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 work presents the development of a virtual agent designed specifically for use in the Metaverse, video games, and other virtual environments, capable of performing intention reading on a human-controlled avatar through a cognitive architecture that endows it with contextual awareness. The paper explores the adaptation of a cognitive architecture, originally developed for physical robots, to a fully virtual context, where it is integrated with a Large Language Model to create highly communicative virtual assistants. Although this work primarily focuses on virtual applications, integrating cognitive architectures with LLMs marks a significant step toward creating collaborative artificial agents capable of providing meaningful support by deeply understanding context and user intentions in digital environments.

Keywords: Metaverse · Virtual Agent · Cognitive Architecture · Large Language Model · Intention Reading.

1 Introduction

The Metaverse will become an increasingly integral part of our lives, offering a decentralized, immersive, and persistent 3D online environment where users, represented by avatars, can engage in creative and collaborative interactions [9]. These interactions encompass human users and AI-driven non-player characters (NPCs), whose realism in appearance and behavior markedly enhances the user experience. Research in artificial intelligence and cognitive science concentrates on developing cognitive capabilities and autonomy for NPCs. In light of recent advances in AI, particularly those about large language models (LLMs), it seems inevitable that AI-controlled avatars will assume a significant role in the Metaverse, particularly in the context of service-oriented activities [7, 5]. Chatbots based on LLM technology demonstrate potential as autonomous agents due to their capacity to simulate human intelligence. Recent advancements have enhanced their utility as virtual assistants, motivating ongoing research to augment their cognitive capabilities, including memory and action planning, with an emphasis on context awareness [15].

The use of cognitive architectures in virtual environments is a prevalent methodology for imparting context awareness and autonomy to virtual agents. Game engines such as Unity and Unreal Engine are well-suited for the development of the Metaverse and other interactive virtual environments so several studies have integrated cognitive architectures like SOAR or ACT-R with these platforms to create autonomous virtual agents [13, 11, 6], though this often necessitated intricate adaptations. The integration of a native cognitive architecture within a single development environment would offer substantial advantages, fully utilizing the engine's capabilities and circumventing compatibility issues, thereby simplifying the creation and maintenance of complex applications.

This paper outlines the implementation of a virtual agent utilizing the "Cognitive Architecture for Social Perception and Engagement in Robots" (CASPER) [14], adapted for Unity. CASPER was created to provide real-world robotic assistance. However, a virtual version has the potential to apply to several other contexts, including the Metaverse, video games, and virtual reality. The new system has been augmented by integrating it with an LLM, which is tasked with communicating in discursive form the operations performed by the cognitive architecture in recognizing the user's intentions. This approach enables the agent to communicate in a more human-like manner, thereby significantly improving the quality of the interaction.

The paper is structured as follows: Section 2 will describe the motivations behind this work followed by a description of the CASPER architecture; 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 performances; Section 5 will present the conclusions and envision future developments.

2 Background

This work is part of a larger research project aimed at developing virtual moral agents capable of guiding users within the Metaverse. In the preceding phase, LLM-based virtual agents were developed, but their contextual knowledge was entered manually via prompts [4]. Therefore, a cognitive architecture was considered to serve as a bridge between the operating environment and the LLM. The environment and virtual agents were developed in Unity, selected for its flexibility, power, and ease of use in Metaverse development [8], as well as its comprehensive library of assets and plugins and its large community of developers.

Several relevant cognitive architectures already interface with virtual reality and have great potential. However, these architectures often use more than one programming language, such as Prolog, Common LISP, C++, and Python, making them complex and less accessible to the programming community [1, 12]. Some of them also depend heavily on ROS to function [10], which adds further complexity. Bringing these architectures into C# would have been very complicated, requiring not only a complete rewrite of the code but also the adaptation of features closely related to physical robotics that are irrelevant in the context of the Metaverse. CASPER, on the other hand, is written in only one language, Python, and was born and already tested in a virtual environment, that of Webots, making it much more suitable to be ported to Unity natively and without excessive complications.

2.1 Casper Architecture and the Qualitative Spatial Reasoning

CASPER is a platform-independent cognitive architecture designed to enable robots to perform intention reading (IR) and collaborative behavior in Human-Robot Interaction (HRI) scenarios. It employs a combination of symbolic and data-driven artificial intelligence methodologies, emphasizing the use of Qualitative Spatial Relations (QSRs) to predict the actions and intentions of a human partner and to calculate an optimal collaborative behavior. QSRs are abstract representations that qualitatively describe spatial relationships between entities, rather than relying on precise quantitative measurements. Instead of measuring the exact distance between two objects, a QSR might categorize the objects as "near" or "far." This approach aligns with how humans typically perceive spatial relationships and allows for efficient and generalizable reasoning about space. QSRs enable the robot to understand the movements and interactions of a human partner relative to objects in the environment without needing precise, continuous measurements, thus supporting scalable and context-independent intention recognition.

CASPER's architecture comprises several key modules that perform perception, reasoning, and action planning. Below is a concise overview of each module. As the objective of this paper is not to present CASPER but rather a specific use case, we strongly encourage readers to consult the dedicated paper for a deep understanding of the employed algorithms and models.

- **Perception Module**: it converts visual observations from the robot's sensors into QSRs. The QSR descriptors include Qualitative Distance Calculus (QDC), Qualitative Trajectory Calculus (QTC), Moving or Stationary (MOS), and Holding Object (HOLD). These descriptors are computed using the QSRlib library, an open-source tool that processes spatial data into qualitative relations [2].
- **Low-Level Action Recognition**: it performs bottom-up inference, identifying and aggregating the human's movements into actions. It comprises three submodules:
  - *Focus Estimator*: calculates an attention score for each Object of Interest (OOI) using a probabilistic model incorporating QDC, QTC, and gaze direction. The OOI with the highest score, exceeding a threshold, is identified as the target.
  - *Movement Classifier*: a decision tree model is employed to map QSRs to specific movements like "Still," "Walk," "Pick," etc. This model is trained on labeled data collected from simulated environments.
  - *Action Predictor*: actions are predicted using a Markov-chain Finite State Machine (FSM) ensemble, which maps sequences of movements to higher-level actions such as "Pick and Place" or "Use."

- **High-Level Goal Prediction**: it uses a Plan Library that models goals as non-binary trees of actions and sub-goals. The intention reading process involves matching observed actions to potential plans in the library. The high-level goal is predicted by selecting the plan with the highest score, balancing between observed and unobserved actions.
- **Verification Module**: it acts as a filter, ensuring that the predictions from the Low-Level and High-Level modules are logically consistent with the known properties of the world, stored in an ontology. The ontology is implemented using OWL2, with the Pellet reasoner verifying each inferred action or goal.
- **Collaborative Intelligence (Supervisor Module)**: it integrates information from other components to generate a collaborative plan. Based on the validated explanations from the High-Level module, it determines which part of the goal remains unachieved and which tasks can be assigned to the robot.

3 Methods

This section describes the process of converting CASPER from the Webots and Python environment to that of Unity and C#. We will call this new version CASPER for Metaverse (CASPER-MV) to distinguish it from the original version, called simply CASPER. Although it is difficult to imagine someone pretending to cook and eat in the Metaverse, we decided to recreate the same scenario used for the CASPER experiments, i.e., an agent observing a human-controlled avatar moving around a kitchen to have breakfast, lunch, and drink. This choice was driven by the need to have an immediate example to follow and a quick comparison. As can be seen, the moral aspects mentioned above have been completely ignored at this stage and will be taken up in a later work. All CASPER modules have been implemented in CASPER-MV as C# classes.

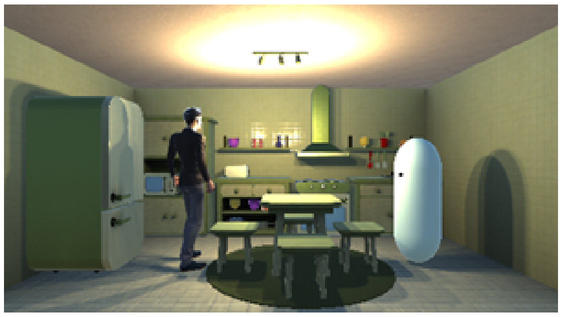

Fig. 1. Game view of the experimental environment. The grey capsule on the right represents the agent.

3.1 Objects detection

In contrast to the physical world, where robots recognize objects visually, a virtual agent in a virtual environment can directly access encoded information about objects, such as size, location, material, and name. The Unity development environment offers several methods to achieve this result. To avoid momentary focus on a single scene area we used the one called Overlap Sphere, which detects objects within an invisible sphere. The sphere is centered on the agent's body, represented by a simple capsule, and the radius is been set to 2 meters. The LayerMask parameter of Overlap Sphere allows filtering the objects to be detected, limiting the search to objects belonging, in our case, to the "OOI" layer (see Fig2). Along with objects, the user's location is also detected.

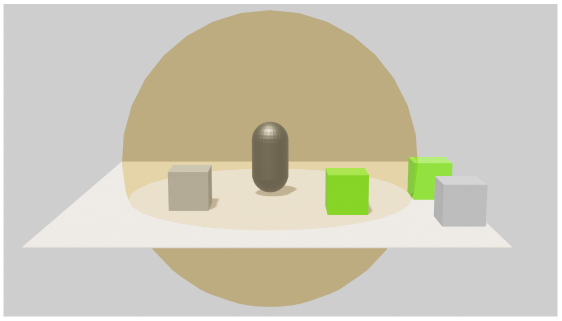

Fig. 2. Schematic of Unity's Overlap Sphere. Objects in the OOI layer that touch or are inside the sphere are detected (highlighted in green), while others are ignored.

3.2 Perception

Given the absence of dedicated libraries for QSRs in C#, it was necessary to program a QSR Engine for QSR calculations. The functions for calculating QDC, QTC, MOS, and HOLD on the data obtained from the Overlap Sphere were created using the built-in vector calculation functions provided by Unity.

The QDC values were calculated by determining the distance between the user's position vector and the OOI's position vector. The function returns one of five strings, depending on the value obtained: "IGNORING," "FAR," "MEDIUM," "NEAR," and "TOUCH."

The QTC values are derived from the vector product between the user and the OOI position. A positive value returns "APPROACHING" while a negative value returns " LEAVING".

The MOS was calculated by determining the distance between the position vectors of two consecutive user positions. If a non-zero value is returned, it is converted to "MOVING." Otherwise, it is converted to "STATIONARY."

Ultimately, the HOLD value is determined by assessing the distance between the user's hand position vector and the OOI position vector. A value approaching zero indicates that the hand and the object are in the same position, thereby yielding a "YES" result; otherwise, a "NO" result is obtained.

The resulting computations are stored in a C# dictionary (list of key-value pairs) called QSRLibrary, and ordered by time. In particular, the keys are the names of the OOIs, while the values are dictionaries comprising the names of the QSRs and their respective values in strings.

3.3 Low-Level

The Low-level module retained a high degree of similarity to that of CASPER, largely due to the availability of libraries that corresponded to those of Python within the C# NuGet package manager. Its submodules are here implemented as methods and they draw data from the QSRLibrary and execute the requisite calculations.

To obtain the target OOI corresponding to the user's action, the FocusEstimator converts the string values of QDC and QTC into numerical values between 0.00 and 0.5. It then applies a probabilistic calculation to these values, taking into account the user's orientation, to obtain a score for each OOI. The OOI with the highest score is designated as the target.

The MovementClassifier employs a DecisionTree created with ModelBuilder of ML.Net. It is trained on a dataset of 300 random combinations of QSRs generated through the user's movement in 3D space. Subsequently, the labels "Pick," "Place," "Still," "Walk," and "Transport" are manually associated. The movements identified by the decision tree are entered as strings into a list that is read by the action predictor.

ActionPredictor employs a Markov-chain FSM ensemble as in CASPER. Each chain within this ensemble describes an action. To illustrate, the "Pick and Place" action is defined by the string chain of movements "Pick Transport Place." Each chain is then compared with the list of observed movements to identify the action the user is performing. Subsequently, a list of strings is generated, wherein the predicted actions are entered. A salient feature of this novel implementation is that recognized actions are linked to their respective target objects before being stored in the list.

3.4 High-Level

In the High-Level module, one of the numerous goal planners available in the NuGet Gallery was utilized. Once the initial states, goal states, subgoals, and actions have been defined, the goal planner generates a plan for each goal. Subsequently, a function transforms each plan into a list of strings, denoting node names, and inserts it into a dictionary. The goal name is designated as the key, thereby generating a Goal Library. Each plan within the Goal Library is compared with the sequence of observed actions derived from the Low-level module.

This comparison determines the user's most probable goal, based on the percentage of observed versus unobserved nodes.

3.5 Supervisor Module

In CASPER, once the user's goal is recognized, the Supervisor Module leverages a goal planner to plan how to assist the user. In CASPER-MV we chose instead 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.

The incorporation of the LLM into the module was achieved through the Microsoft Azure API, which facilitates access to a GPT-4 model and is entirely compatible with Unity. An initial prompt instructs the model on how to behave as if it were a role-playing game:

"""You are part of a cognitive architecture for a virtual agent that observes a human-controlled avatar. The architecture consists of four modules. You are part of the final module and receive information from the other modules about the environment, the user, and the user's goal. You have to ask if the goal is indeed the one predicted and propose a way to help him taking into account what is present in the environment."""

All data from the previous modules (user position, target, OOIs, current action, and user goal) are entered into a variable-based sentence that is then transmitted to the LLM.

4 Performance Test and Discussion

The performance tests previously conducted with CASPER were repeated with CASPER-MV yielding results that closely mirror those reported in the original study, with some notable improvements in the performances. For convenience and considering that this would not have affected the agent's perception, the experiment was conducted using a desktop and keyboard to control the human avatar rather than a VR kit. The results of the predictions generated by the various modules of the system are displayed in a textual format directly on the screen. This setup could readily be adapted for use in a VR context. The OOIs included the meal, plate, microwave, biscuits, sink, bottle, and glass. The human-controlled avatar was directed to complete a series of tasks: Breakfast (Pick and Place Biscuits, Eat, Pick and Place Plate, Wash), Lunch (Pick and Place Meal, Cook, Pick and Place Meal, Eat, Pick and Place Plate, Wash), and Drink (Pick and Place Bottle, Drink, Pick and Place Glass, Sink).

The Focus Estimator demonstrated a high degree of accuracy in identifying the target object. However, some challenges were observed, particularly when the human avatar was engaged in eating, where both the plate and the meal were within its line of sight, and when washing the plate, where both the plate and the sink represented potential targets. In these instances, the Focus Estimator occasionally misidentified the target, particularly when the gaze direction was unclear or slightly off-center. The estimator is dependent on exceeding a threshold value of 0.5 to confirm a target. This threshold was typically only exceeded when the user's gaze was perfectly aligned with the object, which highlights the system's reliance on precise gaze alignment for accurate target identification.

The Movement Classifier, trained using ML.Net's Model Builder, demonstrated robust performance, with an average accuracy of 94% in recognizing the avatar's movements. During the training process, Model Builder explored 533 distinct models over a total experimental duration of 58 seconds. The top-performing models, including the LbfgsLogisticRegressionOva and FastForestOva trainers, demonstrated a macro accuracy of 1.0000. In conclusion, the FastForestOva model was chosen due to its optimal balance of accuracy and performance, with a duration of 0.323 seconds. The classifier effectively mapped the observed QSRs to discrete movement categories, thereby enabling the system to accurately track the avatar's actions within the environment.

In action recognition, the FSMs were assigned generating three sequences of nine movements, following the methodology employed in the CASPER test. Each FSM was assigned a score based on the Ratcliff-Obershelp pattern recognition algorithm, with scores ranging from 0 to 1. A threshold of 0.8 was used to determine when an FSM had sufficiently matched the observed sequence of movements, allowing for the prediction of the ongoing action. The accuracy of the algorithm was consistently high, achieving maximum precision in all cases, thereby ensuring reliable and real-time action classification.

The Goal Reasoner, which associates actions with specific targets (e.g., "Pick and Place Biscuits" instead of a generic "Pick and Place"), has markedly enhanced the system's capacity to accurately infer the intended goals of the user. In the absence of consideration of the target, as in CASPER, the generation of actions frequently resulted in considerable ambiguity when the number of observations was limited. However, the incorporation of target consideration enabled the system to attain 100% certainty in goal recognition, even with the observation of a single action. Furthermore, this allows for the identification of the specific stage of the plan at which the current action is situated.

In the Supervisor Module, the LLM model successfully processes the information and responds following the initial prompt with a message on the screen. The following is an illustration of the model's response after the identification of the objective, "Lunch", following the observation of the actions "Pick and Place Meal" and "Eat": *"It looks like you're having lunch, can I help you with the dishes when you're done eating?"*

The LLM also proved useful as a filter for the operations performed by the low-level and high-level. Therefore, instead of implementing a standalone verification module, it was decided to use the knowledge base of the model itself. As can be noted, the component of the CASPER Supervisor Module designated for the agent's interaction with the environment to assist the user has not been implemented in CASPER-MV. Nevertheless, the deployment of an LLM presents an intriguing opportunity in this regard, particularly in light of the Function Calling functionality offered by the Azure OpenAI Service. This feature enables language models to interact with user-defined functions, thereby expanding the potential applications of such models. However, this implementation is specific to the particular scenario and thus left for others to implement freely.

In conclusion, the adaptation of CASPER-MV to a different virtual scenario can be done without significant changes to the code. Each module is based on classes with attributes and parameters accessible from the Unity interface. These characteristics of flexibility and code reuse are significant advantages, but further refinements are necessary to ensure the full modularity and portability of the architecture.

5 Conclusions and future steps

We presented CASPER-MV, a repurposing for Unity of the CASPER architecture to create virtual agents in the Metaverse capable of intention reading. For this purpose, the architecture was enriched with an LLM.

However, key issues need further attention. It is essential to enhance the modularity and portability of the architectural framework to guarantee its adaptability to diverse virtual scenarios without substantial code modifications. Improving user interaction to be more natural and intuitive through speech recognition, text-to-speech, and multimodal interactions, is also a priority. Furthermore, rigorous evaluations and user experience studies must be conducted to ascertain the technology's impact and to inform future development.

The CASPER-MV GitHub repository is freely available for use, modification, and improvement in this research field [3].

Author Contributions G.F.G. developed CASPER-MV, performed the experiments, and wrote the manuscript. S.V. provided valuable support in understanding the technical details of the CASPER architecture, which he developed, and directly supervised this work. A.C. and A.Ch. supervised the research and provided guidance and valuable insights.

Acknowledgments. This work is supported by the 2022 European PRIN project "AlterEGO: how to emulate intentionality and awareness in remote communications by means of software surrogates". For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising from this submission.

References

1. Beetz, M., Tenorth, M., Winkler, J.: Open-EASE. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 1983–1990. IEEE, Seattle, WA, USA (May 2015). https://doi.org/10.1109/ICRA.2015.7139458, http://ieeexplore.ieee.org/document/7139458/
2. Cohn, A., Burbridge, C., Hogg, D., Alomari, M., Hawes, N., Duckworth, P., Lightbody, P., Gatsoulis, Y., Dondrup, C., Hanheide, M.: Qsrlib: a software library for online acquisition of qualitative spatial relations from video. In: Workshop on qualitative reasoning (QR16), at IJCAI-16 (2016)

3. Gaglio, G.F.: Casper-mv github page (2024), https://github.com/gFulvio/CASPER-MV [Accessed: (26-08-2024)]
4. 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)
5. 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
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. Lee, G.I., Han, S.H., Lee, Y.H.: Implementation of metaverse virtual world using unity game engine. Journal of the Semiconductor & Display Technology **22**(2), 120–127 (2023)
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. Ros, Y., Lemaignan, S., Ferrini, L., Andriella, A., Irisarri, A.: ROS4HRI: Standardising an Interface for Human-Robot Interaction. In: HRI 2023 Workshop. pp. 3020–3027. IEEE, Stockholm, Sweden (Mar 2023). https://doi.org/10.1109/IROS51168.2021.9636816, https://ieeexplore.ieee.org/document/9636816/
11. 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)
12. Sarthou, G.: Overworld: Assessing the Geometry of the World for Human-Robot Interaction. IEEE Robotics and Automation Letters **8**(3), 1874–1880 (Mar 2023). https://doi.org/10.1109/LRA.2023.3238891, https://ieeexplore.ieee.org/document/10024303/
13. 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)
14. 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
15. 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