of robotics. In particular, they are now widely used in task planning, where LLMs interpret natural language instructions and generate feasible action plans with common-sense reasoning [5], [6], [7], [8], [9]. However, applying LLMs to robotics remains challenging, as they are designed exclusively for text processing, while robots must operate based on continuous, high-dimensional sensor streams.

Fig. 1: **Robot completes a new task guided by a long-term memory of self-reflective experiences.** When executing a novel task, the robot maintains a short-term memory that helps it reflect and learn how to complete the task (illustrated in the grey clip). The experience is then stored as long-term memory and retrieved to guide the VLM's task planning whenever a similar scenario is encountered (illustrated in the main figure).

To address the limitations of text-only input, research has increasingly shifted toward multimodal approaches, especially vision-language models (VLMs) that jointly process visual and textual data. Recent VLMs [10], [11], [12] exhibit strong multimodal reasoning and high-resolution visual processing. Building on these capabilities, recent work has leveraged VLMs to enable robots to reason about visual inputs and operate in closed-loop, autonomous settings [13], [14], [15], [16].

Yet, transferring these internet-trained models to physical robots remains challenging. While VLMs excel at abstract reasoning and visual understanding, they are not inherently aligned with the embodiment, skill sets, and limitations of real-world robotic systems. For example, in Figure 1, a VLM may confidently instruct a grasp on a partially occluded tennis ball, consistent with human intuition, yet the robot may fail due to limited manipulation capabilities under partial observability. This raises a critical question: How can the robot align the VLM with its own capabilities to achieve better task planning performance?

This paper introduces PRAGMABOT, a framework that enables robots to learn task planning through real-world experience. Our method is inspired by the principles of verbal reinforcement learning [17], where the LLM improves its performance via in-context learning on past self-reflective

K. Qu, G. Lan, R. Zurbrügg, C. Chen, and M. Hutter are with the Robotic Systems Lab, ETH Zürich, Switzerland. R. Zurbrügg is also with the ETH AI Center, ETH Zürich, Switzerland. C. E. Mower and H. Bou-Ammar are with Huawei Noah's Ark Lab, London, UK. H. Bou-Ammar is also with the UCL Centre for AI, London, UK. Corresponding author: K. Qu (e-mail: kaixqu@ethz.ch).

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab), by Huawei Tech R&D (UK) through a research funding agreement, by an ETH RobotX research grant funded through the ETH Zürich Foundation, and partially by the ETH AI Center. This work was also conducted as part of ANYmal Research, a community to advance legged robotics. We would also like to thank Cesar Cadena for his support and helpful discussions.

experience, without incurring the expense of fine-tuning the LLM through weight updates. PRAGMABOT utilizes a VLM in three different ways: (i) to plan actions for execution, (ii) to verify that the action was completed successfully, and (iii) to summarize experiences. It maintains a short-term memory (STM) that keeps track of executed actions and associated feedback signals, along with a long-term memory (LTM) that stores lessons learned from past successful task executions. PRAGMABOT uses the STM to perform self-reflection, allowing the robot to adapt its behavior toward achieving the task goal. Upon successful task completion, the VLM summarizes the STM experience and stores it in the LTM. When presented with a similar task in the future, the robot employs retrieval-augmented generation (RAG) to retrieve relevant knowledge from the LTM for planning actions while accounting for its own capabilities and limitations. Additionally, PRAGMABOT enhances the spatial understanding of VLMs through an on-demand image annotation module.

This paper presents the following key contributions:

- We introduce PRAGMABOT, which adapts verbal reinforcement learning to real-world robotic task planning by integrating VLMs with visual feedback. We demonstrate that the combination of short-term and long-term memory enables efficient learning of task planning tailored to the robot's embodiment and capabilities.
- We introduce retrieval-augmented generation (RAG) into the verbal reinforcement learning framework, allowing the robot to selectively leverage task-relevant, self-reflective experiences. Extensive evaluations show that RAG improves planning accuracy significantly compared to naive prompting.
- We design an on-demand image annotation module that enhances the VLM's spatial reasoning across diverse skills. We demonstrate that this module leads to more accurate action execution in complex, real-world environments.

Extensive real-world experiments show that PRAGMABOT significantly outperforms state-of-the-art methods in task success rates and generalizes well to similar but previously unseen tasks. To support future research, our code is available on the project webpage[1].

II. RELATED WORK

Recent advances in robotic task planning have explored methods enabling robots to understand their own capabilities and limitations. SayCan [6] grounds language understanding in robotic affordances by combining LLM outputs with a trained visual affordance network, which requires extensive training resources. Alternative frameworks focus on building robotic memory through dense human feedback and corrections to improve decision-making [18], [19]. Building on this direction, BUMBLE [20] integrates short-term memory for online replanning with long-term memory of human-annotated failure cases to guide VLMs in avoiding past errors. However, these methods typically rely heavily on dense human supervision for feedback and correction.

[1]https://pragmabot.github.io/

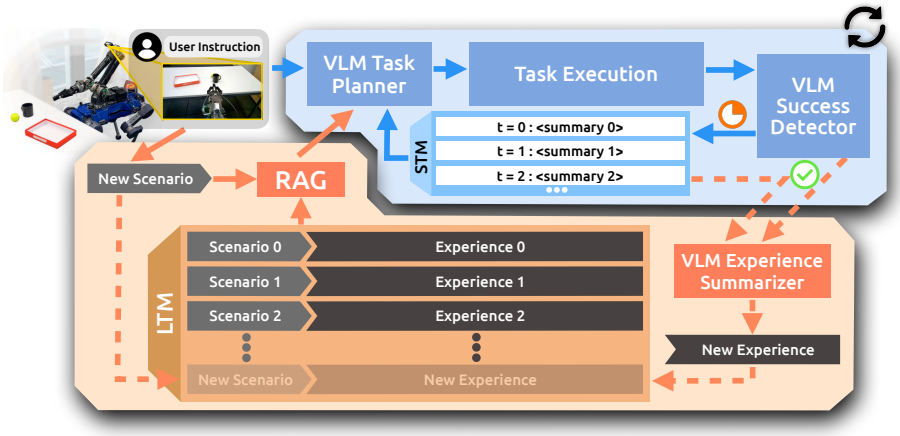
TABLE I: **Comparison with baselines.** PRAGMABOT is the only method that achieves self-reflection, learning by experiencing, interactive replanning, and creative tool use (highlighted in bold). System components are listed in regular font (• indicates presence).

	Self-reflection	**Learning by exp.**	**Interactive replan**	**Creative tool use**	Short-term memory	Long-term memory	Unified visual feedback
CaP [5]	✗	✗	✗	✗	○	○	○
SayCan [6]	✗	✗	✗	✗	•	○	○
Inner Mono. [8]	✗	✗	✗	✗	•	○	○
RoboTool [23]	✗	✗	✗	✓	○	○	○
DROC [18]	✗	✓	✗	✗	•	•	○
REFLECT [21]	✓	✗	✓	✗	•	○	○
COME [14]	✓	✗	✗	✗	•	○	•
ReplanVLM [15]	✓	✗	✓	✗	•	○	•
BUMBLE [20]	✓	✗	✓	✗	•	•	○
PRAGMABOT	✓	✓	✓	✓	•	•	•

More recent work explores autonomous failure recovery through self-reflection mechanisms. Systems such as [21], [22] enable LLM/VLM-powered robots to analyze their own failures and adapt subsequent actions to accelerate task completion. Building on this idea, Reflexion [17] goes further by not only allowing the LLM agent to reflect on its failures but also storing these self-reflective experiences in long-term memory to inform future behavior. This approach, known as verbal reinforcement learning, improves agent performance not by updating model weights, but by incorporating additional contextual information for reasoning. While Reflexion eliminates the need for extensive training data, it has so far been evaluated primarily in simulated environments that do not account for real-world embodiment or the complexities of physical interaction. Moreover, observations must be supplied either as ground truth from the simulator or via external, hand-engineered scene descriptors.

PRAGMABOT investigates how physical robots can visually evaluate the outcomes of their actions in challenging tasks, self-reflect on failures, and construct long-term memory to guide future planning—all through real-world experience. It enables the system to learn without relying on explicit human annotation or dense human feedback. Additionally, we incorporate Retrieval-Augmented Generation (RAG) to retrieve relevant past experiences from memory. While RAP [24] proposes using RAG with a memory built from diverse experiences for simulated LLM agents, it does not consider self-reflection with VLMs. Table I compares PRAGMABOT with prior LLM/VLM task planners. "Learning by experiencing" implies that the robot learns from failure, adapts, and memorizes the solution for future planning (with or without human correction). "Interactive replanning" refers to interacting with non-target objects after failure to aid task completion. "Creative tool use" involves autonomously using unmentioned objects as tools. "Unified visual feedback" refers to the robot using the same VLM for both planning and verification, thereby aligning more closely with the concept of self-reflection.

Beyond task planning, VLMs can also be used to determine



Algorithm 1 PRAGMABOT

Given: Instruction $\mathbf{I}$, initial observation $\mathbf{o}_0$
Internal: Long-term memory $\mathbf{M}$

$\mathbf{K}' \leftarrow (\mathbf{I}, \mathcal{D}(\mathbf{o}_0))$
$\{(\mathbf{K}_i, \mathbf{E}_i)\}_{i=0}^{k-1} \leftarrow \arg\operatorname{top\,k}_{(\mathbf{K},\mathbf{E})\in\mathbf{M}} \left[\frac{\mathcal{E}(\mathbf{K})^T \mathcal{E}(\mathbf{K}')}{\|\mathcal{E}(\mathbf{K})\|\|\mathcal{E}(\mathbf{K}')\|}\right]$
$t \leftarrow 0$, $\mathbf{m} \leftarrow \emptyset$
repeat
  $\mathbf{a}_t \leftarrow \mathcal{P}\left(\mathbf{I}, \mathbf{o}_t, \mathbf{m}, \{(\mathbf{K}_i, \mathbf{E}_i)\}_{i=0}^{k-1}\right)$
  Execute $\mathbf{a}_t$ and receive $\mathbf{o}_{t+1}$
  $\mathbf{r}_{t+1} \leftarrow \mathcal{R}(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$
  $\mathbf{m} \leftarrow \mathbf{m} \cup (\mathbf{a}_t, \mathbf{r}_{t+1})$
  $t \leftarrow t + 1$
until $\mathbf{r}_t$.completed
$\mathbf{M} \leftarrow \mathbf{M} \cup \{(\mathbf{K}', \mathcal{S}(\mathbf{m}))\}$

Fig. 2: **PRAGMABOT pipeline (illustration: *left*, algorithm: *right*).** At the start of each task, the system takes the user instruction $\mathbf{I}$ and egocentric observation $\mathbf{o}_0$, which the VLM combines into a scenario (key). RAG retrieves relevant experiences from long-term memory $\mathbf{M}$ and, together with the instruction and observation, feeds them into the VLM task planner $\mathcal{P}$ to generate the next action $\mathbf{a}$. After execution, success is checked by the VLM success detector $\mathcal{R}$. If the task is not completed, the action $\mathbf{a}$ and its feedback $\mathbf{r}$ are accumulated into short-term memory $\mathbf{m}$ and fed back into planning. Once the task is completed, the short-term memory $\mathbf{m}$ is summarized and stored in long-term memory $\mathbf{M}$ for future use.

precise action parameters through mask-based techniques [25], [26]. Recent approaches combine object-centric annotations with grasping tools to support challenging semantic manipulation tasks [27], [28], [29]. However, these methods are typically limited to planar tabletop settings or grasping-only scenarios. In contrast, PRAGMABOT introduces an on-demand image annotation tool that enables grounded actions in 3D space, supporting a broader range of skills beyond 6-DoF grasping.

III. PROBLEM FORMULATION

Consider a robot system equipped with K predefined parameterized skills $\{\pi_k\}_{k=1}^{K}$, where each π_k is a low-level policy mapping observations (e.g., camera inputs) to actionable commands (e.g., joint motions), operating until certain termination conditions are met. These skills are assumed to be provided in advance, either learned through imitation or reinforcement learning, or implemented as optimal controllers such as model-predictive control.

We focus on task planning—specifically, how the robot can effectively sequence these skills to accomplish a task described by a human in natural language. To address this, we introduce a higher-level skill selection policy Π, implemented using foundation models such as VLMs. At each time step, the selected skill is sampled according to

$$\pi_k \sim \Pi_\theta(\mathbf{I}, \mathbf{o}_t, \mathbf{c}_t), \tag{1}$$

where $\mathbf{I}$ denotes the natural language instruction, $\mathbf{o}_t$ is the current environmental observation (e.g., an image from the robot's camera), $\mathbf{c}_t$ represents additional contextual information, and θ denotes the parameters of the VLM. We aim to investigate how to effectively adapt and improve the skill selection policy (1) without requiring extensive training resources or dense human supervision.

IV. METHOD

PRAGMABOT enables the robot to learn to plan tasks through interaction with the real world, following a paradigm akin to reinforcement learning. This is achieved through several core components: a success detector, a memory mechanism, and a memory retrieval strategy. Each component plays a distinct role in the learning pipeline, as detailed below. An overview of the full system is illustrated in Figure 2.

A. Task Planner

The task planner $\mathcal{P}$ leverages a VLM to interpret the user instruction $\mathbf{I}$ and, conditioned on the current RGB observation $\mathbf{o}_t$, outputs the next action $\mathbf{a}_t$—which consists of a selected skill π_k along with its associated parameters. This behavior is governed by an underlying policy Π_θ, which samples actions using the VLM's visual-language understanding and commonsense reasoning capabilities encoded in its parameters θ. However, this initial policy is insufficient on its own—it lacks awareness of the robot's capabilities and limitations. To bridge this gap, $\mathcal{P}$ must be improved through real-world interaction, adapting its decisions to better align with the robot's embodiment.

B. Success Detector

The success detector $\mathcal{R}$, also implemented as a VLM—the same model used by $\mathcal{P}$—provides a feedback signal $\mathbf{r}$ that assesses the outcome of an executed action $\mathbf{a}$ by comparing the observations before and after the action:

$$\mathbf{r}_{t+1} \leftarrow \mathcal{R}(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1}) \tag{2}$$

The resulting feedback signal provides two binary indicators: whether the action was successful and whether the overall task is complete. Additionally, it includes a semantic description of the scene changes induced by the action, which can be leveraged by the task planner to determine subsequent actions.

C. Short-Term Memory and Online Adaptation

PRAGMABOT keeps a short-term memory (STM) module, denoted $\mathbf{m}$, which functions as a dynamic log that interacts iteratively with $\mathcal{P}$ and $\mathcal{R}$. It stores a sequence of past interactions up to time t, where each entry consists of an executed action $\mathbf{a}_\tau$ and the corresponding feedback signal $\mathbf{r}_{\tau+1}$ generated by $\mathcal{R}$:

$$\mathbf{m} = \{(\mathbf{a}_\tau, \mathbf{r}_{\tau+1})\}_{\tau=0}^{t-1}. \tag{3}$$

Upon action failure, $\mathcal{P}$ performs self-reflection, identifying potential causes and propose better ways to complete the task. This is part of the chain-of-thought before outputting the final action [2]. This process echoes gradient computation in RL, but instead of updating network weights, the planner uses this "linguistic gradient"—expressed in natural language—to improve its behavior. For example, if the robot fails to pick up an apple partially occluded by a can, the linguistic gradient might read: "Previous attempts to pick up the apple directly have failed. The apple is next to a cylindrical container, which might be causing interference. To create more space and ensure a successful grasp, I will push the can to the right, away from the apple. This should allow for a clearer path to pick up the apple." In addition, we observe that the robot is capable of learning to use tools for manipulating small objects after an initial action failure. This STM serves as a crucial component for online adaptation, effectively substituting weight updates with in-context learning guided by linguistic feedback.

D. Long-Term Memory and Experience Summarization

The STM is episodic and resets upon task completion, thus it does not retain knowledge across tasks. To enable cumulative learning and transfer of experience over time, we introduce a persistent long-term memory (LTM), denoted $\mathbf{M}$. Upon successful task completion, the robot summarizes its STM into LTM using a VLM-based experience summarizer $\mathcal{S}$. This summary is stored as a key-value pair $(\mathbf{K}, \mathbf{E})$, where the key $\mathbf{K}$ is a scenario description and the value $\mathbf{E} = \mathcal{S}(\mathbf{m})$ is the corresponding summarized experience:

$$\mathbf{M} \leftarrow \mathbf{M} \cup \{(\mathbf{K}, \mathbf{E})\}. \tag{4}$$

The key $\mathbf{K}$ combines two components: (1) the user instruction $\mathbf{I}$ and (2) a natural language description of the initial scene, generated by a VLM scene describer $\mathcal{D}$ from the first RGB observation $\mathbf{o}_0$ (e.g., "The apple is on the right side of the table, next to a salt container. The plate is on the left side, with ample space around it."). This combined context provides a rich, semantically meaningful index for future retrieval. To enable fast and effective lookup, the key $\mathbf{K}$ is embedded into a dense vector using a text embedding model $\mathcal{E}$ and cached in the LTM. In this way, the LTM facilitates lifelong learning by transforming transient episodic memories into reusable, generalizable knowledge.

E. PragmaBot Algorithm

When presented with a new task, the task planner $\mathcal{P}$ leverages past experience through a retrieval-augmented generation (RAG) mechanism. A retrieval key $\mathbf{K}'$ is constructed from the current instruction $\mathbf{I}$ and the initial scene description. Using the text-embedding model $\mathcal{E}$, the system computes the embedding of $\mathbf{K}'$ and retrieves the top-k most similar past experiences from the LTM $\mathbf{M}$ via cosine similarity:

$$\{(\mathbf{K}_i, \mathbf{E}_i)\}_{i=0}^{k-1} \leftarrow \underset{(\mathbf{K},\mathbf{E})\in\mathbf{M}}{\arg\operatorname{top\,k}} \left[\frac{\mathcal{E}(\mathbf{K})^T \mathcal{E}(\mathbf{K}')}{\|\mathcal{E}(\mathbf{K})\| \, \|\mathcal{E}(\mathbf{K}')\|} \right]. \tag{5}$$

These retrieved experiences are incorporated into the planning prompt, enabling $\mathcal{P}$ to leverage in-context learning by drawing on prior knowledge to make more informed decisions in new but related scenarios.

The STM, initialized as an empty set, is provided to the robot for online adaptation. At each time step, the task planner $\mathcal{P}$ (the high-level policy to be learned) generates an action based on the current observation, the current STM, and the retrieved LTM entries. After execution, the success detector $\mathcal{R}$ evaluates the action outcome and provides a success or failure signal as the feedback. Upon action failure, the planner performs self-reflection and takes a step along the linguistic gradient to select the next action. Upon task completion, the STM is summarized and stored in LTM for future use.

This closed-loop process—combining retrieval from long-term memory, in-context adaptation via short-term memory, and experience accumulation—forms the core of the PRAGMABOT framework. It enables both rapid initialization on new tasks and continuous improvement over time, supporting effective lifelong learning in real-world environments. The complete algorithm is outlined in Algorithm 1. The prompt templates for VLM modules are shown in Figure 3; full examples are available on the project webpage.

Scene Describer

Instruction: <Instruction>.

Robot's current observation: <IMG>.

Provide a short scene description.

Experience Summarizer

Instruction: <Instruction>.

Scene: <Initial scene description>.

Summarize the robot's experiences: <STM>.

Task Planner

Instruction: <Instruction>.

Robot's current observation: <IMG>.

Here is the action history: <STM>.

List of planning rules and constraints.

Here are the past experiences: <LTM>.

Choose the next best action.

Success Detector

Instruction: <Instruction>.

Action the robot just attempted: <Action>.

Observation before the action: <IMG>. Observation after the action: <IMG>.

List of success/failure criteria and reasoning rules.

Output your structured evaluation.

Fig. 3: **Prompt templates used in different VLM modules.**

F. Enhanced Skillset with Image Annotations

For the low-level skills, we focus on three manipulation ones—pick, place, and push. Identifying a suitable location for each action is non-trivial: purely geometric grasp planning may seize an undesirable part (e.g., the meat on a skewer or the ice-cream top rather than its cone), and effective placing or pushing likewise demands semantic scene understanding. To address these challenges, we introduce an on-demand image-annotation tool that is shared across all skills. Given a user instruction $\mathbf{I}$ and the current RGB frame $\mathbf{o}_t$, VLM first selects the skill to execute with its parameters (e.g., object name, whether

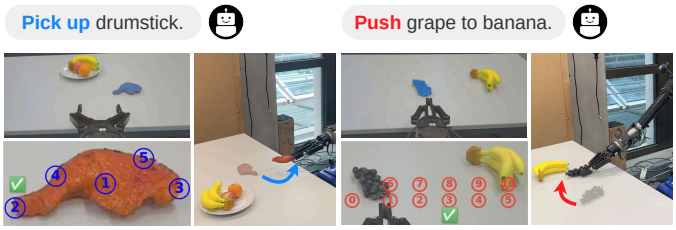


Fig. 4: **Illustration of image annotation tools.**

image annotation is needed). The robot then performs open-vocabulary segmentation with Grounded SAM [30], which integrates Grounding DINO [31] and SAM [32], producing an initial object mask. If the VLM requests a second annotation pass, our image overlays a set of candidate location masks on the image, allowing the VLM to choose the most appropriate location for the current action.

For placing, we apply farthest-point sampling (FPS) [33] on the segmented mask to generate candidate placement locations, whereas for pushing, we draw candidate goal masks that denote the end points. In both cases, the VLM evaluates the annotated options and selects the most suitable location, as illustrated in Figure 4. For grasping, the segmented point cloud is first passed to AnyGrasp [34], which returns a set of grasp hypotheses accompanied by confidence scores $s_{conf} \in [0, 1]$. Grasp poses that violate kinematic constraints are filtered out through inverse-kinematics checks using Pinocchio [35], yielding the feasible subset $\mathcal{G}$. If the VLM determines that image annotation is beneficial (typically not the case for simple objects like an apple), our annotation tool similarly performs FPS within the object mask to generate a collection of numbered location masks for the VLM to select from. The final grasp is selected by maximizing the product of two scores:

$$g^* = \arg\max_{g \in \mathcal{G}} s_{conf}(g) \cdot s_{loc}(g), \tag{6}$$

where $s_{loc}(g) \in [0, 1]$ denotes the location score, computed based on the normalized Euclidean distance between grasp g and the chosen location.

V. Results

A. Experiment Setup

In our experiments, we use a legged manipulator that combines ANYmal [36], a quadrupedal robot, with a 6-DoF arm. The arm is equipped with a Robotiq 2F-140 gripper for object manipulation and a ZED X Mini Stereo Camera mounted on the elbow for perception. We employ `gpt-4o` [10] for the VLM model and use OpenAI's `text-embedding-3-large` [37] for the text embedding model $\mathcal{E}$. If the robot fails an action and alters the environment, a human operator can choose to reset the scene, after which the robot resumes execution.

B. Evaluation of Short-Term Memory and Self-Reflection

To evaluate the effectiveness of the STM and reflection module at efficiently generating successful episodes, even after initial failures, we designed four challenging object manipulation tasks (top two rows in Figure 5a). For the baseline, we use CaP-V, which enhances CaP [5] by incorporating visual feedback. Note that CaP-V does not have STM and selects the next action solely based on the current image and user instruction, without the ability to reflect on failed actions. Experimental results in Table II highlight the critical role of STM and reflection in achieving successful task completion. Without STM, the robot tends to repeat the same failures without adapting, leading to poor task performance.

TABLE II: **Effect of STM on task success rates.** Each task is tested 5–10 times with two attempts allowed.

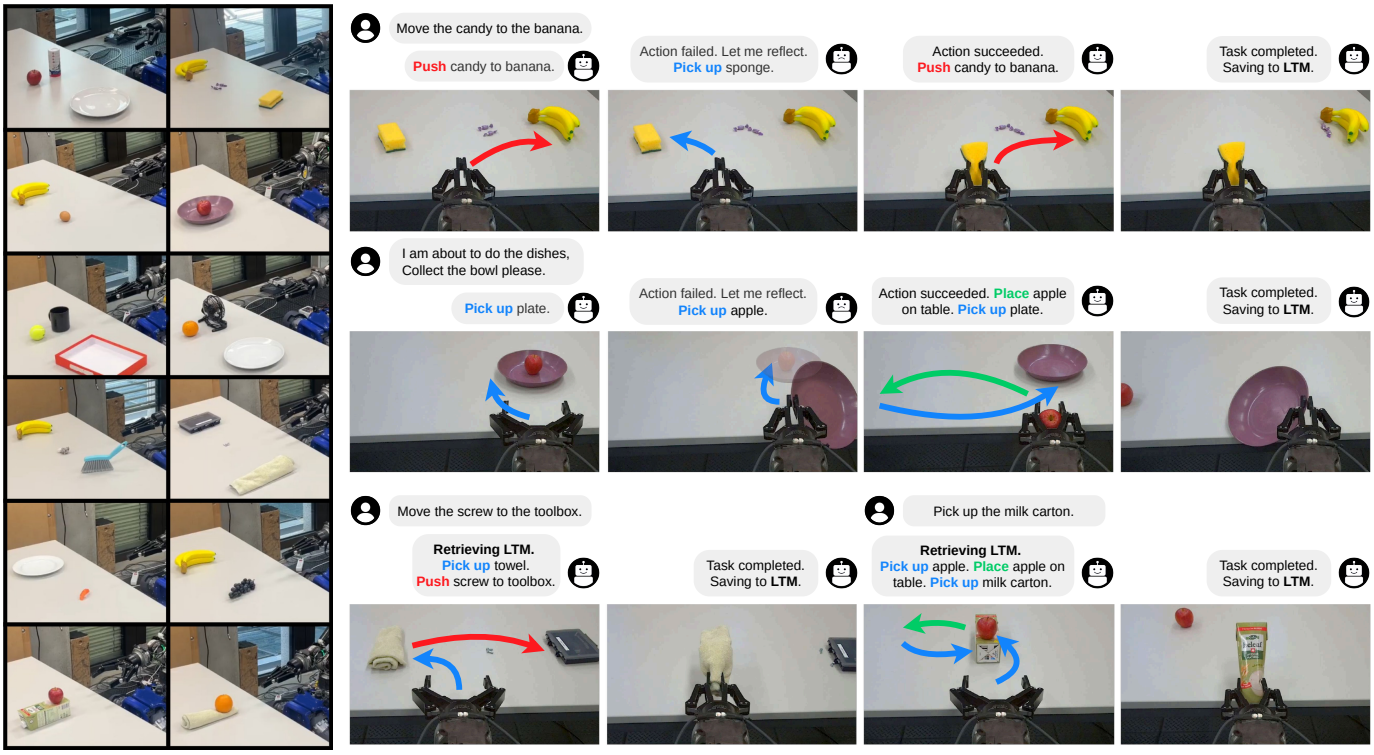
Task	CaP-V	PragmaBot
Put apple on plate (container obstructs)	43%	**86%**
Move tiny candy (sponge/towel nearby)	22%	**67%**
Move egg (open view)	40%	**100%**
Pick up bowl (apple inside)	33%	**83%**

With STM, the robot can successfully reflect on its failures, leading to emergent intelligent object interactions (including tool use) and ultimately task success. For example, when instructed to "put the apple on the plate," and faced with a partially obstructing container, the robot initially fails to grasp the apple due to poor perception and obstruction. After detecting the failure, the VLM decides to push away the container and successfully retries the grasp (see Figure 1). When asked to "collect the bowl," where an apple is already inside, the robot initially tries to lift the bowl directly. However, the apple falls out during the action execution, and the robot reflects and revises its plan: "I should first move the apple to the table before picking up the bowl" (see Figure 5b). Similarly, when told to "move the candy to the banana," the robot fails to push the candy with its gripper due to insufficient contact. Upon reflection, the VLM autonomously chooses to use a sponge as a tool to push more effectively (see Figure 5b). After cracking an egg while grasping, the VLM similarly learns to push rather than grasp fragile objects.

Note that these instances of autonomous intelligent adaptation would not be possible without the high accuracy of the VLM's success detection. In object-picking tasks, we observe a false negative rate of 6.67% (4/60) and a false positive rate of 5% (3/60), with most false positives occurring when the object remains on the table but appears visually enclosed in the gripper.

C. Evaluation of Long-Term Memory and Generalization

Upon completing the task in Section V-B, the STM is autonomously transferred to the LTM for future use. Together with 96 limited instructional experiences from simpler tasks, we build an LTM containing 100 entries. For the baseline, we use COME [14], which does not have access to the LTM. We keep this generated LTM frozen during evaluation for a fair comparison. We first evaluate the effectiveness of the LTM in the same four scenes with the same objects. As shown in Table III, the robot successfully recalls the correct action from the LTM with RAG, demonstrating improved performance compared to COME. We further investigate its ability to generalize to other similar scenarios. To do so,



(a) Illustration of all scenes.

(b) Examples of PRAGMABOT on different scenarios.

Fig. 5: **Overview of all experimental scenes (a) and demonstration of PRAGMABOT's performance across four representative scenarios (b).** In the first two examples (top and middle rows), the robot successfully completes the tasks after self-reflection. These experiences are then summarized and stored in LTM, enabling the robot to generalize its learning to similar future scenarios (bottom row).

TABLE III: **Effect of LTM on single-trial task success rates.** Each task is tested 5–10 times.

Task	COME [14]	PRAGMABOT
Put apple on plate (container obstructs)	29%	**100%**
Move tiny candy (towel nearby)	11%	**78%**
Move egg (open view)	20%	**100%**
Pick up bowl (apple inside)	17%	**83%**
Put tennis ball in box (mug obstructs)	29%	**71%**
Put orange/ball on plate (fan blocks)	10%	**80%**
Move crumpled paper (brush nearby)	25%	**63%**
Move screw (towel nearby)	0%	**86%**
Move sushi (open view)	14%	**71%**
Move grape/cherry (open view)	20%	**70%**
Pick up box (apple on top)	43%	**86%**
Pick up towel (orange on top)	50%	**75%**

we modify the scene to create new but structurally similar scenarios, as illustrated in Figure 5a (bottom four rows). The results in Table III show that experience gained from one task successfully transfers to related tasks under similar conditions and significantly improves the task success rate. For example, when asked to "Move the screw to the toolbox," the robot immediately decides to use a towel to push the screw successfully. Similarly, when tasked with "Pick up the milk carton," it remembers to reposition the apple first before retrieving the target item.

We analyzed the 19 first-failure cases out of 91 total trials, as shown in Figure 6. Of these failures, 8 stem from action execution issues—either poor grasp generation or inaccurate depth estimates from the stereo camera. Another 7 failures

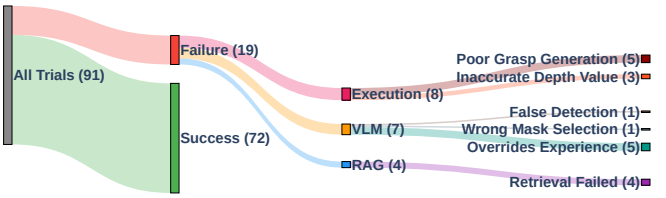


Fig. 6: **Failure flow diagram illustrating the sources of initial failure across two hierarchical levels.**

arise from VLM reasoning errors, most notably when the VLM perceives the retrieved memory as misaligned with the current visual observation and consequently downweights or ignores the relevant experience. For instance, in the "Put tennis ball in the box (mug obstructs)" scenario, the retrieved memory involved clearing a similar occlusion, but the VLM might deem the mug's obstruction negligible and overrode that strategy, leading to failure. The remaining 4 failures occur when the RAG fails to retrieve the relevant experience.

D. Ablation Study of Memory Retrieval

To evaluate the effectiveness of our memory retrieval strategy, we measure the accuracy of the first planned action (without execution) across 12 tasks under three retrieval settings, as shown in Figure 7. We reuse the LTM described in Section V-C. Randomly selecting $k = 5$ memories yields the worst performance, with an average accuracy of only 17% on unseen tasks, as task-relevant experiences are rarely retrieved

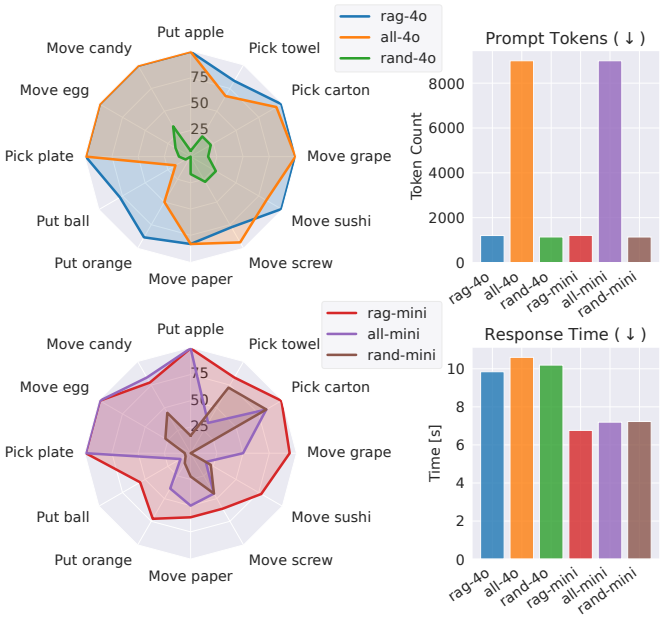


Fig. 7: **Ablation study of the memory retrieval module.** The radial axes of the radar charts represent the accuracy of the first planned action in percentage. RAG with `gpt-4o` (rag-4o) provides the highest accuracy.

by chance. Providing the entire LTM improves accuracy to 74%, but unfiltered retrieval introduces irrelevant or distracting information, leading to unstable behavior. This aligns with prior findings that excessively long or noisy contexts can degrade model performance, as LLMs may struggle to focus on the most relevant content [38], [39]. In contrast, our RAG-based retrieval strategy achieves the highest accuracy at 89%. We also evaluated PRAGMABOT using the smaller model `gpt-4o-mini`. We found that this smaller model tends to behave more conservatively, making it more aligned with our robot when it does not have access to relevant experiences. When provided with relevant memories, `gpt-4o-mini` also shows clear performance gains, though the improvement is less pronounced than for larger models. Consequently, both rag-4o and all-4o outperform their mini counterparts, albeit at the cost of higher latency. Moreover, feeding the full LTM increases text prompt length by a factor of 7.5 (excluding images), resulting in substantially higher monetary cost.

E. Ablation Study of Image Annotation Module

To evaluate the effectiveness of image annotation on grasping tasks, we compared success rates with and without it across 7 objects. A grasp was considered successful if it targeted the correct object section (e.g., the stick of a meat skewer). As shown in Figure 8, annotation significantly improves success rates for objects with complex shapes that require grasping specific sections (e.g., drumsticks, skewers). Without annotation, AnyGrasp [34] often favors larger surfaces due to its reliance on geometric cues. The failures with annotation are mostly due to inaccurate 3D point clouds. We also evaluated pushing by measuring the distance error to the target location and found that image annotation consistently reduces this error, highlighting its benefits beyond grasping. While effective, this method introduces additional latency, requiring an average of 5.15 s to retrieve a response from `gpt-4o`.

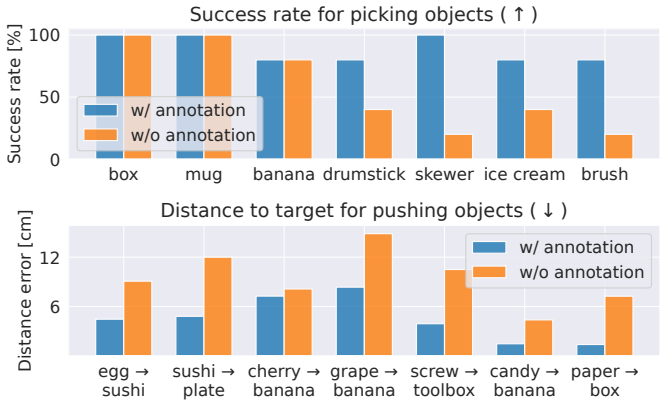


Fig. 8: **Ablation study of the image annotation module.** *Top:* success rates for picking objects (higher is better). *Bottom:* distance errors in pushing one object to another (lower is better). The results demonstrate that incorporating image annotation consistently enhances performance.

VI. CONCLUSION AND FUTURE WORK

This work introduces PRAGMABOT, a novel method that enables robots to learn to plan tasks by experiencing the real world. Empirical results demonstrate that PRAGMABOT allows robots to autonomously reflect and adapt using short-term memory, significantly improving task success rates and facilitating intelligent object interactions—such as creative tool use. Furthermore, these experiences can be stored in the long-term memory, enabling the robot to plan correctly on its first attempt in the future, even in previously unseen scenarios. Extensive evaluations show that integrating this persistent memory with RAG yields substantial performance gains over current state-of-the-art methods across 12 challenging scenarios. Our framework provides a general and efficient paradigm that does not rely on substantial training resources, making it well-suited for deployment in real-world robotic systems.

While PRAGMABOT demonstrates strong potential in learning task planning, it still presents several limitations. In practice, certain real-world scenarios involve information that cannot be captured by vision alone. Integrating additional modalities–such as tactile or auditory signals–could greatly improve the system's capacity to interpret complex, multimodal feedback. Second, our current memory database is limited by the cost of real-world data collection. Scaling to much larger memory databases raises important questions: should memory pruning be applied when the database becomes extremely large? If so, which memories should be retained and which forgotten? Could the VLM itself perform filtering without relying heavily on human-designed heuristics? Moreover, as the memory grows, our current top-k retrieval mechanism may become ineffective, warranting exploration of more sophisticated strategies—such as maximum marginal relevance (MMR). Finally, sharing memories among multiple

robots is a promising direction. This is clearly feasible when robots share identical hardware and skillsets, but what if they have similar morphologies (e.g., another quadrupedal robot with a single arm, but much smaller)? Could a robot initially attempt to apply transferred memories and, upon failure, discard unsuitable experiences and replace them with its own? These are compelling directions for future work.

REFERENCES

[1] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[2] 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.

[3] J. Ahn, R. Verma, R. Lou, D. Liu, R. Zhang, and W. Yin, "Large language models for mathematical reasoning: Progresses and challenges," *arXiv preprint arXiv:2402.00157*, 2024.

[4] Z. Fei, X. Shen, D. Zhu, F. Zhou, Z. Han, S. Zhang, K. Chen, Z. Shen, and J. Ge, "Lawbench: Benchmarking legal knowledge of large language models," *arXiv preprint arXiv:2309.16289*, 2023.

[5] 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.

[6] 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.

[7] C. E. Mower, Y. Wan, H. Yu, A. Grosnit, J. Gonzalez-Billandon, M. Zimmer, J. Wang, X. Zhang, Y. Zhao, A. Zhai *et al.*, "Ros-llm: A ros framework for embodied ai with task feedback and structured reasoning," *arXiv preprint arXiv:2406.19741*, 2024.

[8] 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," in *Conference on Robot Learning*. PMLR, 2023, pp. 1769–1782.

[9] 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," in *Conference on Robot Learning*. PMLR, 2023, pp. 540–562.

[10] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[11] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4," *arXiv preprint arXiv:2303.12712*, 2023.

[12] G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang *et al.*, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," *arXiv preprint arXiv:2403.05530*, 2024.

[13] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[14] P. Zhi, Z. Zhang, Y. Zhao, M. Han, Z. Zhang, Z. Li, Z. Jiao, B. Jia, and S. Huang, "Closed-loop open-vocabulary mobile manipulation with gpt-4v," *arXiv preprint arXiv:2404.10220*, 2024.

[15] A. Mei, G.-N. Zhu, H. Zhang, and Z. Gan, "Replanvlm: Replanning robotic tasks with visual language models," *IEEE Robotics and Automation Letters*, 2024.

[16] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao, "Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning," *arXiv preprint arXiv:2311.17842*, 2023.

[17] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 8634–8652, 2023.

[18] 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," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 15 172–15 179.

[19] L. Bärmann, R. Kartmann, F. Peller-Konrad, J. Niehues, A. Waibel, and T. Asfour, "Incremental learning of humanoid robot behavior from natural interaction and large language models," *Frontiers in Robotics and AI*, vol. 11, p. 1455375, 2024.

[20] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. Martín-Martín, "Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation," 2025.

[21] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," in *Conference on Robot Learning*. PMLR, 2023, pp. 3468–3484.

[22] Z. Wang, B. Liang, V. Dhat, Z. Brumbaugh, N. Walker, R. Krishna, and M. Cakmak, "I can tell what i am doing: Toward real-world natural language grounding of robot experiences," in *8th Annual Conference on Robot Learning*, 2024.

[23] M. Xu, P. Huang, W. Yu, S. Liu, X. Zhang, Y. Niu, T. Zhang, F. Xia, J. Tan, and D. Zhao, "Creative robot tool use with large language models," *arXiv preprint arXiv:2310.13065*, 2023.

[24] T. Kagaya, T. J. Yuan, Y. Lou, J. Karlekar, S. Pranata, A. Kinose, K. Oguri, F. Wick, and Y. You, "Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents," in *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

[25] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao, "Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v," *arXiv preprint arXiv:2310.11441*, 2023.

[26] 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 vlms," in *International Conference on Machine Learning*. PMLR, 2024, pp. 37 321–37 341.

[27] K. Fang, F. Liu, P. Abbeel, and S. Levine, "Moka: Open-world robotic manipulation through mark-based visual prompting," *Robotics: Science and Systems (RSS)*, 2024.

[28] G. Tziafas and H. Kasaei, "Towards open-world grasping with large vision-language models," in *8th Annual Conference on Robot Learning*, 2024.

[29] Y. Qian, X. Zhu, O. Biza, S. Jiang, L. Zhao, H. Huang, Y. Qi, and R. Platt, "Thinkgrasp: A vision-language system for strategic part grasping in clutter," *arXiv preprint arXiv:2407.11298*, 2024.

[30] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.

[31] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European Conference on Computer Vision*. Springer, 2024, pp. 38–55.

[32] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.

[33] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE transactions on image processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[34] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3929–3945, 2023.

[35] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[36] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[37] OpenAI, "New embedding models and api updates," 2024, accessed: 2025-08-08. [Online]. Available: https://openai.com/index/new-embedding-models-and-api-updates/

[38] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024.

[39] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval augmentation reduces hallucination in conversation," *arXiv preprint arXiv:2104.07567*, 2021.