Intention Reading Architecture for Virtual Agents

Giuseppe Fulvio Gaglio[1][0009−0005−8749−377X], Samuele Vinanzi[2][0000−0003−0241−9983], Angelo Cangelosi[3][0000−0002−4709−2243], and Antonio Chella[1][0000−0002−8625−708X]

[1] University of Palermo, Palermo, Italy
[2] Sheffield Hallam University, Sheffield, UK
[3] University of Manchester, Manchester, UK

Abstract. This paper presents the creation of a virtual agent equipped with an advanced cognitive architecture that gives it context awareness and the ability to understand human user intentions in a virtual scenario. Such capabilities are crucial for improving the interaction between users and virtual agents within the growing Metaverse domain, making it more realistic, fluid, and efficient. The paper delves into how the cognitive architecture, initially designed for physical robots, was successfully adapted to a fully virtual environment and implemented with an LLM to create virtual assistants capable of interacting effectively with users. Despite the primarily virtual nature of this work, integrating cognitive architectures with LLMs is an important starting point toward the development of collaborative artificial agents capable of providing hands-on support through a deeper understanding of the context and user intentions.

Keywords: Cognitive Architectures · Virtual Agents · Large Language Models · Intention Reading · Context-awareness.

1 Introduction

The Metaverse is increasingly relevant in the contemporary technological scenario and in it, users, represented by avatars, can engage in social and economic interactions in creative and cooperative ways [13]. However, the user experience will also include interactions with virtual avatars controlled by Artificial Intelligence (AI), which would be used primarily for services and support. The ability of these agents to recognize users' needs and adapt to their requirements is critical to ensuring satisfactory interaction. Today's use of virtual avatars controlled by Large Language Models (LLMs) has opened new perspectives to make up for this shortcoming. However, for truly effective interaction, they need to understand our intentions. The AI landscape has experienced recently a profound revolution thanks to LLMs which are becoming increasingly sophisticated and high-performing. These models have shown great potential as virtual assistants in various domains. Still, to become true autonomous artificial agents they must be able to perceive and interpret the surrounding context and users' intentions

proactively. Cognitive architectures represent a promising approach to building artificial agents with contextual awareness and the ability to understand human intentions [4].

Through this paper, we propose an architecture based on the "Cognitive Architecture for Social Perception and Engagement in Robots" (CASPER) [12], redesigned for implementation in the popular game engine Unity. Originally developed to make robots capable of intervening to help a human partner in the real world, the fully virtual version of this architecture can be successfully applied to scenarios such as video games and the Metaverse. It has also been enhanced by integrating it with an LLM, whose task is to communicate in discursive form the operations of recognizing both the context and the user's intentions, which are already performed by the cognitive architecture above. This approach allows specific agent functions to be activated in response to the user's needs and goals, thus giving the agent greater autonomy and adaptability.

The paper is structured as follows: Section 2 will discuss the state of the art of the involved technologies and will report a presentation of the original architecture and the work that inspired and motivated this experimentation; Section 3 will outline the work of readjusting the architecture from the original environment to that of Unity; Section 4 is devoted to a discussion of the results achieved and the actual differences of the new version from the original; finally, Section 5 will present the conclusions and envision future developments.

2 Background

One of the most comprehensive definitions of Metaverse reads: "Metaverse, a crossword of "meta" (meaning transcendency) and "universe", describes a (decentralized) three-dimensional online environment that is persistent and immersive, in which users represented by avatars can participate socially and economically with each other creatively and collaboratively in virtual spaces decoupled from the real physical world" [9]. However, people will not always be dealing only with other people: today AI is making considerable strides, and applications in which is possible to converse with virtual avatars are already quite widespread. It is more than plausible, therefore, to think of AI-controlled avatars in the Metaverse, especially for service activities [7]. Thus equipping such agents with the ability to recognize our needs and adapt to our requirements would lead to an incredibly satisfying interaction [3]. In video games, so-called Non-Player Characters (NPCs) are program-controlled avatars that famously serve as friends, assistants, allies, traders, or extras within a video game. The success of many video games and virtual reality platforms lies precisely in the realism of these entities. Their physical appearance and behavior can greatly influence the user experience. People empathize with them in the same way as they do with a character in a movie, and very often even go so far as to apply the Theory of Mind, thus attributing mental states to them based on the situation. Even in the field of social robotics, when virtual agents and avatars are used to carry out experiments before using the physical robot, it is possible to see the effects

these entities have on participants' emotions and sense of trust [15]. This is why research in the field of AI and Cognitive Science has been taking a lot of interest in NPCs lately and trying to endow them with cognitive aspects and autonomy. Already, by providing contextual awareness to an LLM we achieve better performance in the model's fulfillment of our needs. Therefore, it becomes a must to explore the combination of NPCs equipped with contextual awareness and modern chatbots [5].

The ability of LLM-equipped chatbots to mimic human intelligence has made them very promising for the development of autonomous agents. More and more attempts are being made to equip such models with cognitive capabilities such as memorizing and planning actions for which context awareness is also necessary. Studies then started on how to effectively make LLMs context-aware in terms of autonomous acquisition of information and, in particular, spatial information [14].

To give agents context awareness and autonomy, the use of cognitive architectures is one of the most currently practiced approaches. There are several examples in the literature of integrating cognitive architectures such as SOAR or ACT-R and the very popular Unity game engine to create context-aware virtual agents capable of making autonomous decisions [11, 10, 6]. However, developing native solutions in Unity would be preferable for consistency, integration with Unity features, and ease of sharing projects among developers. This would also encourage continuous updates and greater sustainability of projects.

The work described in this paper is part of a broader project aimed at creating virtual moral agents able to counsel users in the Metaverse. The previous step [2] led to the implementation of LLM-based virtual agents, whose context knowledge, however, is entered manually by the programmer through prompts. This is why a cognitive architecture that serves as a bridge between the operating environment and the LLM was considered. The choice fell on CASPER, even though it is an architecture designed for robotics and thus for the physical world.

CASPER is designed to improve the perception and social engagement of robots, enabling them to interpret and predict human intentions and collaborate in shared tasks. It uses a symbolic approach and qualitative spatial reasoning (QSR) to process and predict human actions and goals in dynamic environments. The CASPER core includes parallel processes that analyze low-level actions and high-level goals, verified by a knowledge-based system to ensure consistency and appropriateness of robot behavior in social contexts.

CASPER is a modular system written in Python that can be integrated into various robotic platforms with vision capabilities. Tested in a simulated environment, CASPER has been shown to identify and adapt to human goals using a probabilistic plan recognizer and a decision-making strategy that combines data and conceptual reasoning.

3 Methods

CASPER's migration from a Python environment to one in C# required several adaptations and substantial changes in the various modules that make up the architecture. The choice was made to ensure the creation of a native architecture for the Unity platform that was highly scalable and easily editable. We will call this new version CASPER for Virtual Environments (CASPER-VE) to distinguish it from the original version, from now on called simply CASPER. In addition, although the Metaverse is a virtual rather than a physical context, the choice to use the same scenario as CASPER, i.e., a kitchen with objectives such as having breakfast or lunch, was primarily driven by the need to have an actual example to follow. This also made it possible to compare the performance of the new implementation with that of the original version and to speed up the architecture conversion process overall.

Fig. 1. Game view of the experimental environment. The agent is represented by the grey capsule on the right.

3.1 Objects detection

As outlined before, a key feature of CASPER is the use of QSRs to understand the surrounding context. In the original implementation, the robot observed its environment and identified objects in its field of view, keeping track of them in memory. In a virtual environment, object detection differs: virtual objects come with encoded information like size, location, material, and name, which can be queried directly without visual perception.

Unity uses a technique called ray casting to obtain information about objects in a virtual scene. This simulates sending a laser ray in a specific direction to return details about the hit objects. Rays can be configured with a maximum distance or a layer mask to ignore certain objects or materials, or to detect specific objects (see Fig2). Multiple rays can simulate a human's field of view.

In our implementation, we used a method called Overlap Sphere, similar to ray casting but gathering information from all directions. It creates an invisible

sphere with a specified center, radius, and layer mask, detecting objects that touch or are inside it (see Fig3). The sphere's center is the agent's body, represented by a simple capsule, with a radius of 2 meters. The layer mask detects only objects tagged "Object of Interest" (OOI). We made the sphere barely visible for more control.

Along with objects, the user's location is also detected. All this information is constantly sent to the first CASPER-VE module, Perception, described below.

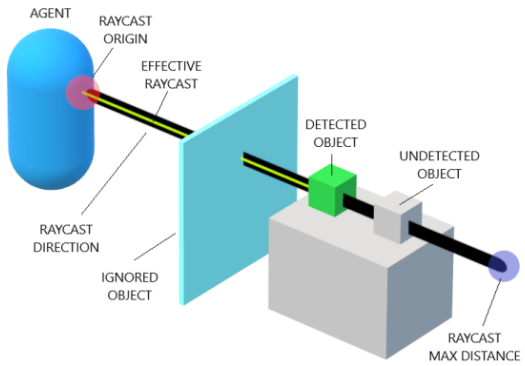

Fig. 2. Schematic functioning of Unity's Raycast method. Once an origin, direction, and max distance (fixed or infinite) are specified, all information about the first object hit is obtained. Objects placed behind that object are not detected unless a layer mask is set.

3.2 Perception

The Perception module computes the QSRs between the user and detected objects in the scene. Due to the absence of dedicated QSR libraries in C#, we developed specialized classes and functions. Leveraging Unity's built-in functions, which utilize geometric relationships between vectors, we processed information for QSR calculations: Qualitative Distance Calculus (QDC) determines proximity between the user and an object by calculating the distance between their position vectors; Qualitative Trajectory Calculus (QTC) calculates the user's trajectory towards an object by comparing his previous and current positions, identifying also if the user is moving or stationary (MOS), and considering the user's orientation angle; HOLD identifies if the user is holding an object by calculating the distance between the position vector of the object and the position vector of the user's hand.

Each QSR's details were stored over time in a custom QSR library, associating objects with key-value pairs. This approach allowed us to maintain a historical record of QSR changes for each object, facilitating access to stored information in subsequent system modules.

Fig. 3. Overlap Sphere and detection of two OOIs displayed in the Scene view. Unlike Raycast's layer mask, which allows to ignore and pass through specific objects, the Overlap Sphere's layer mask allows only detection of specific objects.

3.3 Low-Level

The low-level module is crucial for action recognition in the virtual environment. As in CASPER, we used a probabilistic calculation on QDC and QTC to identify the user's object of attention. Then a decision tree model recognizes movements (WALK, STILL, TRANSPORT, PICK, PLACE) after receiving all the QSRs as input. Each new movement different from the previous one is added to a list to form an action. A Markov-Chain generator creates three possible movement sequences for predefined actions (RELOCATE, PICK AND PLACE, USE) based on transition probabilities. These sequences are compared with observed ones to identify the user's current action using the Ratcliff-Obershelp algorithm. Depending on the object with which it is associated, the USE action can become COOK, EAT, SIP, or WASH. For more details, see the related article.

A key feature of this new implementation is that recognized actions are associated with their target objects before being stored for the next module. Instead of generic actions like Pick and Place, more specific actions such as Pick and Place Biscuits, Pick and Place Meal, Pick and Place Bottle, and Pick and Place Plate are identified. This specificity is important for the next module to better recognize user intent based on observed actions.

3.4 High-Level

The High-Level module compares observed actions with sequences in an action plan library using an algorithm that evaluates the percentage of observed versus unobserved actions to predict the user goal. For more details on the algorithm, refer to the dedicated article.

A goal planner was used to embed actions, starting states, subgoal states, and goal states to populate the goal library. The goal planner's basic elements are actions, implemented through classes. Each new action requires a name, target

object, preconditions that must be met before it can be executed, and effects that result from it.

The identified goal, along with some preceding information, is then used by the final module, which communicates and performs actions in the environment.

3.5 Supervisor

The Supervisor module is the one assigned to the interaction between the agent and the user. In CASPER, once the user's goal is recognized, the agent leverages another goal planner to act in such a way as to assist the user. In CASPER-VE we chose to integrate an LLM as an intermediary between the architecture and the user. Now the agent communicates the results of the architecture's IR processes and provides advice based on the information already collected by the architecture. In addition, instead of using another goal planner, the LLM could interact directly with the system code thus allowing the activation of actions to help the user according to his needs effectively.

The implementation of the LLM in the module was made via the Microsoft Azure API, which provides access to a GPT-4 model. A series of initial prompts then instructs the model on how to behave as if it were a role-playing game.

4 Results and Discussion

To test CASPER-VE performances, we repeated the same experiments performed with CASPER. The machine learning models used to recognize users' movements and actions were trained on the same data of CASPER and therefore, although they were implemented through libraries belonging to the .Net framework, with particular use of the ML.Net library, they produced the same results. Some differences, however, were noted in goal recognition precisely because of the inclusion of the target object within the plans. In fact, without the inclusion of the targets, the prediction results were the same of CASPER, but with their inclusion, there was greater accuracy from the first observation, as can be seen in Table 1.

In addition, the information provided by the High-Level module to the Supervisor also contains the position of the action within the current plan. In this way, the Supervisor can know exactly what the user has already done and what actions he lacks to achieve his goal. This then generates a response from the LLM that is much more accurate.

Proof. The user is having lunch (the plan is composed of the actions Pick and Place Meal, Cook, Pick and Place Meal, Eat, Pick and Place Plate, Wash Plate). Information received by Supervisor Module:

- User position;
- Objects' names, positions, and QSRs;
- Observed actions: Pick and Place Meal, Eat;

Table 1. Comparison of the responses of the two architectures to certain sequences of actions. Percentages indicate the confidence level of predictions.

CASPER and CASPER-VE (no targets)			CASPER-VE	
Trial	**Observations**	**Outcomes**	**Observations**	**Outcomes**
1	Pick and Place Eat	Drink (23%) Breakfast (53%)	Pick and Place Biscuits Eat	Breakfast (100%) Breakfast (100%)
2	Pick and Place Sip	Drink (23%) Drink (100%)	Pick and Place Bottle Sip	Drink (100%) Drink (100%)
3	Pick and Place Cook	Drink (23%) Lunch (100%)	Pick and Place Meal Cook	Lunch (60%) Lunch (100%)
4	Pick and Place Pick and Place Eat	Drink (23%) Breakfast (28%) Lunch (100%)	Pick and Place Meal Pick and Place Meal Eat	Lunch (60%) Lunch (100%) Lunch (100%)
5	Pick and Place Pick and Place Pick and Place	Drink (23%) Breakfast (28%) Lunch (100%)	Pick and Place Meal Pick and Place Meal Pick and Place Plate	Lunch (60%) Lunch (100%) Lunch (100%)
6	Pick and Place Pick and Place Wash	Drink (23%) Breakfast (28%) Drink (30%)	Pick and Place Bottle Pick and Place Glass Wash Glass	Drink (100%) Drink (100%) Drink (100%)
7	Sip	Drink (100%)	Sip	Drink (100%)
8	Eat	Breakfast (70%)	Eat	Breakfast (70%)
9	Wash Cook	Drink (41%) unidentified	Wash Cook	Drink (41%) unidentified

– Predicted goal: Lunch with Pick and Place Meal at position 3 and Eat at position 4.

LLM's response: *It looks like you're having lunch, can I help you with the dishes when you're done eating?*

It is important to specify that the LLM is not the central mind or main reasoning engine of the architecture. It primarily serves as a communication interface between the cognitive architecture and the human user, fostering smoother and more intuitive communication based on information from previous modules. While the LLM can generate hints and trigger actions in the virtual environment, its understanding of the context and user's intentions is limited to the information from the underlying cognitive architecture. However, integrating the LLM offers a unique benefit: it can inform users about the architecture's reasoning process. This is like hearing the inner speech of the virtual agent, which can increase the human user's trust and acceptance, similar to interacting with a physical robot that expresses its reasoning. [8].

Another important feature of CASPER-VE is its easy adaptability to different virtual scenarios without substantial code modifications: each module is built with classes that have attributes and parameters easily modified from the Unity interface. This flexibility and code reuse are significant advantages, but further refinements are needed to ensure complete modularity and portability of the architecture.

5 Conclusions and future steps

In this paper, we presented a new implementation of the CASPER cognitive architecture, originally designed for social robotics, adapted to operate in virtual environments such as the Metaverse. By migrating from Python to C# and integrating with Unity, we successfully replicated essential CASPER functionalities. This adaptation empowers virtual agents to perceive their surroundings, recognize user actions and intentions, and engage with users interactively.

A notable enhancement in this version is the incorporation of an LLM within the Supervisor module, serving as a natural communication interface between the architecture and human users.

However, there are still critical issues and limitations to address. A key focus should be refining the architecture's modularity and portability to make it easily adaptable and "plug-and-play" for different virtual scenarios without major code changes.

Additionally, we aim to enhance the user experience by making interaction with the virtual agent more natural, intuitive, and user-friendly. This may involve integrating speech recognition, and text-to-speech capabilities, and exploring multimodal interaction modes involving user gestures and movements.

As mentioned earlier, we will apply this architecture to scenarios involving moral dilemmas, assessing the ability of specially trained LLMs to provide advice and support to users in ethically complex situations based on contextual awareness not directly provided by the user.

Furthermore, it is crucial to conduct rigorous evaluations and user experience studies to fully understand the impact of this technology and guide its future development. These evaluations could include usability testing, measurements of user immersion and engagement, and analysis of users' perceptions and expectations regarding contextualized virtual agents.

In conclusion, this is also one of the first native implementations of a cognitive architecture in Unity. We have made a GitHub repository of CASPER-VE available [1] to be freely cloned, modified, used, and improved to be useful in this field of research.

References

1. Gaglio, G.F.: Casper-ve github page (2024), https://github.com/gFulvio/CASPER-VE [Accessed: (26-06-2024)]
2. Gaglio, G.F., Augello, A., Pipitone, A., Gallo, L., Sorbello, R., Chella, A.: Moral Mediators in the Metaverse: Exploring Artificial Morality through a Talking Cricket Paradigm. In: CEUR Workshop. pp. 30–43. CEUR-WS, Rome, Italy (2023)
3. Gatto, L., Gaglio, G.F., Augello, A., Caggianese, G., Gallo, L., La Cascia, M.: MET-iquette: enabling virtual agents to have a social compliant behavior in the Metaverse. In: 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). pp. 394–401 (Oct 2022). https://doi.org/10.1109/SITIS57111.2022.00066

4. Gustavsson, O., Iovino, M., Styrud, J., Smith, C.: Combining Context Awareness and Planning to Learn Behavior Trees from Demonstration. In: 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). pp. 1153–1160. IEEE, Napoli, Italy (Aug 2022). https://doi.org/10.1109/RO-MAN53752.2022.9900603, https://ieeexplore.ieee.org/document/9900603/
5. Hasani, M.F., Udjaja, Y.: Immersive Experience with Non-Player Characters Dynamic Dialogue. In: 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI). pp. 418–421. IEEE, Jakarta, Indonesia (Oct 2021). https://doi.org/10.1109/ICCSAI53272.2021.9609725, https://ieeexplore.ieee.org/document/9609725/
6. Juliani, A., Berges, V.P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D.: Unity: A General Platform for Intelligent Agents (May 2020), http://arxiv.org/abs/1809.02627, arXiv:1809.02627 [cs, stat]
7. Kyrlitsias, C., Michael-Grigoriou, D.: Social Interaction With Agents and Avatars in Immersive Virtual Environments: A Survey. Frontiers in Virtual Reality **2**, 786665 (Jan 2022). https://doi.org/10.3389/frvir.2021.786665, https://www.frontiersin.org/articles/10.3389/frvir.2021.786665/full
8. Pipitone, A., Chella, A.: What robots want? Hearing the inner voice of a robot. iScience **24**(4), 102371 (Apr 2021). https://doi.org/10.1016/j.isci.2021.102371, https://linkinghub.elsevier.com/retrieve/pii/S2589004221003394
9. Ritterbusch, G.D., Teichmann, M.R.: Defining the Metaverse: A Systematic Literature Review. IEEE Access **11**, 12368–12377 (2023). https://doi.org/10.1109/ACCESS.2023.3241809, https://ieeexplore.ieee.org/document/10035386/
10. Salt, L., Wise, J., Sennersten, C., Lindley, C.A.: REACT-R and Unity Integration. In: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). p. 31 (2016)
11. Smart, P., Scutt, T., Sycara, K., Shadbolt, N.: Integrating ACT-R Cognitive Models with the Unity Game Engine. In: Integrating cognitive architectures into virtual character design, pp. 35–64. IGI Global (2016)
12. Vinanzi, S., Cangelosi, A.: CASPER: Cognitive Architecture for Social Perception and Engagement in Robots. International Journal of Social Robotics (Mar 2024). https://doi.org/10.1007/s12369-024-01116-2, https://link.springer.com/10.1007/s12369-024-01116-2
13. Wang, H., Ning, H., Lin, Y., Wang, W., Dhelim, S., Farha, F., Ding, J., Daneshmand, M.: A Survey on the Metaverse: The State-of-the-Art, Technologies, Applications, and Challenges. IEEE Internet of Things Journal **10**(16), 14671–14688 (Aug 2023). https://doi.org/10.1109/JIOT.2023.3278329, https://ieeexplore.ieee.org/document/10130406/
14. Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W.X., Wei, Z., Wen, J.: A survey on large language model based autonomous agents. Frontiers of Computer Science **18**(6), 186345 (Dec 2024). https://doi.org/10.1007/s11704-024-40231-1, https://link.springer.com/10.1007/s11704-024-40231-1
15. Zhou, M.X., Mark, G., Li, J., Yang, H.: Trusting Virtual Agents: The Effect of Personality. ACM Transactions on Interactive Intelligent Systems **9**(2-3), 1–36 (Sep 2019). https://doi.org/10.1145/3232077, https://dl.acm.org/doi/10.1145/3232077