Robot with Memory: Integrating Memory, Dialogue, and Fetching for Table-top Robots

Robert Bao
Department of Computer Science
University of Virginia, United States
cb5th@virginia.edu

Jade Gregoire
Department of Computer Science
University of Virginia, United States
dze3jz@virginia.edu

Abstract: Lots of elderly people have retired in recent years. Among them, many face challenges with memory and mobility: they sometimes struggle to remember where they placed items, and even when they do, retrieving them can be difficult. This project develops a robot that helps them with both tasks, attempting to make their lives much easier—by utilizing a JSON memory system and making use of vision to LLM to analyze robot camera feeds. With testing, the robot was shown to be able to recall item locations very successfully, but struggled with fetching larger items.

Keywords: Robots, Learning, Dementia

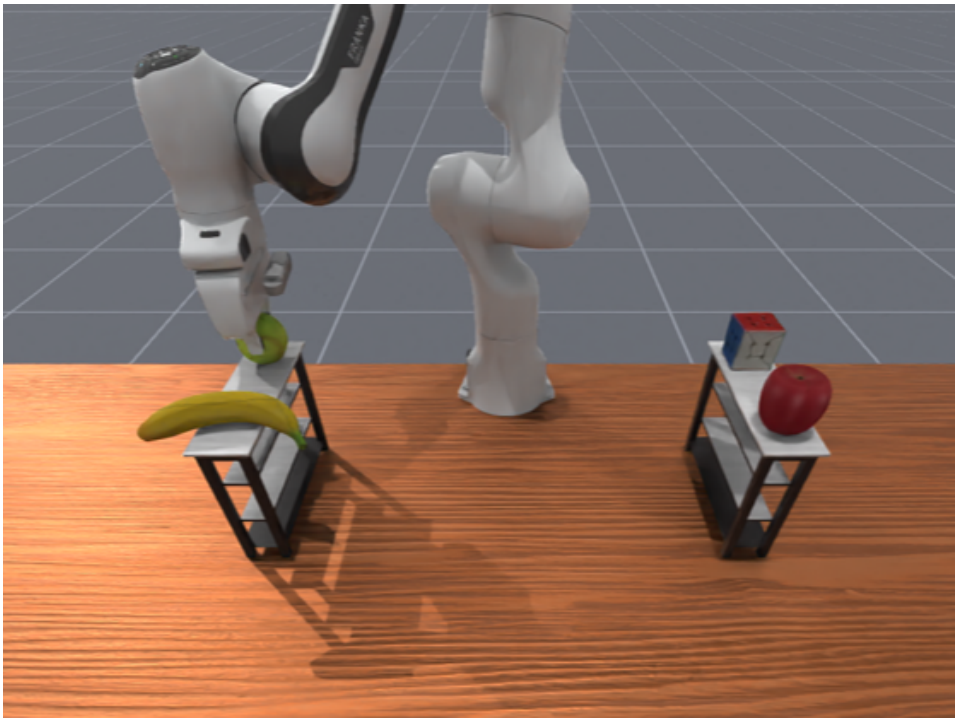

Figure 1: The Pandas robot picks up a softball in a table-top setting.

1 Introduction

As of 2021 in the U.S., almost 40% of people above age 65 experience cognitive impairment and memory issues without dementia—and about 10% of people above age 65 get diagnosed with dementia [1]. Previous works in the robotic field have explored developing assistive robots for the

elderly, particularly to help with memory issues, as well as mobility issues that arise from older age [2, 3]. However, these fail to account for ease of use, or lack user customization. This project expands on these works and combines multiple features to make a robot with three main helpful components: (1) memory, (2) dialogue, and (3) physical interaction. A memory system means that elderly people can just ask a robot for where items are, instead of struggling to remember by themselves. By developing a dialogue system, we also remove a barrier to using the robot, making it intuitive, non-restrictive, and having no learning curve that comes with a GUI or other sort of interface. The last component is straightforward—the robot will be able to fetch items and objects for the elderly, allowing them to stay put as the robot helps them.

We evaluate our robot with recall success rate—testing its object identification and memory skills—as well as a fetch success rate—testing its item retrieval capabilities. The robot excels at recalling objects, and is successful at fetching, but struggled with picking up larger objects.

2 Related Works

Previous works have explored helping the elderly with robots. One paper specifically focuses on elders with dementia—the researchers develop an episodic memory model for salient objects [2]. Their method identifies important objects to the user, and uses a detection software to label the objects, tracking where the object might change location over time from the robot's point of view. Users are able to interact with the robot with a GUI to fetch items in its memory—our project furthers this idea by making the robot dialogue-based, and we take inspiration from how the researchers' robot's memory system works. We also intend our system to be usable for non-salient objects, making sure our object detection system can pick up on subtle nuances to help users with whichever object they would like.

Another paper similarly explores and presents a robot-dialogue framework, KURT [3]. the KURT system stores information about entities with 3 characteristics: locations, objects, and properties. The system is also proactive: it can ask the user questions for clarity on their instructions or alternatives for objects. For our project, do not care so much about being proactive, but about the dialogue method developed in the paper, and potential tags we could add to the data in memory storage—we mimic this structure in our own work. Rather than asking users leading questions for clarity, our robot informs users of failures that occur, allowing them to determine which next steps they would like to take.

A more general household assistant robot is created with an interactive learning framework that includes perception, action, learning, memory, and reasoning, all of which could be useful to the processes of our own robot [4]. The robot has a camera on it for object detection, a long term memory system for things they have learned through a user, and short term memory for things the robot perceives on its own (where a *thing* may be an idea, order of steps to execute an instruction, object location, etc). The robot performs object fetch tasks in a home environment, which we also attempt. Similar to paper [2], this robot relies on GUI interactions (though they mention replacing the GUI with a text-to-speech system as being feasible with their setup, as future exploration). We take inspiration on how the paper uses perception and sets up memory, but do not implement learning—we also replace the GUI with a dialogue system.

Our work contributes new exploration into the assistive robot field by building upon and combining existing robot features on robotic memory, dialogue-frameworks, and fetching systems.

3 Problem Setup

In this project, we build a household robot that (1) remembers and recalls object locations and (2) physically fetches objects based on user prompts. The robot's main goal is to help elderly users—who may struggle with issues related to memory loss—find common items in a home setting.

For further detail, the robot will take the following information as inputs:

- **User prompts to recall and fetch an object:** The robot includes a language component, which enables it to converse with the user. This provides it with an intuitive interface for testing.
- **Images from the robot's camera feed:** The robot relies on its camera feed, provided by the ManiSkill simulation framework, to "see" objects and scenes in the environment. The camera feed simulates sensors in real-life that are attached to either a fixed location nearby, or on the robot itself. The image it captures enables the robot to have enough information about the simulated world to carry out the task prompted by the user.

The robot will produce the following types of outputs:

- **Natural language response on object locations:** The robot is programmed to respond with the locations of the objects requested, which helps it answer questions from the user. It gives feedback if it fails to perform any action, and lets the user know why it was unable to.
- **Successful delivery of the fetched objects:** A major purpose of the robot is to fetch and deliver objects to the user. Even though this step does not provide any output, we intend to collect video footage of the robot completing the task as proof of such.

Finally, given the above setup, the robot performance will be evaluated with two metrics: object recall accuracy and fetch completion success rate.

- **Object Recall Accuracy:** The probability of the robot successfully recalling the correct object when prompted in testing.
- **Fetch Completion Success Rate:** The probability of the robot successfully fetching an object—whether using location stored in its memory, or acquired through camera feed/point cloud—and brings it to the user. This task is expected to be more difficult than pure recall, as it involves both locating the object and navigating/fetching it.

Both metrics are expressed as percentages. They measure the robot's performance on its two most important tasks. The results are discussed in Section 5. An image of the robot's setting during testing is shown in Figure 1.

4 Methods

4.1 Environment Setup and Robot Motion Planning

Our system design decisions evolved as we developed the elderly care robot / memory bot. Initially, we used the RepliCAD ManiSkill scene for our robot's simulated environment, which has a kitchen and living room area set up. This environment relied on the Fetch robot, which had 13 joints, including an arm and a Roomba-like bottom for environmental navigation. As perfect as this environment was to mimic a real life scenario for the robot, there were issues handling motion planning—and since motion planning is a problem solved by many others that is not within the focus of what we are attempting to address, we took a different approach.

In order to circumvent issues with complex motion planning, we decided a different environment would help us better evaluate our robot. We switched to using the Franka Emika Panda Robot, in a table-top setting. With only eight joints, this robot was simpler to implement motion planning for. We adopted two motion planning algorithms—Screw and Rapidly exploring Random Tree (RRT)—to control the robot arm. The motion planning algorithms calculates the desired robot arm poses based on the current position and destination, and enable the robot arm to move to arbitrary positions within its reach. We also build logic to enable the robot to reliably grasp objects that are narrow enough to be held by the robot gripper. This setup minimizes complexity previously posed by the complex and large area of movement in the RepliCAD apartment scene, and makes it possible for us to focus on the robot's memory component.

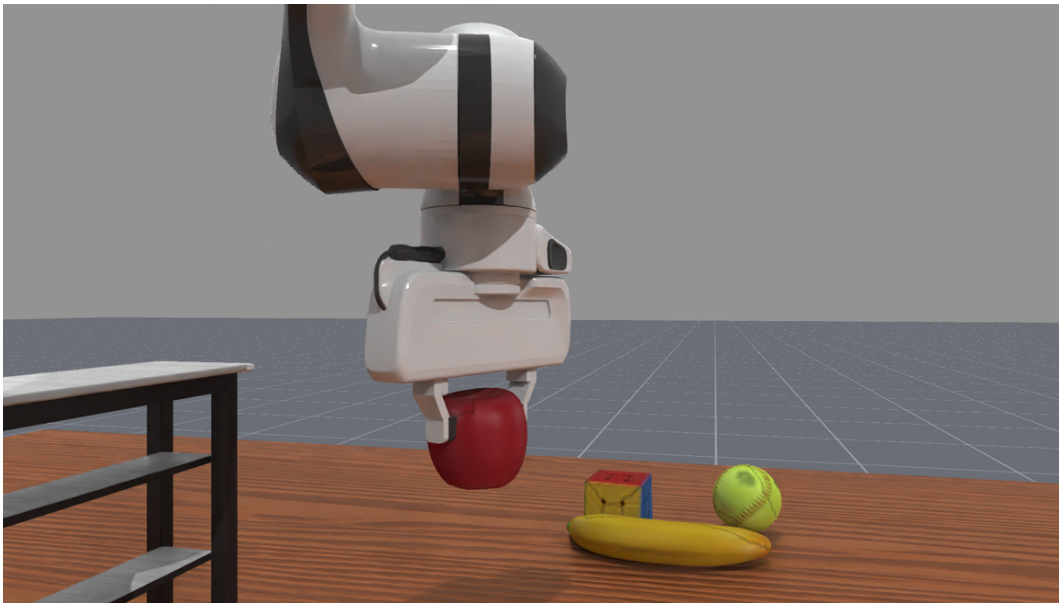

Figure 2: A view from the robot's camera as it fetches an apple.

4.2 Camera

To observe the environment around it, the robot uses a custom mounted camera mounted on its base link (Joint 1). The camera is high-resolution, and it captures images with dimensions of 1920 x 1080. The Pandas robot's base link rotates horizontally, which allows the camera to point in different directions. We intentionally limited the field of view of the camera to 45 degrees ($\frac{\pi}{3}$) radian). This way, the camera can only capture a limited view of the environment, preventing the robot from having an overly wide view of the environment; this makes it easier to test the robot's memory component in the evaluation phase of the project, as the robot is forced to rotate around to identify objects.

Figure 2 shows the camera's captured output from the robot's perspective. As shown, the camera is clear enough to show the color and texture of each object in the scene: the apple, banana, softball, and Rubik's cube are clearly visible in the picture. This makes it easier for the robot's VLM component to analyze and identify objects in the images.

4.3 Vision

The robot uses a vision model to observe the "world" (e.g., the environment around it). Originally, the robot recognized objects using the YOLOv8 model, which lacked the ability to derive further details beyond object name. This gave us issues when prompting the robot to distinguish between two slightly different objects of the same type (e.g., two books of different cover colors) in the same scene. We fixed this issue by switching to GPT-4o, which was able to analyze camera feeds much better, giving us information in a format we could manipulate appropriately to enable the robot to identify objects. The VLM would be prompted to identify objects with their name, additional details (size, color, etc), a description of their location, and their location relative to the robot's camera view. To note, this model was much more computationally expensive, and still had its flaws—it struggled with ambiguous or poor-quality images (e.g., would identify a "red can" instead of a "tomato soup can").

4.4 Command Handling

To simulate robot dialogue, our set-up allows a user to type any command they wish, and the robot prints responses to the terminal. Once a command is given to the robot, we use ChatGPT to parse the instruction into a JSON format containing the command's action (whether to recall an object's location, fetch an item, or reply generically), name(s) of the object(s) relevant to the command, and detail(s) about the object(s).

The three actions our robot can handle are as follows:

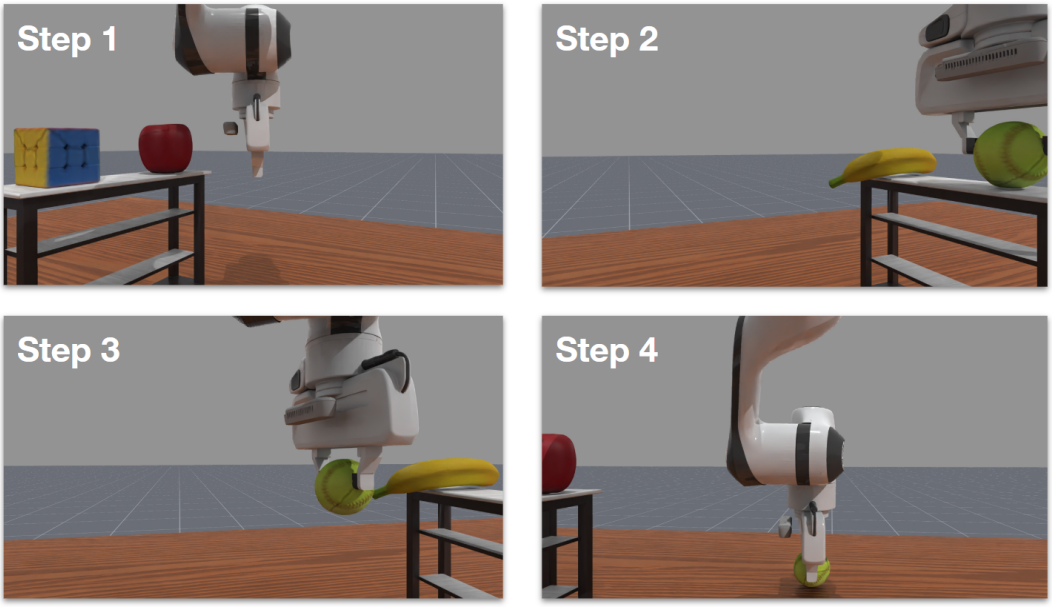

Figure 3: Images from the robot's camera as it goes through the steps of searching, identifying, and fetching a softball.

- **Recall:** A user can ask the robot for an item's location.
- **Fetch:** A user can request for the robot to fetch an object.
- **Generic:** The system is also able to respond to generic communication input.

Recall is done by querying the items in the robot's database, and making use of calls to ChatGPT to check whether the items the robot has stored in memory are the same as the items requested by the user. This is to avoid queries from failing due to a change in vocabulary use—for example, the robot may have previously stored an object under the name of "trash bin", but the user might request for the location of the "garbage can", which are the same objects, just under different names. Of course, there may also be two of these items, and they would be distinguished by other object properties such as details, which are also compared in this query search. If the object is not found in memory, the robot searches for the item to save its location and tell the user. The robot will use its camera to identify objects within its field of vision, rotating to view all possible angles around it. A call to GPT-4o is made, to identify objects in view and where they are located in the robot's field of vision. Once the robot's surroundings are analyzed and the item is found, the robot will store the requested item's location in its memory. If the object is not found, the robot will notify the user.

To fetch an object, the robot first recalls where the object is. After successfully recalling an object's location, the robot will go to it's most recently saved location, grab the item, and then bring it back to the original spot the robot was when it received the command. Images from the robot's camera as it satisfies the steps to recall and fetch a softball can be observed in Figure 3. If the robot is unable to recall an object, it will let the user know through the command-line interface.

As for other generic communication, the robot retains a chat history with the user, and has basic human interactions. For example, a user can ask questions such as "What have we done so far?" and "List the objects you fetched", which the robot will be able to respond to. It can also notify users that it is not able to handle commands outside of recalling and fetching. This is done to enable natural interaction between the user and robot.

A full demo of the robot's main components and a video of it in action are are available here.

4.5 Memory

For the robot's memory component, we utilized a lightweight JSON database to record each object's location. Objects are stored in this database with a name, detail (color and size, or any other relevant information), location description (relative, human-readable location description), and 3D location coordinates. Initially, we intended to store these information in a SQL database, along with a timestamp of when that information was saved. We envisioned the design to enable us to track an object's location over time. However, later in the development, we switched to a JSON memory system for better ease-of-use, and timestamps were not needed as a most recent entry could easily be fetched without the timestamp field.

In order to limit the project's scope, we decided to use a location oracle. The oracle is a JSON file, and it provides the robot with objects' 3D coordinates once it was spotted and identified, simplifying the object retrieval process significantly. As the robot observes the environment, it simply matches the object's name to the coordinates in the oracle. The robot's object detector component then combines the coordinates with the information captured by the VLM, and stores it in the memory DB. This setup dramatically simplifies the process of locating objects, making the integration step easier given our project's scope. Additionally, in the future, the oracle component could be replaced by the use of a point cloud to simulate LiDAR.

Finally, in addition to the specific object locations (tracked by the Memory DB), the robot also has basic temporal memory over what happened during a simulation session. The robot's command handler keeps a record of all past message exchanges between the robot and the user, and it passes the message history to the VLM component when generating each new message. This means the robot would "remember" the sequence of events that happened as it converses with the user and carries out requests, and enables it to answer questions like "What objects did I ask you to fetch?" or "What did you do during this conversation?". Finally, based the command handler is based on OpenAI's GPT-4o model—a model with a context window of 128K tokens—the robot's memory is quite scalable to longer interactions.

5 Experiments & Results

To evaluate the Pandas robot's accuracy and success rate, the robot was instructed to recall and fetch different items from a tabletop setting. Two miniature shelves were placed on the table, with two objects on each—for a total of four items in the environment. These objects are all present in the 180-degree area in front of the robot, to keep the project scope smaller and more manageable. However, this mechanism can generalize to robots capable of rotating 360 degrees.

We tested recall and fetching on six different objects: an apple, banana, tomato soup can, pear, and toy airplane. These object models are all pulled from the Yale-CMU-Berkeley (YCB) object dataset, which makes use of high-resolution RGB images and texture-mapped 3D mesh models to best simulate real world items [5]. A total of six trials were conducted per object, where prompt wording was varied per trial, and the other 3 items in the environment were randomly instantiated each time.

Our experiment's results are viewable in Table 1. Our robot had almost perfect recall accuracy, but failed once to identify a pear—it mistakenly labeled the object as a "green apple". The robot also had decent fetch rates for smaller objects: it was able to successfully fetch the apple and banana every time. However, it failed a few times to retrieve the pear and tomato soup can, as it accidentally

Object	Robot Recall Accuracy (%)	Robot Fetch Success Rate (%)	Comment
Apple	100	100	Easy to identify and fetch
Banana	100	100	Easy to identify and fetch
Tomato Soup Can	100	83.3	Identified the object as "can" for all Recall operations; this lacks detail but is acceptable. One fetch failed after the robot knocked the can off of the shelf.
Pear	83.3	66.7	Misidentified the object as a "green apple" for one Recall operation. Knocked the pear off the shelf twice during fetch operations.
Toy Airplane	100	0	Correctly identified the object as "toy plane," "toy airplane," and "biplane toy". Failed all fetch attempts because the the toy was too wide for robot grippers.

Table 1: Recall accuracy and fetch success rate for different objects. The data was produced by repeating the robot operation six times. The environment was randomized with 3 other objects.

knocked the items off of the shelves a few times as it was retrieving them. The reason this issue occurred was most likely two-fold: the Pandas' grippers were very close to being the size of the objects' widths, and the robot did not notice that it knocked the items over—as it stopped observing the item's location change during the retrieval process. This meant that if the object's location changed after the robot finished identifying its location, the robot would not be able to correctly pick the item up because it would go to the object's old location.

The Pandas grippers were also smaller than the toy airplane's width, leading to the robot not being able to fetch the item even once. The robot was still able to identify the toy airplane's location, so this fetching success rate would theoretically be decent if the robot had a larger arm.

6 Conclusion

In this project, we were able to successfully integrate a dialogue system, memory, and motion into our robot, making it fully functional for serving its basic tasks. Our results showed that a simple memory database was effective for storing object locations and that the VLM in place was able to accurately distinguish objects from each other almost all of the time. Our project represents a "complex coordination" among different components: getting each component to work is simple, yet combining all into a functional whole is challenging. We were still able to produce good results.

There is much to explore in future work. At the current stage, we were limited in scope by using the Pandas robot; future work could be done in the more realistic RepliCAD apartment scene (with full living area / kitchen setup + Fetch robot). Additionally, the current Pandas robot system struggled with fetching larger objects due to its small grippers, and it would be beneficial to explore motion planning designs that intelligently grasp objects based on their shape to resolve this issue.

There are other potential expansions for the memory and vision components of the system. One such avenue would be to explore more lightweight vision components: using GPT-4o can be expensive when its use case is scaled up, and testing alternative smaller VLMs like Llama-11B / Llama-90B instead could prove promising. Exploring the effect of switching the memory database to MongoDB or SQL (e.g., Postgres) would also be beneficial to ensuring scalability for more complex object environments. Instead of using a fuzzy search to query the memory database, objects could be stored as a text embedding—observing how that would affect recall accuracy would be interesting.

There are many avenues left to continue exploring this topic of helpful assistant robots—our work was able to combine existing ideas into one seamless robot, but improvements can always be made.

References

[1] Y. Qian, X. Chen, D. Tang, A. S. Kelley, and J. Li. Prevalence of Memory-Related Diagnoses Among U.S. Older Adults With Early Symptoms of Cognitive Impairment. *The Journals of Gerontology: Series A*, 76(10):1846–1853, 02 2021. ISSN 1079-5006. doi:10.1093/gerona/glab043.

[2] J. Shah, A. Ayub, C. L. Nehaniv, and K. Dautenhahn. Where is my phone? towards developing an episodic memory model for companion robots to track users' salient objects. In *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '23, page 621–624, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399708. doi:10.1145/3568294.3580160.

[3] M. Kraus, N. Wagner, W. Minker, A. Agrawal, A. Schmidt, P. Krishna Prasad, and W. Ertel. Kurt: A household assistance robot capable of proactive dialogue. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '22, page 855–859. IEEE Press, 2022. URL https://dl.acm.org/doi/abs/10.5555/3523760.3523892.

[4] A. Ayub, C. L. Nehaniv, and K. Dautenhahn. A personalized household assistive robot that learns and creates new breakfast options through human-robot interaction. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 2387–2393, 2023. doi:10.1109/RO-MAN57019.2023.10309575.

[5] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015. doi:10.1109/ICAR.2015.7251504.