2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Searching in Space and Time: Unified Memory-Action Loops for Open-World Object Retrieval

Taijing Chen[1], Sateesh Kumar[1], Junhong Xu[1], Georgios Pavlakos[1], Joydeep Biswas[1,*], Roberto Martín-Martín[1,*]

***Abstract*— Service robots must retrieve objects in dynamic, open-world settings where requests may reference attributes ("the red mug"), spatial context ("the mug on the table"), or past states ("the mug that was here yesterday"). Existing approaches capture only parts of this problem: scene graphs capture spatial relations but ignore temporal grounding, temporal reasoning methods model dynamics but do not support embodied interaction, and dynamic scene graphs handle both but remain closed-world with fixed vocabularies. We present STAR (*SpatioTemporal Active Retrieval*), a framework that unifies memory queries and embodied actions within a single decision loop. STAR leverages non-parametric long-term memory and a working memory to support efficient recall, and uses a vision-language model to select either temporal or spatial actions at each step. We introduce STARBench, a benchmark of spatiotemporal object search tasks across simulated and real environments. Experiments in STARBench and on a Tiago robot show that STAR consistently outperforms scene-graph and memory-only baselines, demonstrating the benefits of treating search in time and search in space as a unified problem. For more information: https://amrl.cs.utexas.edu/STAR.**

I. INTRODUCTION

We are interested in the problem of *open-world object retrieval*, where a service mobile robot is asked to retrieve arbitrary objects referred to through combinations of open-vocabulary appearance ("the red mug"), spatial properties ("the mug on the dining table"), and temporal properties ("the mug that was on the coffee table yesterday"). This problem comes with several challenges. First, since the robot does not know in advance which objects it will be asked to retrieve, it cannot explicitly track and remember all possible solutions. Second, even if it has observed the relevant object before, that object may no longer be at its most recently observed location. Third, retrieving an object may require the robot to interact with the environment, such as opening drawers or cabinet doors.

We note that solving this problem requires reasoning jointly about *space* (where the object is, how to reach it, what to manipulate to access it) and *time* (when it was last observed, how often it appears there, and whether it may have moved). Crucially, the robot must do so over an open set of objects, natural language referring expressions, and arbitrary temporal dynamics. Existing approaches for this problem focus either on searching in space or in time, but rarely both. Approaches to searching in space include Object Maps [1], Scene Graphs [2], and most recently Open-Vocabulary Scene Graphs [3]. Approaches to searching in time include video retrieval and grounded temporal reasoning methods [4]. A few recent approaches explore both space and time, such as dynamic scene graphs [5], [6], which however assume a closed set of objects and fixed relational vocabularies. To date, no prior work enables a robot to retrieve arbitrary, open-world objects described in natural language with both spatial and temporal references.

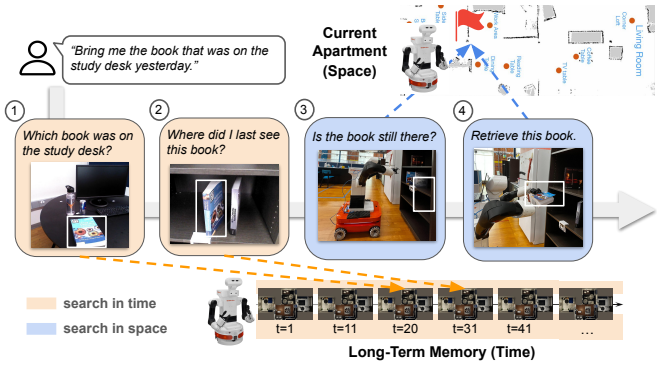

Fig. 1: A fundamental skill for a robot is to retrieve desired objects, given a combination of spatial and temporal references (e.g., *"bring the book that was on the desk yesterday"*). STAR is a framework that integrates long-term memory with a VLM-based decision-making module, enabling the robot to decide whether to search in time (recall past observations) or in space (probe the current environment), and to produce actions that gather the information needed and retrieve the correct object.

In this paper, we introduce STAR (SpatioTemporal Active Retrieval), a unified approach that integrates spatial and temporal search for open-world object retrieval. Our key insight is to treat memory queries (searching in *time*) and physical actions (searching in *space*) as elements of a unified action space inside an active interaction decision-making loop. This design combines the strengths of prior approaches: as in retrieval-augmented agents, it can leverage memory of past observations; and as in reactive agents, it can act in the current environment to gather new perceptual evidence and execute navigation or manipulation actions. To reason about past observations, STAR maintains a *long-term memory* of its observations in the world represented as a lightweight sparse language-captioned set of observations inspired by ReMEmbR [7], coupled with periodic full visual snapshots

[1]The authors are with the Department of Computer Science, The University of Texas at Austin, Austin, TX, USA. {taijingchen, sateesh, jh.xu, pavlakos, joydeepb, robertomm}@utexas.edu

S. Kumar is funded by the Amazon AI PhD fellowship.

This work was supported by the Amazon Research Awards (ARA) program. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

of the robot's observation. When given a new object retrieval task, STAR uses a large language model (LLM) in a decision-making loop to decide which spatial or temporal actions to execute, compiling results in a *working memory* that is used to decide on successive actions until it ascertains that it has retrieved the object that the user requested.

Evaluating such a capability requires benchmarks that capture long-term, dynamic environments. Since existing benchmarks for object search either assume static scenes or focus on passive reasoning over recorded videos for Visual Q&A, we developed STARBench, a benchmark for spatiotemporal object search in dynamic households. STARBench spans 360 tasks across visible, interactive, and commonsense settings and five instruction families. STAR consistently outperforms baselines across all task types and transfers to a Tiago robot in a mock apartment, with pronounced advantages on tasks requiring reasoning about object properties and references to past observations.

Our contributions are threefold:

- We propose STAR, a framework that unifies search in *time* (memory retrieval) and search in *space* (embodied actions) for open-world object retrieval.
- We introduce STARBench, a benchmark for spatiotemporal object search in dynamic household environments.
- We show STAR outperforms scene-graph and temporal-search-only baselines across all task types in STARBench, and demonstrates successful transfer to a Tiago robot in a mock apartment.

II. Related Work

STAR tackles embodied object search by unifying interactions with dynamic memory in a framework that searches in both space and time. We discuss connections to prior work in object search, benchmarking, LLM-based agents, and memory in robotics.

Object Search. Methods for object search aim to find a desired object through navigation and/or manipulation. In *object navigation*, robots move through the environment to detect a visible target object. While early works relied on geometric reasoning [8], more recent approaches use semantic representations such as scene graphs [9]–[11] or learned priors [12]–[15]. To deal with the continuously changing nature of human-populated environments, recent approaches integrate a dynamically updating representation of predefined or open-vocabulary objects [5], [6], [16]–[18]. However, such methods assume the critical information is only in the current state; they cannot deal with retrospective queries such as "bring me the book that was here yesterday." Analogously, in *interactive object search* (i.e., Mechanical Search [19]) robots exploit manipulation actions —opening cabinets or drawers, or removing occluding objects— to find the target object [19]–[27], making decisions with either a direct sensorimotor policy or a scene representation restricted to a predefined vocabulary of objects and updated to reflect only the current state. While STAR also uses interactions to search, it leverages an open-vocabulary memory representation with history that enables it to combine interactions (i.e., interactive searches in space) with searches in time.

Benchmarking Object Search. Early benchmarks to evaluate object search methods originated in the computer vision community under the umbrella of EmbodiedQA [28] and VisualQA [29], [30]. They evaluate embodied agents on broad tasks that combine navigation, perception, and language understanding, with object search included as one of the components. Instruction-following suites like ALFRED [31] extend this to household tasks requiring both navigation and manipulation. More recent work targets object navigation more directly on a predefined set of objects [32] or extended to open-vocabulary settings [33], [34]. While these benchmarks advance object search, they only evaluate search tasks based on the current state of a static environment; no existing benchmark includes tasks that require the agent to reason about previous states of a dynamically changing world to retrieve the correct item, a common situation in assistive robotics. *STARBench* fills this gap with a benchmark for spatiotemporal interactive object search in dynamic environments.

LLM Agents for Robotics. The impressive advances in the last years in large language and vision–language models (LLM/VLM) have enabled multiple applications of their common-sense reasoning as planners and controllers in robotics [35]–[42]. As pioneers, *Code as Policies* [36] generates executable robot programs directly from natural language, leveraging the structure of code to integrate perception and control primitives while supporting iterative refinement through execution feedback, while LM-Nav [37] combines large-scale language grounding with pretrained visual recognition and navigation modules, parsing instructions into semantic waypoints and re-planning as needed based on perceptual feedback. These systems illustrate how LLM/VLM can enable plan–execute–check loops by coupling reasoning with perception and control, yet they largely operate over the current scene and short contexts. In contrast, STAR leverages an LLM to generate code as memory queries and embodied actions within the same decision loop, enabling agents to reason jointly over past and present conditions.

Memory and Scene Representations for Robot Navigation. Robotics often relies on memory to represent objects, places, and their relations. Parametric structures such as scene graphs and their open-vocabulary or dynamic extensions [3], [43], [44] capture the current world state but overwrite past information. A parallel line of work in long-term semantic mapping develops persistent object-level maps that are updated across sessions to improve robustness in navigation [45]–[51], yet these maps also emphasize the current belief rather than retaining accessible histories of past states. Due to the complexity of building and maintaining these explicit representations, recent non-parametric approaches store instead raw observations for retrieval, as in ReMEmbR [7] or Embodied-RAG [52]. While these systems demonstrate the value of retrieval, memory typically remains an external module consulted before action. Inspired by the flexibility to handle open-vocabulary queries, STAR

integrates a non-parametric memory into the action space itself, enabling agents to recall earlier states, use them even when they differ from the present, and adapt behavior when discrepancies arise.

III. Problem Statement

In this section, we describe the problem of open-world object retrieval in dynamic, partially observable environments. Let $t \in \{0, 1, 2, \ldots\}$ denote the discrete timestep and $e_t \in \mathcal{E}$ the environment state at t, e.g., a configuration of an apartment, which may change over time due to exogenous factors, such as human activities. At each step, the available information $s_t = (t, o_t, x_t)$ to the robot consists of the current timestep t and a sensory observation $o_t \in \mathcal{O}$ of e_t at the robot pose x_t.

At task time $t = T$, the robot receives an open-ended natural-language instruction $\ell \in \mathcal{L}$ from the user, specifying a target object to be retrieved. The instruction may describe the target via attributes, spatial relations to other objects, or temporal cues. As a result, the space of instructions $\mathcal{L}$ is vast and diverse. Some instructions contain pointed temporal reference ("yesterday"), others have habitual patterns ("usually on the desk"), and some depend on objects' attributes and spatial context ("the blue mug in the kitchen"). We assume that before the task time, the robot patrols the environment and receives observations $S_{\leq T} = \{s_i\}_{i=0}^{T}$. To complete the task, the robot needs to maintain $S_{\leq T}$ to understand which object ℓ refers to. Since observations are high-dimensional and histories may span days, the robot must *store and index* past observations into queryable representations for efficient retrieval at task time. We formalize this process with a memory construction operation: $M = \Phi(S_{\leq T})$ where Φ maps the history of observations into a long-term memory M. Different works realize this construction differently, such as structured scene graphs [2] or semantic mapping [45]–[51]. Importantly, since the variations of the user instructions are enormous and are unknown before the task time, memory must be constructed in a task-agnostic way.

After receiving ℓ at time T, the robot executes for at most K steps to retrieve the requested object. Let $S_{T:T+k} = \{s_i\}_{i=T}^{T+k}$ denote the observation stream accumulated since T, where $k \in \{1, \ldots, K\}$ denotes the timestep of the policy execution. For simplicity, we will use timestep k to represent timestep $T+k$ whenever the context is clear. At each step, the policy selects an action based on the instruction, observation stream, and the long-term memory:

$$a_k \sim \pi(a \mid \ell, M, S_{T:T+k}), \tag{1}$$

which moves the robot to a new pose and obtains a new observation. The robot's objective is to reach the instructed object before the K-step budget expires, preferably in fewer steps. Because the environment may have changed, i.e., e_{T+k} may differ from $(e_1, ..., e_T)$, long-term memory M may be stale. The robot needs to decide how much to rely on information stored in M versus how much new evidence to acquire: selectively recall from M to form hypotheses about likely object locations, and, when those expectations are not met, act to gather new observations to refine where to search next, so the robot reaches the goal within the K-step budget.

IV. STAR: SpatioTemporal Active Retrieval

A. System Overview

To solve the open-world retrieval problem, robot needs to search in time by recalling historical evidence in M to form hypotheses about the object's current states, and search in space by acting in the world to validate those hypotheses and acquire observations that guide where to search next.

To realize this capability, we present STAR shown in Fig. 2. It uses a vision-language model as the policy, which allows the robot to handle open-vocabulary object descriptions and understand spatial and temporal cues. To search over M, a naive approach would condition the entire M in the model's context, but this is impractical: M spans long histories that can exceed the context window, and instruction-irrelevant details can distract the policy. Accordingly, STAR is organized around two complementary memories: (i) an efficiently searchable long-term memory M that supports fast retrieval of task-relevant evidence, and (ii) a working memory H that only retains task-relevant information, initialized at task time T and updated online with evidence retrieved from M and observation streams $S_{T:T+k}$. As a result, the VLM policy needs only to act based on the information in the working memory and the instruction, focusing on task-relevant information:

$$a_k \sim \pi(a|\ell, H_k) \cong \pi(a|\ell, M, S_{T:T+k}), \tag{2}$$

where timestep k denotes the policy execution step. To decide when to search in time (query M) versus when to search in space (take embodied actions), we expose a unified action space to the VLM policy. At each step, the policy selects either a temporal action (retrieve from M) or a spatial action (navigate, observe, or manipulate). The resulting evidence is used to update the working memory.

In the following, we provide details on: (1) the design of the efficiently searchable long-term memory M in Section IV-B, (2) the process of updating the working memory H with temporal and spatial actions in Section IV-C, (3) and how the policy $\pi(a \mid \ell, H)$ uses the updated working memory to make decisions in Section IV-D.

B. Non-Parametric Long-term Memory

We now detail the design of the long-term memory. This memory stores past observations in a non-parametric form in a vector database, together with lightweight descriptors that make them searchable during task execution. Following prior works [7], [52], we maintain multiple vector indices that allow the agent to query memory along different modalities:

- *Semantic index:* Embeddings $\texttt{embed}(o_t)$, computed by captioning each observation and embedding its caption. It allows for text-based semantic queries.
- *Temporal index:* Timestamp vectors t, which support queries about what was observed around a given time.
- *Spatial index:* Robot poses x_t, which allow robots to find out visible objects around that location.

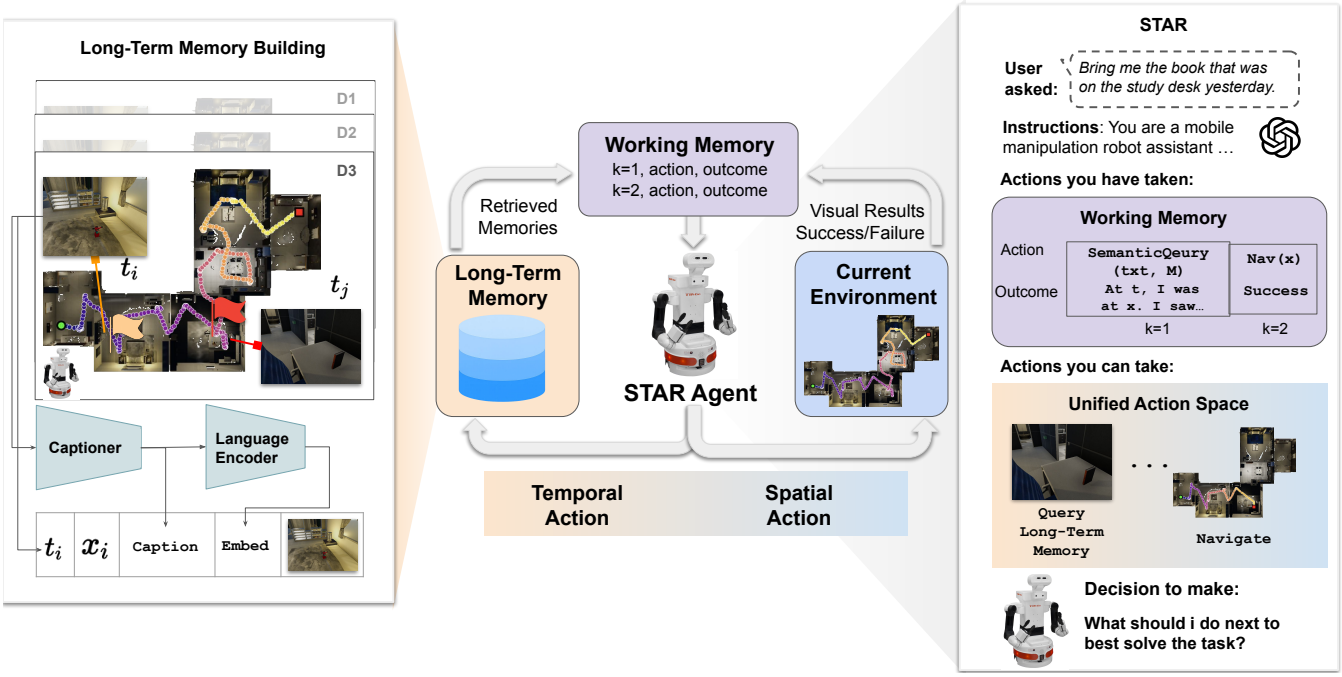

Fig. 2: STAR system. The robot patrols dynamic environments over multiple days to build a non-parametric long-term memory of past observations (left). When the user requests "bring me the book that was on the study desk yesterday", the agent initializes its working memory with the task. Guided by working memory, STAR chooses actions from a unified space: recalling past observations (search in time) or probing the current environment (search in space). Each outcome updates the working memory, and the loop continues until the robot successfully retrieves the target object.

This structure makes the memory naturally extensible: new modalities can be supported by adding indices from their vector representations without altering the storage structure.

Building on prior approaches, a key design choice in our memory is to also retain the raw visual observations o_t alongside their indices. If the agent determines that the embedding $\texttt{embed}(o_t)$ misses task-relevant details, it can revisit the original observation o_t to recover the necessary information. Each record at timestep t stores its temporal, spatial, semantic, and raw components in tuples: $m_t = (t, x_t, \texttt{embed}(o_t), o_t)$. We will next introduce how these memory records from M are selectively retrieved by the temporal action to update the working memory H.

C. Working Memory Update with Spatio-temporal Actions

The working memory H contains the *task-relevant information* that policy uses to make decisions. It is initialized at task time T and updated by (i) evidence selectively retrieved from the long-term memory M, and (ii) observations after taking a spatial action since T. Formally, the working memory at timestep k is an action-outcome trajectory $H_k = (a_{1:k-1}, y_{1:k-1})$, where a_k denotes the action taken at k, and y_k indicates the execution outcome of a_k. Each action is a tool-argument pair, where the tools are implemented as callable programs. Let $\mathcal{F} = \{f_1, ..., f_N\}$ denote the available tools and $\Theta = \{\Theta_1, ..., \Theta_N\}$ the tool-specific parameter spaces. An action at time k is $a_k = (f, \theta)_k$ with $f_i \in \mathcal{F}$ and $\theta_i \in \Theta_i$. We call $\mathcal{A} \coloneqq (\mathcal{F}, \Theta)$ the unified action space as the $\mathcal{F}$ contains both temporal and spatial search tools.

To update the working memory, the VLM policy examines H_k and selects the tool f and its argument θ. The action is then executed either in M or in the environment to retrieve information appended to H_k. In this work, we use three types of tools (temporal, spatial, or semantic) for the temporal actions. Their function arguments θ are text outputs from the VLM policy, which the tools convert to vector representations for retrieval via standard similarity metrics:

$$y_k = \{m_i : i \in I_r(\theta; M)\} \coloneqq f(\theta; M), \tag{3}$$

where f is the selected temporal action and $I_r(\theta; M) = \{i_1, \ldots, i_r\}$ are the indices of the top-r records in M nearest to the vector representation derived from θ under the chosen similarity metric. For example, `SemanticQuery` embeds the text argument θ and retrieves the top-r matching records from the semantic index in M using Eq. 3, grounding the temporal references, such as "the cup you saw yesterday".

Spatial actions allow the agent to *look and act* in the current environment. Tools f for spatial actions are the robot skills (e.g., navigation, grasping, detection), and arguments θ are skill parameters (target pose, grasping point). After the policy executes a spatial action, it receives feedback from the environment. Depending on the skills executed, the feedback can be raw sensor observations, structured detections (e.g., bounding boxes/masks with scores), or a task-level execution signal (e.g., success/failure). We denote the outcome as y_k. Effectively, the agent queries the real world for evidence: the outcome y_k verifies whether the current observation matches memory, refreshes knowledge with up-to-date observations, and tests action feasibility in a dynamic environment.

With the retrieved temporal and spatial outcomes, we update the working memory by appending these outcomes to it, along with the executed action, a_k:

$$H_{k+1} = H_k \oplus (a_k, y_k) = (a_{1:k}, y_{1:k}) \tag{4}$$

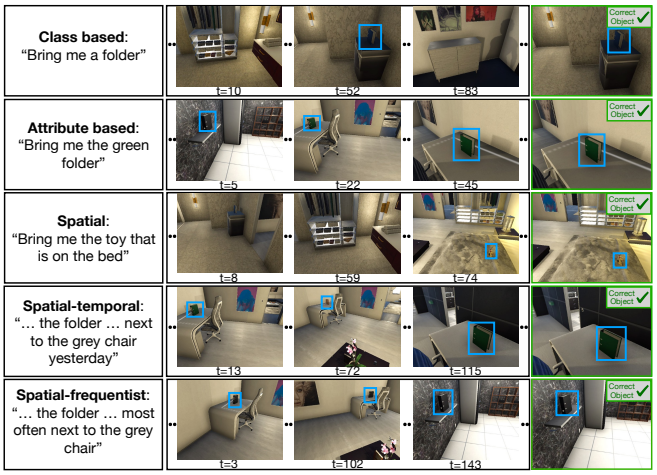

Fig. 3: **Task families in STARBench.** Each row shows one task family with its instruction (left), the agent's prior observations (middle), and the correct target object (right). *Class-based*: the folder seen at $t = 52$ is the target object. *Attribute-based*: despite a black folder at $t = 5$, the correct target is the green folder last seen at $t = 22$. *Spatial*: the toy observed on the bed at $t = 74$ is the target. *Spatial-temporal*: the folder seen by the grey chair at $t = 8$ later moved; its new position at $t = 115$ is the target. *Spatial-frequentist*: the black folder most often found next to the grey chair, but last observed at $t = 143$ in a new location is the target.

D. Action Selection with Working Memory

The goal of the VLM policy is to select the tool f and its argument θ from the unified action space to gather information and retrieve an object instructed in ℓ within a fixed budget K. To make the policy aware of the budget, we also provide the policy a *remaining budget* $R_k = (T+K) - k$ explicitly in its context so that planned actions respect the constraint (prefer less exploratory actions when R_k is small). By conditioning on the current working memory and the remaining budget, the policy $a_k \sim \pi(a \mid \ell, H_k, R_k)$ selects an action $a_k = (f, \theta)_k$, executes it, observes the outcome y_k, and updates the memory to H_{k+1} as in Eq. 4. The execution loop stops if the object is retrieved (completes the instruction) or when R_k=0 (budget exhausted).

V. STARBench: A Benchmark for Spatiotemporal Object Search

Existing object-search benchmarks largely assume static environments; we instead evaluate open-world retrieval in homes that evolve over time. We introduce *STARBench*, built on VirtualHome's apartment-style scenes and Unity3D simulator, which provides interactive objects and articulated receptacles across multiple furnished apartments [53]. VirtualHome offers several distinct layouts, enabling controlled scene changes across days.

In STARBench, each task is specified by a natural language instruction to retrieve a target object. Before task time, the agent patrols the environment, collecting egocentric observations, poses, along with captions of its visual observations that can later be used as memory.

To comprehensively evaluate agent ability, we introduce three task types. The first, *Visible Object Search*, tests whether the agent can resolve references when the target object is directly accessible. The second, *Interactive Object Search*, builds on the same object reference but requires embodied interaction, such as opening receptacles, to uncover hidden objects. The third, *Commonsense Object Search*, goes beyond memory recall: here the target object may never have been observed, and the agent must rely on commonsense reasoning, for example inferring that a book is likely to be on a desk. Since both visible and interactive search require reasoning over memory, we further structure them into five task families (Fig. 3): *(1) Class-based* (e.g., "find the book"), *(2) Attribute-based* (e.g., "find the red mug"), *(3) Spatial* (e.g., "find the mug on the table"), *(4) Spatial Temporal* (e.g., "find the book that was on the desk yesterday"), and *(5) Spatial Frequentist* (e.g., "find the mug that is usually by the sink"). Together, these task types and families span a spectrum from memory grounded reasoning, to embodied interaction, to generalization beyond observed experience.

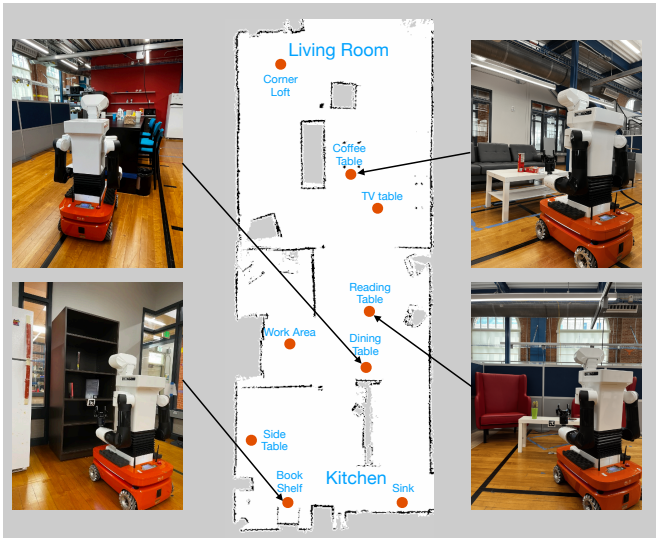

Fig. 4: **Mock Apartment for real-world evaluations.** We construct a mock apartment with a kitchen, a living room, and a study area.

To prepare each task in STARBench for execution at time T, the agent patrols the environment for 3–6 days, collecting 1200–1500 observations per day. Each observation $s_t = (t, o_t, x_t)$ records the timestep t, an egocentric camera observation o_t, and the agent pose x_t, along with the caption describing o_t. This sensor history $S_{\leq T} = \{s_i\}_{i=0}^{T}$ serves as input for constructing the long-term memory in STAR, as described in Sec. IV-B. We also provide scene graph representation of the environment for each task. Table I summarizes more statistics about the benchmark.

Task Type	Visible	Interactive	Commonsense
No. of scenes	3	3	3
No. of Tasks	225	90	45
No. of Object Classes	5	2	5
Task families	5	5	1

TABLE I: **Summary of STARBench.** We construct three task types with a total of 360 tasks: **Visible** (Visible Object Search), **Interactive** (Interactive Object Search) and **Commonsense** (Commonsense Object Search)

VI. Experimental Evaluation

Setup: In our experiments, we measure the performance of STAR on our benchmark STARBench and in the real

world. We report *execution success*, defined as whether the agent retrieves the user-requested object from the current environment within the step budget ($K = 20$). We use GPT-o3 [54], [55] as LLM backbone and query it at each step to select one action to search either in space or time. As discussed in Sec. IV-C, STAR's agent is equipped with tools to query past text and image observations stored in long-term memory (temporal retrieval, TR) for task-relevant information, and can also execute navigate, detect, pick, and open skills in the simulator (spatial search, S).

To separate perception errors from decision-making, we provide the agents with two modes of environmental knowledge when building long-term memory. In the *Oracle* environment mode, the agents are given ground-truth class labels of all objects: during long-term memory building, the captioner has ground-truth class labels of all visible objects, and scene graphs store ground-truth node labels and edge relationships; for the scene-graph agent, we additionally provide ground-truth data association for nodes. In the *Realistic* mode, no privileged information is given: agents rely entirely on predictions from state-of-the-art perception models to construct long-term memory. To isolate predicted perception errors, all agents use segmentation masks from the VirtualHome simulator for object detection during skill execution.

We evaluate five types of solutions: **Random** serves as a lower bound by navigating to random locations to search for the target object. **SG+S** and **SGAR** use ground-truth scene graphs (SG) of the environment over time in its working memory. **SG+S** makes a single attempt to retrieve the object (+S), while **SGAR** makes multiple attempts, actively interacting and searching for the objects (AR). **TR+S** uses our temporal retrieval design (TR) and then executes a one-shot plan (+S). **STAR** is our full approach, combining temporal retrieval with spatial search to retrieve the object.

To evaluate the performance of STAR in the real world, we deploy it on a Tiago robot in a mock household apartment environment (Fig. 4). We followed the STARBench procedure and created 9 tasks per type. A trial is considered successful if the robot leads the user to a correct landmark position where the target item can be found.

Experimental Results

In our experiments, we aim to answer a series of research questions:

(1) *Can STAR jointly reason over object attributes, space and time to identify and retrieve the correct object requested by users in dynamic environments?* Fig 5 reports execution success across 5 task types in visible object search. TR+S and SG+S each show complementary strengths: TR+S performs better on tasks requiring temporal recall, while SG+S retains an advantage on class-based and spatial reasoning. **STAR consistently outperforms the baselines on tasks requiring attribute, spatial, and temporal reasoning,** combining the benefits of temporal memory and spatial reasoning in a unified loop. This advantage is especially pronounced in attribute-based and spatiotemporal tasks, where reasoning

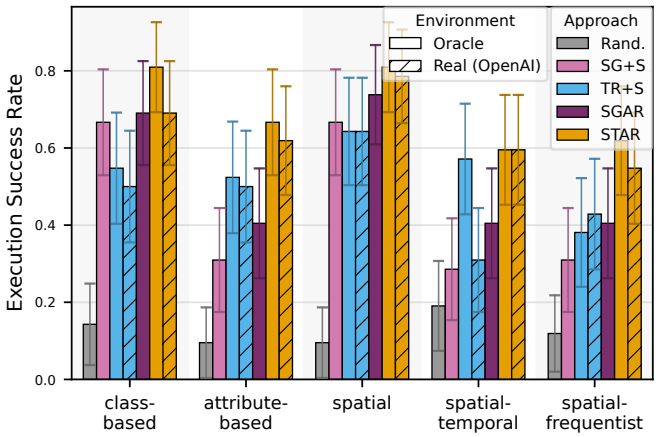

Fig. 5: Execution success rates across five task types of Visible Object Search tasks in STARBench (45 tasks per type). Bars indicate approaches; hatching denotes the environment knowledge used to construct long-term memory. *Oracle* builds memory with ground-truth object class labels; *Realistic* builds memory from model predictions only. SG+S uses full scene-graph history for a one-shot attempt; TR+S queries non-parametric memory for a one-shot attempt; SGAR augments SG+S with multiple retrieval attempts; STAR (ours) combines temporal retrieval with spatial search and achieves the highest success across task types.

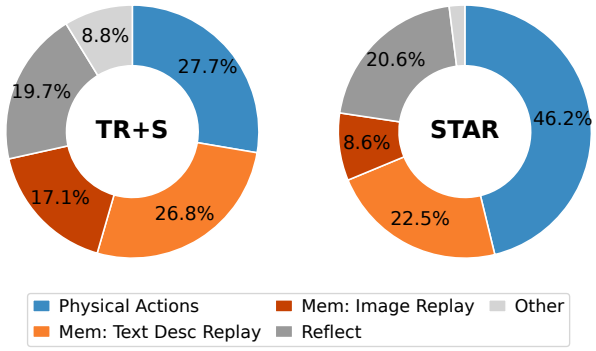

Fig. 6: Distribution of executed actions for all tasks.

about both object properties and past states is required. In addition, Fig. 6 shows that **actively gathering perceptual evidence when temporal recall is insufficient helps STAR retrieve objects in dynamic environments,** indicated by STAR's larger proportion of physical actions compared to TR+S, which reduces reliance on replaying stored captions and images. By navigating and detecting when memory is uncertain or potentially outdated, STAR updates stale information and acquires fresh perceptual evidence.

TABLE II: Execution success rates for interactive search tasks in STARBench. Task abbreviations: C = Class-based, A = Attribute-based, S = Spatial, ST = Spatial-Temporal, SF = Spatial-Frequentist.

Task	TR+S		STAR	
	Oracle	Realistic	Oracle	Realistic
C	0.56 ± 0.21	0.28 ± 0.19	0.67 ± 0.20	0.67 ± 0.20
A	0.33 ± 0.20	0.17 ± 0.17	0.50 ± 0.21	0.61 ± 0.21
S	0.22 ± 0.18	0.06 ± 0.12	0.50 ± 0.21	0.50 ± 0.21
ST	0.17 ± 0.17	0.28 ± 0.19	0.61 ± 0.21	0.28 ± 0.19
SF	0.28 ± 0.19	0.11 ± 0.15	0.61 ± 0.21	0.44 ± 0.21

(2) *Can STAR retrieve objects that are not directly observable by interacting with the environment (e.g., opening receptacles before picking)?* We observe that in interactive object search tasks, **temporal recall and visual context enable**

TABLE III: Average number of physical actions executed per successful run. "Optimal" indicates the ground-truth minimal #action required for success.

Visible Object Search					
Method	Env.	Percep.	Nav.	Manip.	Total
STAR	Oracle	1.80	1.62	1.52	4.93
	Realistic	1.70	1.60	1.44	4.74
Optimal	–	1	1	1	3
Interactive Object Search					
Method	Env.	Percep.	Nav.	Manip.	Total
STAR	Oracle	3.21	2.00	3.06	8.27
	Realistic	3.18	2.11	2.87	8.16
Optimal	–	2	1	2	5

STAR to open the correct receptacle and retrieve hidden objects, where other agents fail. Since these tasks require retrieving objects stored inside similar (a common situation in household environments) random execution fails on long action chains, and scene graphs struggle to disambiguate nearby identical receptacles. By leveraging temporal recall of how objects became visible and the surrounding visual context, STAR identifies the correct receptacle to open and successfully retrieves the target (Table II).

(3) *How efficiently does STAR make use of physical actions (navigation or interactions) to fulfill search tasks?* Table III shows that **STAR takes about 1.5–2× the minimal (oracle-based) number of physical actions to retrieve the objects.** Different from *computational actions* (reasoning, detecting objects) that can be optimized with increasing compute, physical actions are more difficult to accelerate and therefore, costly to perform; we will explore how to further optimize their use in future work.

TABLE IV: Execution success rates for common sense tasks.

Method	Oracle	Realistic
Random	0.10 ± 0.13	–
SG+S	0.10 ± 0.13	–
TR+S	0.19 ± 0.16	0.43 ± 0.19
STAR	0.57 ± 0.19	0.57 ± 0.19

(4) *Can STAR leverage common sense knowledge to locate objects never observed in memory?* When an object has never been observed, the only way to search efficiently in environments is to exploit common sense knowledge (e.g., *milk is probably in the kitchen*). As shown in Table IV, **STAR is able to find never observed objects resorting to common sense knowledge** by reasoning about likely locations.

(5) *Does the performance of STAR in STARBench extend to object search in the real world?* In our evaluation in the real world, we observe that **STAR repeatedly and robustly locate objects in the environment.** Table V shows that STAR surpasses TR on several task types and performs well across all, while random fails almost entirely.

TABLE V: Execution success rates for real-world deployment. (9 tasks per type; 54 tasks in total.)

Task Type	Random	TR+S	STAR
Class-based	0.00	0.44	0.67
Attribute-based	0.00	0.67	0.78
Spatial	0.11	1.00	1.00
Spatial-Temporal	0.00	0.56	0.56
Spatial-Frequentist	0.00	0.44	0.56
Common-Sense	0.00	0.33	0.33

VII. Discussion and Conclusions

We introduced STAR and STARBench, addressing the challenge of spatiotemporal object retrieval in dynamic environments. Our framework unifies memory queries and embodied actions, enabling agents to reason jointly over past and present states. To evaluate this setting, we presented STARBench, a benchmark that tests object search across visible, interactive, and commonsense tasks in evolving homes.

During our experiments, we observed that failures often stem from the VLM policy misusing tools or misreading tool outcomes, which limits its ability to refine subsequent actions. In particular, the policy sometimes expect a capability or feedback granularity that the low-level skills do not reliably provide, leading to misalignment between high-level decisions and skill execution. We are interested in reducing this policy–skill misalignment in future work.

Looking forward, we see two promising directions. One is *forgetting*: developing strategies to prune or compress memory in response to user actions while preserving what matters most. Another is *policy abstraction*, where high-level primitives are distilled from low-level action traces, in the spirit of code-as-policy [36], to better adapt to and explore skills for new user instructions.

References

[1] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *CVPR*, 2015, pp. 3668–3678.

[2] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *ICCV*, 2019, pp. 5664–5673.

[3] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," in *ICRA*, 2024, pp. 5021–5028.

[4] R. Arora, N. Narendranath, A. Tambi, S. S. Zachariah, S. Chakraborty, and R. Paul, "G^2tr: Generalized grounded temporal reasoning for robot instruction following by combining large pre-trained models," 2024. [Online]. Available: https://arxiv.org/abs/2410.07494

[5] Z. Yan, S. Li, Z. Wang, L. Wu, H. Wang, J. Zhu, L. Chen, and J. Liu, "Dynamic open-vocabulary 3d scene graphs for long-term language-guided mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 4252–4259, 2025.

[6] Q. P. M. Pham, K. T. N. Nguyen, L. C. Ngo, T. Do, D. Song, and T.-S. Hy, "Tesgnn: Temporal equivariant scene graph neural networks for efficient and robust multi-view 3d scene understanding," 2025. [Online]. Available: https://arxiv.org/abs/2411.10509

[7] A. Anwar, J. Welsh, J. Biswas, S. Pouya, and Y. Chang, "Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation," in *ICRA*, 2025, pp. 2838–2845.

[8] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and autonomous systems*, vol. 8, no. 1-2, pp. 47–63, 1991.

[9] G. Kumar, N. S. Shankar, H. Didwania, R. D. Roychoudhury, B. Bhowmick, and K. M. Krishna, "Gcexp: Goal-conditioned exploration for object goal navigation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 123–130.

[10] S. Amiri, K. Chandan, and S. Zhang, "Reasoning with scene graphs for robot planning under partial observability," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5560–5567, 2022.

[11] I. B. d. A. Santos and R. A. Romero, "A deep reinforcement learning approach with visual semantic navigation with memory for mobile robots in indoor home context," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 3, p. 40, 2022.

[12] M. Chang, A. Gupta, and S. Gupta, "Semantic visual navigation by watching youtube videos," *NeurIPS*, vol. 33, pp. 4283–4294, 2020.

[13] Y. Liang, B. Chen, and S. Song, "Sscnav: Confidence-aware semantic scene completion for visual semantic navigation," in *ICRA*, 2021.

[14] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning," in *CVPR*, 2019, pp. 6750–6759.

[15] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *arXiv preprint arXiv:1810.06543*, 2018.

[16] M. Chang, T. Gervet, M. Khanna, S. Yenamandra, D. Shah, S. Y. Min, K. Shah, C. Paxton, S. Gupta, D. Batra *et al.*, "Goat: Go to any thing," *arXiv preprint arXiv:2311.06430*, 2023.

[17] M. Lingelbach, C. Li, M. Hwang, A. Kurenkov, A. Lou, R. Martín-Martín, R. Zhang, L. Fei-Fei, and J. Wu, "Task-driven graph attention for hierarchical relational object navigation," *arXiv preprint arXiv:2306.13760*, 2023.

[18] A. Kurenkov, M. Lingelbach, T. Agarwal, E. Jin, C. Li, R. Zhang, L. Fei-Fei, J. Wu, S. Savarese, and R. Martın-Martın, "Modeling dynamic environments with scene graph memory," in *International Conference on Machine Learning*. PMLR, 2023, pp. 17 976–17 993.

[19] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg, "Mechanical search: Multi-step retrieval of a target object occluded by clutter," in *ICRA*, 2019, pp. 1614–1621.

[20] D. Honerkamp, M. Büchner, F. Despinoy, T. Welschehold, and A. Valada, "Language-grounded dynamic scene graphs for interactive object search with mobile manipulation," *IEEE Robotics and Automation Letters*, 2024.

[21] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, "Learning hierarchical interactive multi-object search for mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8549–8556, 2023.

[22] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," *arXiv preprint arXiv:2210.05714*, 2022.

[23] A. Kurenkov, R. Martín-Martín, J. Ichnowski, K. Goldberg, and S. Savarese, "Semantic and geometric modeling with neural message passing in 3d scene graphs for hierarchical mechanical search," in *ICRA*, 2021, pp. 11 227–11 233.

[24] A. Kurenkov, J. Taglic, R. Kulkarni, M. Dominguez-Kuhne, A. Garg, R. Martín-Martín, and S. Savarese, "Visuomotor mechanical search: Learning to retrieve target objects in clutter," in *IROS*, 2020.

[25] H. Huang, M. Dominguez-Kuhne, V. Satish, M. Danielczuk, K. Sanders, J. Ichnowski, A. Lee, A. Angelova, V. Vanhoucke, and K. Goldberg, "Mechanical search on shelves using lateral access x-ray," in *IROS*, 2021.

[26] H. Huang, M. Danielczuk, C. M. Kim, L. Fu, Z. Tam, J. Ichnowski, A. Angelova, B. Ichter, and K. Goldberg, "Mechanical search on shelves using a novel "bluction" tool," in *ICRA*, 2022, pp. 6158–6164.

[27] H. Huang, L. Fu, M. Danielczuk, C. M. Kim, Z. Tam, J. Ichnowski, A. Angelova, B. Ichter, and K. Goldberg, "Mechanical search on shelves with efficient stacking and destacking of objects," in *The International Symposium of Robotics Research*. Springer, 2022, pp. 205–221.

[28] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *CVPR*, 2018, pp. 1–10.

[29] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *ICCV*, 2015, pp. 2425–2433.

[30] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," in *CVPR*, 2018, pp. 4089–4098.

[31] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *CVPR*, 2020.

[32] R. Ramrakhya, E. Undersander, D. Batra, and A. Das, "Habitat-web: Learning embodied object-search strategies from human demonstrations at scale," in *CVPR*, 2022, pp. 5173–5183.

[33] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation," in *CVPR*, 2023, pp. 23 171–23 181.

[34] N. Yokoyama, R. Ramrakhya, A. Das, D. Batra, and S. Ha, "Hm3d-ovon: A dataset and benchmark for open-vocabulary object goal navigation," in *IROS*, 2024, pp. 5543–5550.

[35] 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*, 2022.

[36] 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," *arXiv preprint arXiv:2209.07753*, 2022.

[37] D. Shah, B. Osiński, S. Levine *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on robot learning*. PMLR, 2023, pp. 492–504.

[38] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[39] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[40] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *ICCV*, 2023, pp. 2998–3009.

[41] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," *arXiv preprint arXiv:2209.11302*, 2022.

[42] 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," in *ICRA*, 2025, pp. 13 337–13 345.

[43] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary queryable scene representations for real world planning," in *ICRA*, 2023.

[44] S. Koch, N. Vaskevicius, M. Colosi, P. Hermosilla, and T. Ropinski, "Open3dsg: Open-vocabulary 3d scene graphs from point clouds with queryable objects and open-set relationships," in *CVPR*, 2024, pp. 14 183–14 193.

[45] O. M. A. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *CVPR*, 2013, pp. 1352–1359.

[46] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *ICRA*, 2017.

[47] J. McCormac, R. Clark, M. Bloesch, A. J. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level SLAM," *arXiv preprint arXiv:1808.08378*, 2018.

[48] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "Panopticfusion: Online volumetric semantic mapping at the level of stuff and things," in *IROS*, 2019.

[49] A. Rosinol, M. Abate, Y. Tian, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic visual-inertial SLAM," in *ICRA*, 2020.

[50] T. B. Martins *et al.*, "OVO-SLAM: Open-vocabulary online simultaneous localization and mapping," *arXiv preprint*, 2024.

[51] A. Adkins *et al.*, "Obvi-SLAM: Long-term object-visual SLAM," *arXiv preprint arXiv:2309.15268*, 2023.

[52] Q. Xie, S. Y. Min, P. Ji, Y. Yang, T. Zhang, K. Xu, A. Bajaj, R. Salakhutdinov, M. Johnson-Roberson, and Y. Bisk, "Embodied-rag: General non-parametric embodied memory for retrieval and generation," 2025. [Online]. Available: https://arxiv.org/abs/2409.18313

[53] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *CVPR*, 2018, pp. 8494–8502.

[54] OpenAI, "Introducing openai o3 and o4-mini," https://openai.com/index/introducing-o3-and-o4-mini/, Apr. 2025, accessed 2025-09-15.

[55] "Openai o3 and o4-mini system card," https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf, OpenAI, Apr. 2025, accessed 2025-09-15.