Sari Sandbox: A Virtual Retail Store Environment for Embodied AI Agents

Janika Deborah Gajo[1], Gerarld Paul Merales[1], Jerome Escarcha[1],
Brenden Ashley Molina[1], Gian Nartea[1], Emmanuel G. Maminta[2],
Juan Carlos Roldan[2], Rowel O. Atienza[1,2]
[1]EEEI, University of the Philippines, Diliman, Quezon City
[2]AI Graduate Program, University of the Philippines, Diliman, Quezon City
jbgajo@up.edu.ph, gmmerales@up.edu.ph, jeescarcha@up.edu.ph
bqmolina@up.edu.ph, gdnartea@up.edu.ph, emmanuel.maminta@eee.upd.edu.ph
jtroldan@up.edu.ph, rowel@eee.upd.edu.ph

Figure 1. Overview of our virtual retail environment for embodied AI and human benchmarking. **Top row**: Photorealistic, high-fidelity 3D product models (left) and their randomized placement within semantically grouped categories (right). **Middle row**: Three store layouts and key features such as a functional checkout, dynamic labels (price, expiration), interactable products and elements, teleportation system, VR support, and a Python API for agents. **Bottom row**: Human participants in VR (left) perform shopping tasks (e.g., picking, inspecting, checkout, navigation), with optional tunneling vignette. An embodied agent (right) completes comparable tasks for benchmarking.

Abstract

*We present **Sari Sandbox**, a high-fidelity, photorealistic 3D retail store simulation for benchmarking embodied agents against human performance in shopping tasks. Addressing a gap in retail-specific sim environments for embodied agent training, Sari Sandbox features over 250 interactive grocery items across three store configurations, controlled via an API. It supports both virtual reality (VR) for human interaction and a vision language model (VLM)-powered embodied agent. We also introduce **SariBench**, a dataset of annotated human demonstrations across varied task difficulties. Our sandbox enables embodied agents to navigate, inspect, and manipulate retail items, providing baselines against human performance. We conclude with benchmarks, performance analysis, and recommendations for enhancing realism and scalability. The source code can be accessed via* `https://github.com/upeee/sari-sandbox-env`.

1. Introduction

The demand for understanding retail behaviors is rapidly evolving. Paolanti et al. [1] developed an RGB-D-based deep learning system for analyzing shopper movement, shelf interactions, and re-identification, allowing store-level behavioral insights. While such systems are valuable for observing real human behavior, there is a growing interest in studying these phenomena within photorealistic simulation environments that support automated agents. These environments enables us to systematically test hypotheses about product selection, navigation, and layout design—on-scale and under controlled conditions. For example, NVIDIA's Omniverse is a powerful platform used by popular retailers like Kroger [2] and Lowe's [3] to construct digital twins of their respective stores. Simulation environments are a great way to enable scalable, cost-effective and realistic evaluations of agents performing these complex tasks [4, 5].

Specialist embodied agents in retail settings are trained in these specialized simulation platforms. In Lowe's case, they created an interactive Omniverse replica of a home improvement store, where retail employees operate augmented reality (AR) headsets to overlay the digital twin in the physical store. Several similar ideas about robotics and digital twin systems proliferate in the Future Convenience Store Challenge (FCSC) at the World Robot Summit [6]. FCSC proposes tasks for the acquisition and placement of items in a custom retail store layout. To this end, the organizers of this event have even exposed Gazebo/ROS packages for the virtual store layout and CAD-based models for these tasks.

Indoor navigation simulators are mainly focused on domestic environments, which typically refer to household settings such as kitchens, bedrooms, and living rooms [7], as seen in Habitat [8], AI2Thor [9], Matterport3D [10] and ThreeDWorld [11]. Little work is done on embodied agent simulation for retail activities, product and shelf stocking, and grocery store layout. We note that the tasks exposed by the FCSC [12] include restocking of items and store renovation. In this work, we propose exploring sim environments modeling tasks such as more complex item selection for checkout and product comparison with a more diverse product set. To accomplish these, we introduce the following contributions.

- **Sari Sandbox** is a virtual retail environment for developing and evaluating embodied agents. Modeled after small convenience stores, it features interactive 3D layouts, photorealistic rendering of 250 diverse products, and three distinct store layouts based on surveys of real local configurations.
- We provide a Python API for agent control and data collection within the Sari Sandbox environment.
- We publish **SariBench**, a benchmark of retail tasks performed in the Sari Sandbox environment plus human demonstrations of these tasks using VR interaction with the Sari Sandbox, as a baseline for testing embodied agent performance on the aforementioned tasks.
- Finally, we design an agentic AI architecture to address the easy tasks in the SariBench. This embodied agent leverages off-the-shelf proprietary and open-source models, employing a straightforward architecture and toolset without requiring any fine-tuning. The discussion can be found in the Supplementary Material.

The next sections cover related virtual store simulators and agent designs, detail the Sari Sandbox environment's design and benchmarks, present performance profiling and comparisons between humans and agents, and conclude with recommendations for future work. An overview of the environment is shown in Figure 1.

2. Review of related work

2.1. Embodied retail store simulators

Table 1. Comparison of visual navigation datasets using RGB sensor data. Notably, these benchmarks primarily focus on domains other than retail, highlighting a research gap (Room-to-Room: household navigation; ION & ALFRED: household tasks; HumanoidBench: humanoid locomotion and manipulation).

Benchmark dataset	Size	Simulator
Room-to-Room [13]	90 scenes	Matterport3D [10]
ION [14]	600 scenes	AI2-THOR [9]
HumanoidBench [15]	27 tasks	MuJoCo [16]
ALFRED [17]	120 scenes	AI2-THOR [9]

Embodied agent simulators provide an avenue for AI to be trained at low cost and with minimal risks, while still allowing for complex, interactive learning experiences. The primary goal of our work is to establish an environment that exposes high-fidelity, physically accurate, and photorealistic interactions. Several simulators like Meta's Habitat-Sim [8], Matterport3D-Simulator [10], Isaac Sim [18] powered by NVIDIA Omniverse [19], iGibson [20, 21], and AI2Thor [9] have been widely used to build benchmarks for visual exploration and diverse embodied tasks in virtual space. As detailed in Table 1, these prominent benchmark datasets leverage such simulators to focus on contexts like household navigation, general manipulation, or locomotion. A survey on sim environments [7] outlines how such environments focusing on goal-driven navigation have slowly progressed from basic navigation to tackling complex interactions, predominantly in these non-retail settings. Another survey on embodied agents [22] evaluated simulators using a set of features that serve as robust evaluation criteria, including environment type, object complexity, physics fidelity, and interactivity. We adopted these criteria as a basis for Sari Sandbox's design, explicitly addressing the under-

explored domain of retail scenarios.

As outlined also from that survey [22], there are three research tasks that provide a foundation for embodied agent adaptation in unfamiliar environments. These are: visual exploration, wherein the embodied agent gathers data about its environment through motion and perception; visual navigation refers to embodied agent navigation usually to achieve a goal; and embodied question answering, wherein the embodied agent would be required to navigate and answer questions. We expect to establish entry points from our research into the three tasks laid above.

While existing embodied agent simulators are increasingly sophisticated, they are not tailored for the operational needs of retail. The prevalent focus on household or industrial settings limits their applicability for retail-specific tasks like inventory stocking, product recognition, or dynamic shelf management. To address this gap, we introduce Sari Sandbox, an environment designed specifically for developing and evaluating embodied agents in a retail context.

2.2. Embodied agents in virtual simulation

Aligning linguistic input with agent capabilities is crucial for grounding language in embodied simulations. However, a persistent challenge is bridging the gap between a language model's semantic output and the spatial precision needed for navigation. Geometric maps inherently offer more structure for this than language-only observations, as highlighted by Huang *et al.* (2023) [23]. Our work, similar to LM-Nav [24], explores language-guided navigation using off-the-shelf models. While LM-Nav targets physical agents and our research focuses on agentic patterns in virtual environments, both approaches grapple with translating high-level language into precise, executable actions. What distinguishes our work is the incorporation of visual inputs alongside language.

The development of LLM-powered embodied agents has shown immense potential in sandbox environments, leveraging advanced reasoning and language abilities for open-ended tasks [25]. Projects such as NVIDIA's VOYAGER [26] and the proposed EMBODIED AGENT INTERFACE [27] provide further validation of their effectiveness. A common framework adopted for such agents is ReAct (Reason and Act) [28], whose core loop (*thought* → *action* → *observation* → *refinement*) effectively emulates human problem-solving and is highly suitable for embodied tasks. The adaptability of ReAct has been demonstrated in other work, such as STATEACT, which extends it for the ALFWORLD household environment [29, 30]. A critical challenge for these LLM- or VLM-powered embodied agents is their stateless nature, where each inference is an isolated event, unlike human cognition which constantly integrates memory. To overcome this, various approaches for explicit memory modeling have been proposed, including cognitive architectures like CoALA [31], which models procedural, working, semantic, and episodic memory components. Furthermore, memory writing operations [32] are commonly used to manage and utilize these memories, often involving techniques like creating concise summaries of immediate surroundings [33] or informative recollections [34] for grounding the agent in its environment.

3. Design

Sari Sandbox is a 3D retail environment designed for evaluating embodied agents in retail tasks. It features an API-controlled avatar, three store layouts, 250 grocery products, and a self-checkout system. Each product is annotated with rich ground truth data which includes category, name, price, net weight, ingredients, nutritional facts, allergens, and manufacturing origin that could enable precise evaluation and data-driven experimentation. The environment is performance-optimized without sacrificing texture fidelity, built using Unity's Universal Render Pipeline (URP) for its balance of rendering quality, efficiency, and decal support (crucial for dynamic on-product labels). Real-time physics and interaction are handled via Unity's default 3D engine, NVIDIA PhysX.

3.1. Design requirements

In order to create the environment, the following design requirements were taken into consideration:

- **Environment performance**. The environment should perform at 60 frames-per-second well without compromising its visual quality.
- **Diverse products**. The product diversity in the environment should closely resemble that of its real-world counterpart. The textures used should be high-fidelity to allow humans and AI to read the smallest text.
- **API**. Users and/or embodied agents should be able to control the avatar and the environment. In addition, they shall also receive environment data such as avatar position, avatar rotation, hand grip state, store layout, and randomization seed.
- **VR capabilities**. The environment should be connected and optimized for VR capabilities for human benchmarking with minimal user fatigue and nausea.

3.2. Models

A total of 250 3D products were created for the environment. The products are based on real-life packaged goods typically found in a convenience store. This number was determined through an initial survey of several convenience stores, from which we compiled a representative and diverse set of food products commonly encountered by consumers. The selection was further limited to food products with packaging that could be flattened for scanning using

(A) (B)

Figure 2. Expiration decal (A) projects the generated date onto the product surface, while the barcode plane (B) is positioned above for scanner detection.

a flatbed scanner. We then used open-source tools, namely GIMP for preprocessing and Blender for 3D modeling.

Product meshes are classified into three types: simple, complex, and deformable. Simple products (e.g., boxes) use basic primitives; complex products (e.g., bottles) combine multiple primitives; and deformable items (e.g., chip bags, juice packs) require detailed modeling due to their non-rigid structure. All products are simulated as rigid bodies. Additionally, they are categorized into 11 food types (Table 2).

Table 2. Product count per category, with a total of 250 products across 11 categories.

Water	12	**Soda**	23	**Juice**	16	**Dairy**	20
Biscuit	50	**Can**	59	**Chips**	40	**Nuts**	15
Soup	6	**Noodles**	7	**Liquor**	2		

As Figure 2 shows, barcodes and expiration date stamps are integrated into Unity prefabs. These elements activate dynamically when the avatar grabs a product and hide otherwise, optimizing rendering and simulating realistic handling—much like how shoppers inspect packages for dates. Instead of scanning the barcode texture directly, a simulated object detection method using ray casting is employed. Rays target specific barcode planes in front of the actual barcode surface; the plane's rotation confirms correct scanning orientation. To keep expiration dates current and adaptable, each product's date is randomly generated at runtime. It is then rendered using Unity's URP decal system, which allows it to conform to various mesh shapes and update procedurally at the start of each session. Price tags, displayed along the front edge of each shelf (Figure 3), dynamically update to reflect the randomly placed products. While placements vary, the system groups similar product types, mirroring real-world grocery arrangements, with aligned price tags.

To enhance space and time efficiency, 3D models underwent optimization. Due to the scene's numerous objects, only box colliders are used to minimize physics computation time. High-resolution image textures, a common source of excessive space consumption, were addressed by using JPG format. We also implemented Level of Detail (LOD), a standard game development technique where objects render with varying detail based on camera distance (Figure 4). These optimized models are then arranged within the environment to simulate real-world grocery store product displays.

Figure 3. Price tags displayed on the front edge of the shelf.

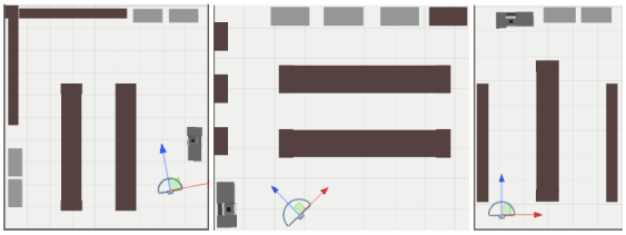
(A) (B)

Figure 4. Level of Detail (LOD) comparison between high (A) versus low (B) quality.

3.3. Environment

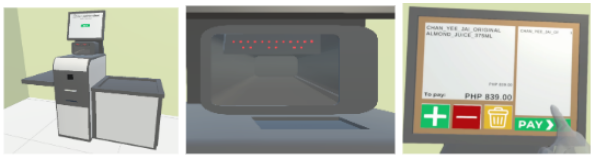
Several real-world small-scale retail stores were surveyed to inform the design of the virtual environment, focusing on store size, arrangement, and layout. From these observations, three distinct store layouts were recreated in the environment, each featuring shelves and a self-checkout counter. The corresponding top-down view for each store layout is shown in Figure 5. Shelves are equipped with overhead labels and are assigned either single or multiple product categories, each with corresponding price tags (see Figure 1). In addition to standard shelving, hinge door cabinets and sliding door cabinets were implemented to simulate real-world refrigeration units.

The self-checkout counter as shown in Figure 6 incorporates a touchscreen interface operable using the tip of the avatar's index finger. This interface allows users to view, add, and remove products from their virtual shopping cart. Dedicated buttons initiate the checkout process or modify scanned items. A fixed barcode scanner on the counter simulates barcode reading by casting a ray toward the barcode plane of a product, which is only successfully registered if the orientation aligns correctly with the scanner.



(A) (B) (C)

Figure 5. The three store layouts: Store 1 (A), Store 2 (B), and Store 3 (C), with the avatar's starting position indicated.



(A) (B) (C)

Figure 6. Depicted are the self-checkout counter (A), barcode scanner (B), and touchscreen (C).

To maintain performance in real-time rendering, frustum culling and occlusion culling were implemented. These techniques ensure that only visible objects within the camera's view are rendered (see Figure 7), significantly improving runtime efficiency in densely packed environments.

(A) (B)

Figure 7. Frustum and occlusion culling visualization: before (A) and after (B) application.

To enhance the immersive experience and mitigate common usability and comfort issues in VR, we implemented several features based on established interaction principles [35, 36]. These support naturalistic interaction, user comfort, and consistency across experimental setups. Specifically, we developed a **hand interaction system** for grabbing, touching, placing, and throwing objects. We also incorporated **haptic feedback** that triggers when users hover over interactable objects, reinforcing interaction and presence. To reduce cybersickness, a **tunneling vignette** limits peripheral vision during movement. Finally, a **teleportation system** with a parabolic reticle was included to support spatial navigation while addressing physical room-scale and tracking limitations.

3.4. Avatar

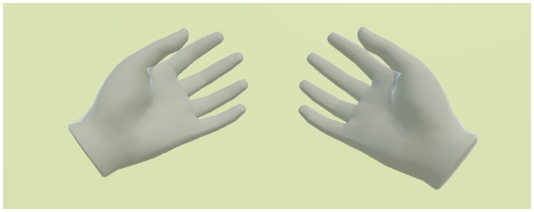


Figure 8. Hand models for the avatar.

The user's in-environment representation is an avatar, comprising a camera and two hands within a character controller. For avatar interaction, we adapted the hand models shown in Figure 8. This configuration supports interaction via either VR hardware or API-driven simulation. If no VR headset is detected at startup, the camera is positioned at an approximate adult human eye height of 1.6 meters, with hands offset 0.5 meters downward to reflect a natural standing posture.

3.5. SariBench

Since this environment is specifically designed for retail store tasks and to the best of our knowledge, there is no existing studies have established a baseline for such scenarios, we propose a set of tasks to serve as a benchmark for evaluating embodied agent performance. These tasks and their corresponding levels are detailed in Table 3.

Table 3. **SariBench** tasks: Baseline tasks of varying difficulty along with the skills involved to execute them.

Difficulty	Skills involved	Example
Easy	Perception, Navigation, Manipulation	Find and pick up a box of **cereal**.
Average	Perception, Navigation, Manipulation, Memory, Task Execution	Pick up a bottle of **soda** and scan at checkout.
Difficult	Perception, Navigation, Manipulation, Memory, Task Execution, Decision Making, Comprehension	Which of these two products has lower sugar content: **strawberry-flavored biscuit** or **chocolate-flavored biscuit**? Scan the answer.

To create the SariBench dataset, we adapted the environment for VR headsets and recruited volunteers to perform

retail tasks. The dataset currently comprises 100 videos of participants completing these tasks. Besides screen recordings, we captured the following environmental data at 10 frames per second: **global head position and rotation**, **global hand position and rotation**, **grip state**, and **hovered or held item**. Twenty human participants tested the environment, completing tasks selected from a curated pool categorized by difficulty. Each participant received a briefing on the data collection's purpose, project background, and specific data types collected. Participants had 15 minutes to familiarize themselves with VR controls in a playground environment shown in Figure 9 to ensure that they could navigate, manipulate objects, and feel comfortable in the virtual space, following established best practices for VR immersion [35]. Once acclimated, each participant was assigned three easy, two average, and two hard tasks. During task execution, participants were encouraged to verbalize their thought processes, which were recorded and later transcribed. Figure 10 shows several participants using the VR system to complete tasks for dataset collection.

Figure 9. The playground environment used to introduce the participants to the VR controls.

Figure 10. Volunteers testing the Sari Sandbox.

3.6. API

The avatar and environment are controlled via a direct Python API, designed for seamless integration with an embodied agent. This API comprises a Python-based client that sends JSON files containing functions for avatar control, environment data retrieval, and simulation resets. These JSON files are received and parsed by a C# (Unity's scripting language) server, which then executes the corresponding actions within the Unity environment. Communication between the client and server is facilitated through a WebSocket server, as illustrated in Figure 11. The API exposes three primary function types: *Agent Actions*, *Information Gathering*, and *Store Manipulation*. Table 4 lists all available functions; translation (T) and rotation (R) parameters accept 3D vector inputs.

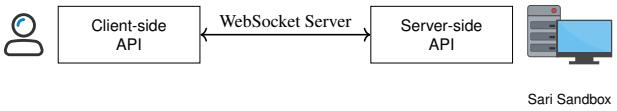


Figure 11. Communication flow between client-side and server-side APIs via WebSocket.

Table 4. Agent control API functions with full arguments. **T**: translation. **R**: rotation.

Function	Description
`TransformAgent(T, R)`	Manipulates the agent's body or camera.
`TransformHands(leftT, leftR, rightT, rightR)`	Transforms the left and right hands.
`ToggleLeftGrip()`	Toggles the left hand grip to grab objects.
`ToggleRightGrip()`	Toggles the right hand grip to grab objects.
`ToggleLeftPoke()`	Toggles the left hand's poke animation.
`ToggleRightPoke()`	Toggles the right hand's poke animation.
`RequestScreenshot()`	Captures the current camera view.
`Reset()`	Resets the environment to its initial state.

4. Experiments and analysis

The environment was developed and executed on a desktop equipped with an Intel i7-8700 3.2GHz CPU, NVIDIA GTX 1080 GPU, and 64 GB of RAM. A Meta Quest 2 VR headset was used for dataset collection. All results and analyses presented in this paper were computed based on these hardware specifications. For the embodied agent evaluation, we used a paid API version of Gemini 2.5 Pro (`gemini-2.5-pro-preview-05-06`) with reasoning budget of 2048 tokens. The discussion about the design and development process of the basic embodied agent is found in the Supplementary Material.

4.1. Performance metrics

Using the Unity Profiler, we measured the frame processing time over 300 frames. Average frames-per-second were 26.73 (Layout 1), 23.23 (Layout), and 35.14 (Layout 3). Layout 3 exhibited the highest performance, followed by Layout 1, then Layout 2. This performance variation stems from differing store sizes, which impact the number of objects and, consequently, physics calculations during object instantiation at startup, causing a processing time spike.

4.2. Texture fidelity

To assess texture fidelity, we utilized PaddleOCR [37], an open-source model recognized for its robust performance in multilingual and complex text recognition [38, 39]. We evaluated PaddleOCR on product label text using precision, recall, and character error rate (CER), achieving high accuracy due to the structured layout: Precision: 0.986, Recall: 0.943, CER: 0.014. However, OCR performance typically degrades with rotated text, especially when vertical and horizontal text coexist (see Supplementary Material). These issues can be mitigated by image rotation or object manipulation. The model also struggles with stylized text, such as brand logos or decorative fonts. Therefore, embodied agents' reading pipelines must account for these limitations.

4.3. Human versus embodied agent evaluation

We assigned 108 tasks, across three difficulty levels and randomized store layouts, to human participants, recording their completion time and success rate. Table 5 shows easy tasks were significantly faster, as they typically involved only one item, while more items increased navigation time. Surprisingly, average tasks took longer than difficult ones, mainly due to participants' unfamiliarity with the barcode scanner and because not all difficult tasks required checkout.

Table 5. Performance evaluation of human versus embodied agent on the SariBench tasks based on average time to complete and completion rate. Embodied agent evaluation is limited to easy tasks. **L1/L2/L3**: Layout 1, Layout 2, Layout 3. **HAT**: Human average time in seconds. **AAT**: Embodied agent average time. **HCR**: Human completion rate. **ACR**: Embodied agent completion rate.

Difficulty	**HAT** ↓	**HCR%** ↑	**AAT** ↓	**ACR%** ↑
Easy-L1	47	88.88	780	68.63
Easy-L2	73	100.00	660	45.10
Easy-L3	61	93.33	420	33.33
Average-L1	158	87.50	-	-
Average-L2	106	100.00	-	-
Average-L3	84	100.00	-	-
Difficult-L1	76	100.00	-	-
Difficult-L2	136	100.00	-	-
Difficult-L3	113	100.00	-	-

Our findings highlight a critical disparity in easy task performance: humans consistently outperformed the embodied agent in efficiency and effectiveness. We evaluated the embodied agent's end-to-end task success by manual visual inspection: success was task completion within 45 minutes; failure was exceeding this limit or if it enters "mode collapse" (i.e., getting lost). The embodied agent's completion times were up to 16 times longer than humans', with success rates under 70% compared to human rates often near 100%. This substantial proficiency gap is partly due to the VLM's inherent computational overhead, where text generation and inference time significantly prolong the embodied agent's task completion. Despite fully utilizing the designed APIs, the embodied agent's performance fell short, indicating challenges with the VLM's overall optimal reasoning and decision-making for these embodied tasks. The embodied agent's Easy-L1 to L3 performance is a solvability proof-of-concept for Sari Sandbox, not an optimized benchmark. Harder task evaluations are future work, as our current focus is the sandbox itself. While humans generally excelled, their performance was nuanced by factors like perseverance and occasional carelessness, explaining slight dips in completion rates (e.g., Easy-L1 at 88.88%). Notably, reported motion sickness among participants is a key consideration; though not directly impacting metrics, this VR response could affect user experience and engagement. Future research should explore mitigation strategies to enhance comfort and data reliability.

4.4. Participant thought process flowcharts

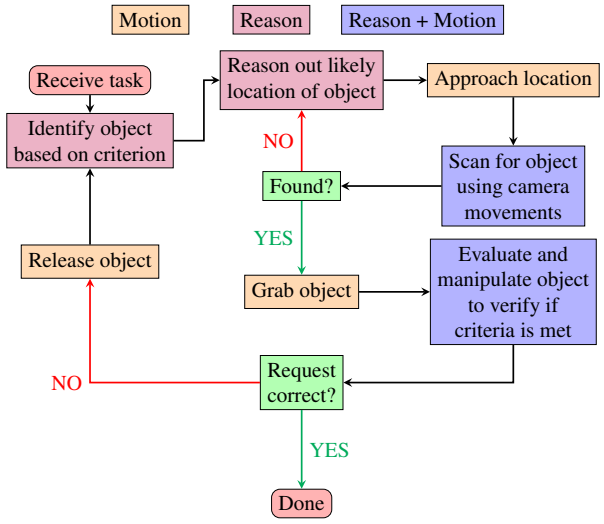


Figure 12. General thought process for easy tasks.

We include flowcharts illustrating the thought processes of participants during object retrieval and checkout tasks, derived from think-aloud protocols. The Easy Task Flowchart (Figure 12, e.g., *"Find and pick up a soda"*) in-

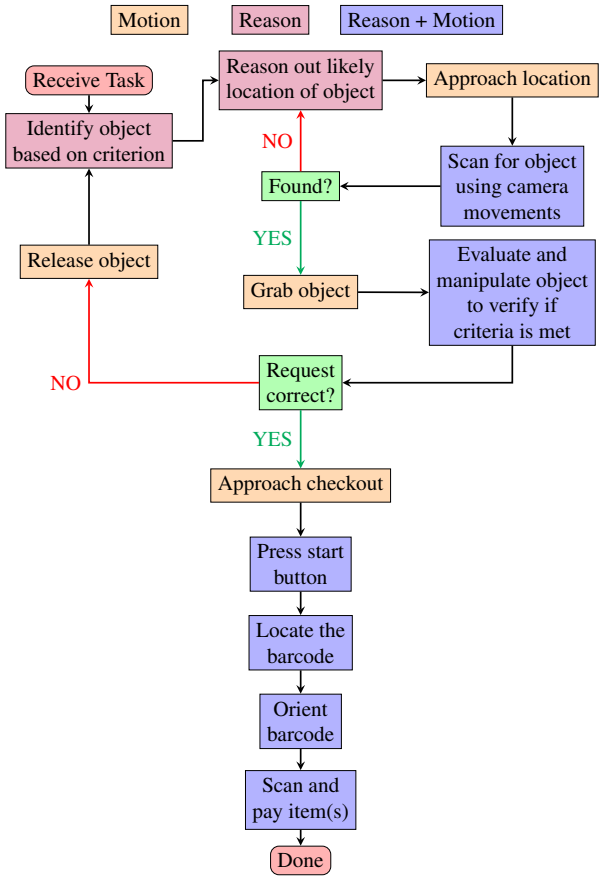


Figure 13. General thought process for average tasks.

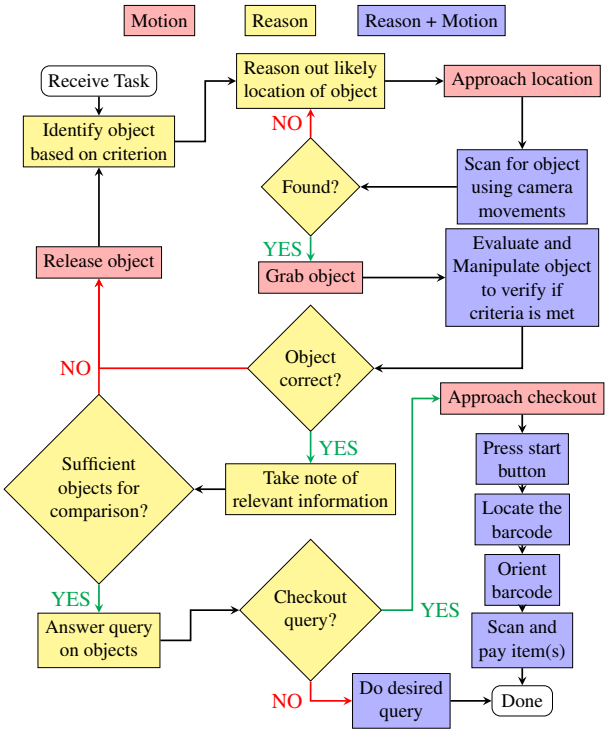


Figure 14. General thought process for difficult tasks.

volves a loop of reasoning about the object's location, scanning, and verifying if it meets the criterion before grabbing or releasing. The Average Task Flowchart (Figure 13, e.g., *"Find Koko Krunch, check for artificial flavors, and scan it if none are present"*) adds interpretation and conditional actions. After verification, participants proceed to checkout, orient the barcode, and complete the scan and payment.

In the Difficult Task Flowchart (Figure 14), participants first gather information by performing simpler subtasks, then apply this knowledge to answer a more complex query. Actions are categorized into *Reason*, *Motion*, and *Reason + Motion*. Decision points often involve reassessment and retries, revealing how humans manage uncertainty and adapt strategies. Compound actions demonstrate how participants mentally bundle related physical and cognitive steps, while selectively recalling earlier information to support later goals which underscores the role of memory and flexible planning in completing complex tasks. These diagrams show how humans integrate reasoning with action, serving as useful models for hybrid agent design.

5. Conclusion and future work

We introduce Sari Sandbox, a synthetic grocery environment with 250 items for training embodied agents, and its accompanying SariBench dataset, both serving as our prime contributions. Sari Sandbox provides an API-driven action set for varied task difficulties and uniquely incorporates human thought processes for benchmarking via SariBench's captured tasks and human demonstrations. Future efforts will focus on optimizing performance to a stable 60 FPS on mid-range desktops, expanding the dataset with dynamic additions, and enhancing realism through deformable objects and mesh colliders. A critical focus is broadening embodied agent evaluation across all SariBench difficulties to design more sophisticated embodied agents with improved navigation, perception, and manipulation, bridging the human-agent performance gap. This includes dedicated research into optimal VLM context engineering to enhance reasoning and planning. We also plan to streamline dataset generation via automated annotations, update store designs, and develop a user-friendly scene creation tool for rapid prototyping and diverse real-world simulations.

References

[1] M. Paolanti, R. Pietrini, A. Mancini, E. Frontoni, and P. Zingaretti, "Deep understanding of shopper behaviours and interactions using rgb-d vision," *Machine Vision and Applications*, vol. 31, no. 7, p. 66, 2020. [Online]. Available:

https://doi.org/10.1007/s00138-020-01118-w 2

[2] The Kroger Co. and NVIDIA Corporation, "Kroger and nvidia to reinvent the shopping experience through state-of-the-art, ai-enabled applications and services," Press release, NVIDIA Newsroom, Mar. 2022, describes a joint AI lab and demonstration center using NVIDIA Omniverse digital twins at Kroger HQ in Cincinnati. [Online]. Available: https://nvidianews.nvidia.com/news/kroger-and-nvidia-to-reinvent-the-shopping-experience-through-state-of-the-art-ai-enabled-applications-and-services 2

[3] Lowe's Innovation Labs, "Store digital twin: Giving associates "superpowers" to better serve customers," Project page, Lowe's Innovation Labs website, Sep. 2022, developed with NVIDIA Omniverse and Magic Leap 2; live in two pilot stores; includes AR restocking support, X-ray vision, and in-store simulation tools. [Online]. Available: http://lowesinnovationlabs.com/projects/store-digital-twin 2

[4] A. Afzal, D. S. Katz, C. Le Goues, and C. S. Timperley, "A study on the challenges of using robotics simulators for testing," *arXiv preprint arXiv:2004.07368*, Apr. 2020, survey of 82 robotics developers, identifying 10 major barriers to simulation use in testing and CI pipelines. [Online]. Available: https://arxiv.org/abs/2004.07368 2

[5] S. M. Kargar, B. Yordanov, C. Harvey, and A. Asadipour, "Emerging trends in realistic robotic simulations: A comprehensive systematic literature review," *IEEE Access*, vol. 12, pp. 1–26, May 2024, systematic review of ROS-enabled simulators, game-engine platforms, and AI-enhanced realistic scenario replication. [Online]. Available: https://ieeexplore.ieee.org/document/10538106 2

[6] World Robot Summit Executive Committee, "Future Convenience Store Challenge 2024 – Post-Event Report," INTEX Osaka, November 13–15, 2024, 2024, includes online rulebook and Gazebo models. [Online]. Available: https://worldrobotsummit.org/en/wrs2025/fcsc/ 2

[7] L. H. K. Wong, X. Kang, K. Bai, and J. Zhang, "A survey of robotic navigation and manipulation with physics simulators in the era of embodied ai," *arXiv preprint arXiv:2505.01458*, 2025. 2

[8] X. Puig, E. Undersander, A. Szot, M. Dallaire Côté, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlaváč, S. Y. Min, V. Vondruš, T. Gervet, V.-P. Bergès, J. M. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi, "Habitat 3.0: A co-habitat for humans, avatars and robots," 2023. 2

[9] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, Dec. 2017, published Dec 14, 2017. 2

[10] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," in *Proceedings of the International Conference on 3D Vision (3DV)*. IEEE, Oct. 2017, pp. 667–676. 2

[11] C. Gan, J. Schwartz, S. Alter, D. Mrowca, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, K. Kim, E. Wang, M. Lingelbach, A. Curtis, K. Feigelis, D. M. Bear, D. Gutfreund, D. Cox, A. Torralba, J. J. DiCarlo, J. B. Tenenbaum, J. H. McDermott, and D. L. K. Yamins, "Threedworld: A platform for interactive multi-modal physical simulation," in *NeurIPS 2021 Datasets and Benchmarks Track*, Dec. 2021. 2

[12] "Future convenience store challenge (fcsc), world robot summit 2025," https://worldrobotsummit.org/en/wrs2025/fcsc/fcsc2024/, World Robot Summit Executive Committee and METI, July 2025, competition stages held November 13–15, 2024 (INTEX Osaka) and July 13–19, 2025 (EXPO Center, Osaka). [Online]. Available: https://worldrobotsummit.org/en/wrs2025/fcsc/fcsc2024/ 2

[13] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge, "Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4392–4412. 2

[14] W. Li, X. Song, Y. Bai, S. Zhang, and S. Jiang, "Ion: Instance-level object navigation," in *Proceedings of the 29th ACM international conference on multimedia*, 2021, pp. 4343–4352. 2

[15] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel, "Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation," *arXiv preprint arXiv:2403.10506*, 2024. 2

[16] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033. 2

[17] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749. 2

[18] NVIDIA, "Isaac sim," https://developer.nvidia.com/isaac/sim, n.d. 2

[19] ——, "Nvidia omniverse," https://www.nvidia.com/en-us/omniverse/, accessed: May 22, 2025. 2

[20] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. P. Tchapmi, M. E. Tchapmi, K. Vainio, J. Wong, L. Fei-Fei, and S. Savarese, "igibson 1.0: a simulation environment for interactive tasks in large realistic scenes," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7520–7527. 2

[21] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, C. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *Proceedings of the Conference on Robot Learning (CoRL)*. PMLR, 2021, pp. 455–465. 2

[22] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied ai: From simulators to research tasks," *IEEE Transactions on Emerging Topics in Computational*

Intelligence, vol. 6, no. 2, pp. 230–244, 2022, accessed: May 22, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9687596 2, 3

[23] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023. 3

[24] D. Shah, B. Osinski, B. Ichter, and S. Levine, "LM-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *6th Annual Conference on Robot Learning*, 2022. [Online]. Available: https://openreview.net/forum?id=UW5A3SweAH 3

[25] B. Liu, X. Li, J. Zhang, J. Wang, T. He, S. Hong, H. Liu, S. Zhang, K. Song, K. Zhu *et al.*, "Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems," *arXiv preprint arXiv:2504.01990*, 2025. 3

[26] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023. 3

[27] M. Li, S. Zhao, Q. Wang, K. Wang, Y. Zhou, S. Srivastava, C. Gokmen, T. Lee, E. L. Li, R. Zhang *et al.*, "Embodied agent interface: Benchmarking llms for embodied decision making," *Advances in Neural Information Processing Systems*, vol. 37, pp. 100 428–100 534, 2024. 3

[28] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023. 3

[29] N. Rozanov and M. Rei, "Stateact: State tracking and reasoning for acting and planning with large language models," *arXiv preprint arXiv:2410.02810*, 2024. 3

[30] M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht, "Alfworld: Aligning text and embodied environments for interactive learning," *arXiv preprint arXiv:2010.03768*, 2020. 3

[31] T. Sumers, S. Yao, K. Narasimhan, and T. Griffiths, "Cognitive architectures for language agents," *Transactions on Machine Learning Research*, 2023. 3

[32] Z. Zhang, X. Bo, C. Ma, R. Li, X. Chen, Q. Dai, J. Zhu, Z. Dong, and J.-R. Wen, "A survey on the memory mechanism of large language model based agents," *arXiv preprint arXiv:2404.13501*, 2024. 3

[33] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "Memorybank: Enhancing large language models with long-term memory," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 724–19 731. 3

[34] A. Modarressi, A. Imani, M. Fayyaz, and H. Schütze, "Ret-llm: Towards a general read-write memory for large language models," *arXiv preprint arXiv:2305.14322*, 2023. 3

[35] R. Atienza, R. Blonna, M. Tan, V. Tan, and A. Mora, "Vrex: A framework for immersive virtual reality experiences," in *2018 IEEE Region Ten Symposium (TENSYMP)*, 07 2018. 5, 6

[36] D. Calandra and F. Lamberti, "A testbed for studying cybersickness and its mitigation in immersive virtual reality," *IEEE transactions on visualization and computer graphics*, vol. PP, 08 2024. 5

[37] P. Authors, "Paddleocr, awesome multilingual ocr toolkits based on paddlepaddle." https://github.com/PaddlePaddle/PaddleOCR, 2020. 7, 1, 4

[38] M. A. N. Hadi, M. Gul, M. Khan, G. N. Alwakid, and N. Z. Jhanjhi, "Benchmarking performance analysis of optical character recognition techniques," in *2024 26th International Multi-Topic Conference (INMIC)*, Karachi, Pakistan, 2024, pp. 1–6. 7

[39] S. A. Francis and M. Sangeetha, "A comparison study on optical character recognition models in mathematical equations and in any language," *Results in Control and Optimization*, vol. 18, p. 100532, 2025. 7

[40] P. Norvig and S. J. Russell, *Artificial Intelligence: A Modern Approach*. Pearson, 2016. 1

[41] X. Yue, Y. Ni, K. Zhang, T. Zheng, R. Liu, G. Zhang, S. Stevens, D. Jiang, W. Ren, Y. Sun *et al.*, "Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9556–9567. 1

[42] G. Team, R. Anil *et al.*, "Gemini: A family of highly capable multimodal models," 2025, arXiv preprint arXiv:2312.11805. [Online]. Available: https://arxiv.org/abs/2312.11805 1

[43] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. 1

[44] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," *Advances in Neural Information Processing Systems*, vol. 37, pp. 21 875–21 911, 2024. 4

Sari Sandbox: A Virtual Retail Store Environment for Embodied AI Agents

Supplementary Material

S1. Texture fidelity

As discussed in Sec. 4.2, we evaluated texture fidelity using PaddleOCR [37]. While it performs well on structured product labels, it exhibits limitations in more challenging scenarios. Figure S1 and Figure S2 provide qualitative examples where OCR accuracy declines. Figure S1 presents a case involving mixed horizontal and vertical text orientations, which often result in incorrect segmentation or recognition. Figure S2 highlights the challenges posed by stylized fonts and logo-like text, where decorative design elements interfere with accurate character detection. These examples encourages OCR-aware scene design and the potential need for active viewpoint control in embodied agent pipelines.

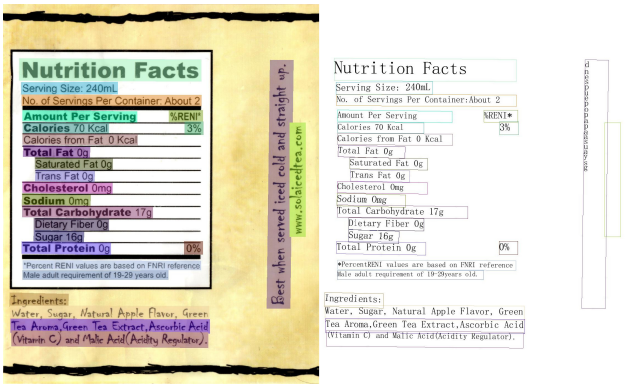


Figure S1. PaddleOCR failed in identifying the rotated text because majority is horizontal text.

S2. Embodied agent

An agent is an entity that perceives its environment through sensors and acts upon it using actuators [40]. As illustrated in Figure S3, the agent, a modular system, operates within its environment (sandbox) through a continuous loop of perception, cognition, and action to achieve a set goal. Upon goal completion, it halts until a human provides a new directive. This paradigm is powerful as it shifts from static programming to a dynamic, interactive model where systems are both observers and actors.

For our agent's cognitive engine, we required a VLM instead of an LLM, incorporating visual sensory inputs. We based our selection on the Massive Multi-discipline Multimodal Understanding and Reasoning (MMMU) [41]. At the time of evaluation, Gemini 2.5 Pro was state-of-the-art, achieving a score of 84.0 on the MMMU validation set (using its experimental Deep Think mode), nearing the human expert benchmark of 88.6 [42].

Figure S2. PaddleOCR failed in identifying stylized text of the Moby Brand despite being able to identify texts in the same image.

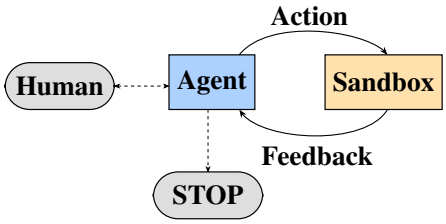


Figure S3. The elementary concept of an autonomous agent.

In our sandbox, the agent faces two core challenges: autonomous navigation and object manipulation, both vital for product search and retrieval. We adapted the ReAct framework for our needs. To focus on core capabilities, we simplified the embodied agent's navigation and manipulation; complex behaviors like multi-item handling or checkout were excluded. We forego A* planning [43] as our embodied agent lacks access to a pre-built grid-map, relying solely on visual input during navigation. Our primary objective is not to develop a state-of-the-art agent, but rather to construct a functional one that rigorously tests the practicality and usability of the APIs designed in Section 3.6 through basic item search and retrieval tasks.

Our embodied agent operates using the pattern in Figure S4, which is designed for both efficiency and control when processing a grocery task (e.g., *"Find a healthy snack"*). Our key approach here is that the embodied

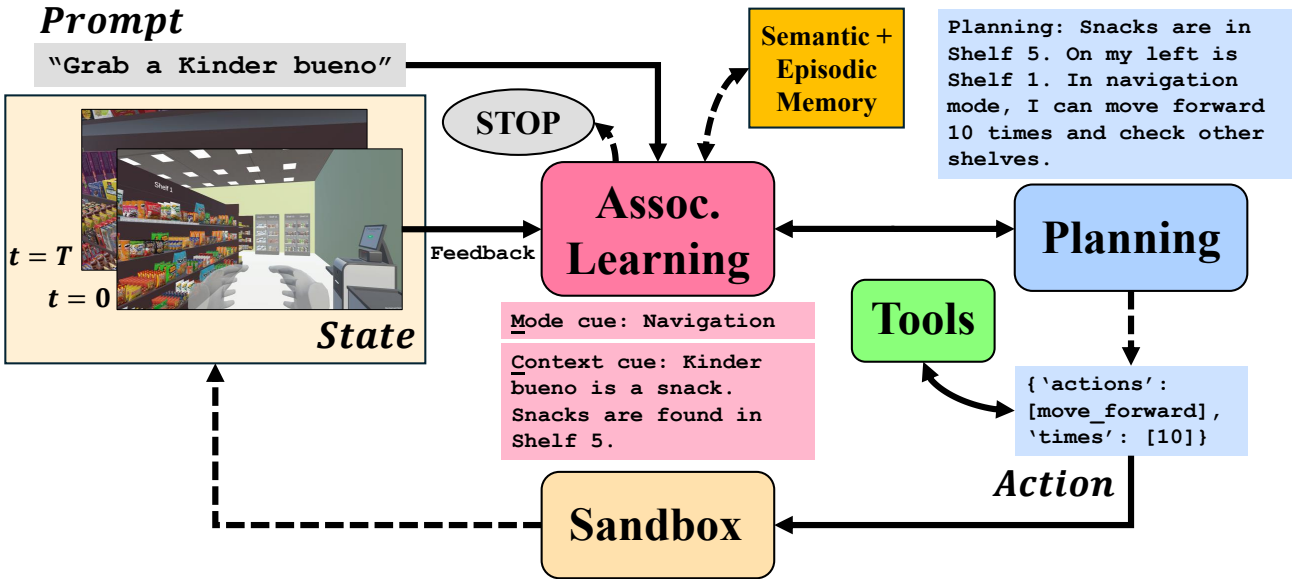


Figure S4. An overview of our agentic pattern. The process begins with a prompt, initiating a continuous loop. In the initial **associative learning** step, the embodied agent consults its semantic and episodic memory, and synthesizes its current state and the given task into mode and context (M+C) cues. These cues are included as inputs for the **planning** step, where the VLM generates an action sequence. This sequence, comprising elementary actions and potential tool calls, is executed within the sandbox. The loop then repeats with the new state, continuing until the associative learning step issues a `STOP` command.

agent generates an action sequence—a chunk of related commands—rather than a single action per cycle. This approach is highly effective for repetitive tasks such as grocery shopping (e.g., *"navigate to Shelf 1, search; navigate to Shelf 2, search; navigate to Shelf 3, grasp"*). This pattern effectively manages such decomposed, sequential steps. Furthermore, this design substantially offloads the cognitive load from the VLM. The associative learning step pre-processes and provides mode cues (e.g., indicating navigation or manipulation) and context cues. This means the planning step does not need to deliberate on these operational states, streamlining its decision-making. Consequently, this pattern makes debugging easier and provides more robust control over the outputs.

The agentic loop operates through three key steps. First, the **associative learning step** queries the sandbox for the embodied agent's current state (coordinates, visual input) at each step. It then processes this state against the prompt to generate two cues for the next stage: a **mode cue** and a **context cue**. The mode cue switches the embodied agent's operational state between navigation and manipulation, constraining the VLM to specific symbolic actions for improved reliability. The context cue provides recall information from the embodied agent's semantic and episodic memory. Second, **planning step** takes this state information, along with the mode cue, context cue, and available tools (function-calling APIs), and passes them to the VLM, which generates a structured action sequence. Finally, during the **execution step**, an external parser translates this sequence into executable API calls dispatched to the sandbox, yielding an observable result. The embodied agent perceives this new state, and the loop repeats until the associative learning step issues a `STOP` command.

To overcome the stateless nature inherent in VLM-powered embodied agents, our embodied agent explicitly models memory, leveraging the four-part cognitive architecture. This framework, managed within our agentic pattern, comprises the following components: **procedural memory** which contains the embodied agent's core rules and skills, implemented as the system instructions provided to the VLM; **working memory** which contains the immediate interaction history, managed by caching and injecting context within the VLM's context window; **semantic memory** which contains the factual knowledge about the environment; and **episodic memory** which contains the distilled takeaways from the embodied agent's own experiences. To implement the semantic memory, we first created a base semantic memory which is a text file containing the store layout, product locations (e.g., *"Shelf 1 contains cereals"*), and rudimentary directions from the embodied agent's spawn point to any shelf (mimicking natural instructions). Similarly, the episodic memory is implemented as a text file, which starts blank at the beginning of each task.

The semantic and episodic memories are managed by a memory writing operation that occurs during the associative learning step. At each timestep, this module consults

both memory files. It uses the semantic memory to ground the embodied agent in its environment. After action execution, the associative learning step updates the episodic memory by synthesizing a three-point reflection on the action that had just been executed: a dense summary of what occurred, what actions worked, and what to avoid in the future. The key information recalled from both memories is then encoded into the context and mode cues, and passed to the planning step to reduce the cognitive load and enhance context. It is important to note that while the associative learning and planning steps use the same Gemini model release version, they function as distinct modules with different system instructions.

S3. Actions and tools

Our embodied agent interacts with the Sari Sandbox environment through a defined set of actions, categorized into distinct operational modes: navigation and manipulation (Table S1). These actions enable precise control over the embodied agent's movement and interaction with objects within the simulated grocery store.

The navigation mode allows the agent to control its body's position and orientation. This includes fundamental actions such as `move_forward`, which advances the agent by 0.1 units, and `pan_left` and `pan_right`, which rotate the agent's view horizontally by 2.5-degree increments. While these appear as single, high-level API calls, their underlying implementation involves iterative, atomic calls to the simulator's core API functions. For instance, `move_forward` directly invokes `TransformAgent((0, 0, 0.1), (0, 0, 0))`, allowing for controlled, fine-grained movement up to a predefined unit limit. Similarly, `pan_left` and `pan_right` are built upon `TransformAgent((0, 0, 0), (0, -2.5, 0))` and `TransformAgent((0, 0, 0), (0, +2.5, 0))`, respectively, to control the agent's yaw rotation.

Table S1. Different modes of operation and their associated actions. Action descriptions: `move_forward` to move the embodied agent forward by 0.1 units in the sandbox. `pan_left` and `pan_right` to pan left and right by 2.5 degrees, respectively. `center_object_on_screen` to center the embodied agent's body on the target object in the frame. `retrieve_item` to approach the target object, grab it with the embodied agent's hand, and inspect it. Navigation and manipulation invokves `TransformAgent` and `TransformHands`, respectively.

Mode	Actions
Navigation	`move_forward`, `pan_left`, `pan_right`
Manipulation	`center_object_on_screen`, `retrieve_item`

The manipulation mode enables the agent to interact directly with items in the environment. This includes actions like `center_object_on_screen` and `retrieve_item`.

The `center_object_on_screen` action leverages Gemini 2.5 Pro for object detection. It uses visual inputs (obtained via first-person point-of-view screenshot within the sandbox) to calculate the target object's bounding box. The VLM's perception output after calling `loc_object` tool, providing `ymin`, `xmin`, `ymax`, `xmax` coordinates, is then translated into pixel coordinates. Based on the object's horizontal and vertical deviation from the screen center, the agent directly invokes `TransformAgent` to perform precise yaw and pitch rotations, aligning its perspective with the object.

The `retrieve_item` action is a more complex, compound behavior that orchestrates several steps:

- **Depth estimation**. The agent first moves generally towards the detected target using visual input and estimated depth via `est_depth` tool. This often involves `move_forward` actions, which, as described, translate to repeated `TransformAgent((0, 0, 0.1), (0, 0, 0))` calls.
- **Fine-tuning orientation**. It then adjusts its orientation to face a cardinal direction for consistent alignment, again utilizing `TransformAgent` for precise yaw control.
- **Horizontal centering**. The embodied agent performs a `strafe_to_center` operation to precisely align itself with the object. This action calculates the horizontal offset of the target object's bounding box from the image center. It then converts this pixel offset into a required linear movement in world units. Subsequently, `strafe_to_center` executes a series of granular calls which correspond to `TransformAgent((+0.1, 0, 0), (0, 0, 0))` or `TransformAgent((-0.1, 0, 0), (0, 0, 0))` to incrementally shift the embodied agent's body sideways until the object is horizontally centered in its view.
- **Final approach and interaction**. The embodied agent moves to the item's immediate vicinity and executes the physical `grab_and_read` operation. This critical step involves a choreographed series of atomic hand movements directly implemented via the `TransformHands` API. Specifically, the agent can extend its hands forward by adjusting their Z-axis position, pull them backward for Z-axis retraction, or raise and lower them to control their Y-axis position. Rotational actions, also achieved by `TransformHands`, allow for precise manipulation of the hand's yaw rotation. Once positioned, the agent can grasp the item using `ToggleLeftGrip` API calls to simulate a grip. Following the grab, the `grab_and_read_item` operation initiates a new screenshot and then processes this image using an Opti-

cal Character Recognition (OCR) tool via `ocr_object` to extract any text present on the item, thereby simulating visual inspection. These primitive hand and vision-based interactions are crucial for the embodied agent to accurately reach, grasp, and inspect the target item, even though they are not exposed as high-level API actions in the table.

The VLM serves as the cognitive engine for these actions. It processes visual information and textual prompts to determine the appropriate sequence of actions and their parameters. Tools found in Table S2 are integral to the embodied agent's perception and decision-making loop, particularly for tasks requiring object identification and precise interaction. The structured nature of these higher-level actions, built upon the fundamental `TransformAgent` and `TransformHands` APIs, ensures the embodied agent can perform complex grocery tasks by decomposing them into manageable, executable steps.

Table S2. Tools that are part of an elementary action and their corresponding purposes.

Tool	**Purpose**
`loc_object`	Uses Gemini 2.5 Pro's object localization capability to output bounding box coordinates of a specified item or item of interest. Format: `[ymin, xmin, ymax, xmax]`. Part of: `center_object_on_screen`.
`ocr_object`	Uses PaddleOCR [37] for item inspection via optical character recognition (OCR). Part of: `retrieve_item`.
`est_depth`	Uses Depth-Anything-V2 [44] Small for computing the distance between target object and embodied agent before item inspection. Part of: `retrieve_item`.