Integrating Computer Vision with Embedded Intelligent Agents

Vinicius Wolosky Muchulski
Federal University of Santa Catarina - UFSC
vinicius.muchulski@grad.ufsc.br

Italo Firmino da Silva
Federal University of Santa Catarina - UFSC
italo.silva@grad.ufsc.br

Bernardo Pandolfi Costa
Federal University of Santa Catarina - UFSC
bernardo.pandolfi.costa@grad.ufsc.br

Heitor Henrique da Silva
State University of Campinas - UNICAMP
heiitorheenrique@gmail.com

Roberto Rodrigues-Filho
Federal University of Santa Catarina - UFSC
roberto.filho@ufsc.br

Antonio Carlos Sobieranski
Federal University of Santa Catarina - UFSC
a.sobieranski@ufsc.br

Alison R Panisson
Federal University of Santa Catarina - UFSC
alison.panisson@ufsc.br

Abstract

In the current technological landscape, we are transitioning to a lifestyle shaped and supported by advanced artificial intelligence systems. In this context, there is a growing tendency to decentralize such systems, moving them to the edge, and placing more responsibility on the devices that compose these systems. The field of Multi-Agent Systems offers a powerful paradigm for distributed artificial intelligence, where intelligent agents can incorporate a variety of AI techniques to make autonomous decisions. In this work, we propose an approach for implementing embedded autonomous intelligent agents. We propose an architecture to implement autonomous intelligent agents embedded in simple microcontrollers, and as a case study, we describe the integration of computer vision, in which agents are capable of extracting information from computer vision components and utilizing this data in their decision-making processes.

Keywords

Artificial Intelligence, Embedded Intelligent Agents, Computer Vision

1 Introduction

Technological development has transformed society through historical contexts linked to industrialization and, more recently, by shaping a society immersed in increasingly sophisticated technologies present in nearly all aspects of daily life [1]. In the same context, significant advances in autonomous behaviour technologies (such as autonomous vehicles and systems) have been driven by developments in Artificial Intelligence (AI). These advances have enabled impactful technologies, both in the context of everyday technological immersion and in advanced industrial reorganization processes, known as Industry 4.0 [2].

A research area investigating intelligent autonomous technologies is the field of intelligent autonomous agents. By employing multiple intelligent autonomous entities (agents) and enabling their interactions, the concept of Multi-Agent Systems (MAS) emerges. MAS provide a paradigm for implementing systems with distributed intelligent autonomous entities (distributed AI), where collaborative and coordinated behaviours emerge to solve complex problems [3]. Multi-agent systems show strong synergy with the development of the next-generation internet, Web 3.0, and emerging technologies built on the Internet of Things (IoT) [4] and the Web of Things (WoT) [5]. With a growing number of devices connected to the internet, vast opportunities for distributed artificial intelligence can be explored, particularly through multi-agent systems in developing Ambient Intelligence and Smart Environments [6], such as smart cities, smart homes, and smart buildings.

In this context, there is a promising direction toward embedding well-established architectures for implementing multi-agent systems that enable sophisticated reasoning and decision-making mechanisms in devices with lower computational power. These devices, though computationally constrained, offer lower development costs, scalability, and low energy requirements. Moreover, incorporating the abstraction of agents into these devices allows us to envision systems architectures with AI at the edge [7], as various AI techniques can be integrated into these embedded intelligent agents, while also recognizing that MAS provides a powerful paradigm for distributed systems.

In this paper, we propose an approach for implementing embedded intelligent agents, comprising an architecture for embedded agents and interfaces for integrating AI techniques. As a case study, we present an embedded intelligent agent equipped with computer vision capabilities. The agent leverages spatial recognition of a person's face within the camera's capture area to make decisions about where to look, guided by its profile. Notably, the agent's profile can be dynamically adjusted, allowing it to transition, for example, from a *shy* profile to a *confident* one, which changes entirely the agent's behaviour. The proposed approach offers several benefits, including enhanced adaptability and personalization of agent behaviour in dynamic environments. By embedding intelligence in resource-constrained devices, it enables real-time decision-making without relying on centralized processing, thereby reducing latency and improving responsiveness. Furthermore, the use of well-defined interfaces allows for seamless integration and combination of additional AI techniques and/or components, extending the system's capabilities and versatility. This approach supports a wide range of applications, from human-machine interaction to robotics, by equipping agents with the ability to adapt their behaviour to context-specific requirements.

2 Background

2.1 Computer Vision

Currently, computer vision systems have become an increasingly integral part of our daily lives. This field of computational research has gained significant attention in recent years, particularly due to the wide range of technological applications that impact society, spanning from social networks to healthcare, such as the more accurate diagnosis of diseases [8, 9].

With this in mind, computer vision is a branch of artificial intelligence that aims to enable computer systems to interpret and understand the world in a manner similar to humans, such as by detecting people, objects, and patterns in images and videos [9]. For this process to function, the computer must decompose an image into a set of pixels, with each pixel representing the smallest unit of information in a given photo or frame. From this, each pixel is associated with a numerical value, which is then used by algorithms for visual pattern recognition [10].

Among the techniques used, we can mention classical computer vision, which emerged before the popularization of machine learning and is primarily based on statistical and mathematical methods applied to data extracted from images. Furthermore, many machine learning processes have emerged over the past few decades that enable the development of systems that automatically extract patterns from large volumes of data, such as the case of Convolutional Neural Networks (CNNs) [11].

2.2 Embedded Intelligent Agents

Advances in embedded computing and wireless technologies are driving the integration of physical objects with the internet, creating a technological landscape of connected devices known as the Internet of Things (IoT) [4] (recently expanded to the idea of Web of Things [5]). This evolving scenario, coupled with concurrent developments in artificial intelligence technologies and methodologies, is fostering an ecosystem of "smart" devices and sensors, establishing paradigms in intelligent edge computing [12, 13] and ambient-intelligence/smart environments [14, 15].

These paradigms incorporate characteristics of ubiquity, transparency, and intelligence, in which the decision-making processes are shifted towards devices. Devices within these systems require autonomous capabilities, enabling them to manage tasks and make decisions in real time [16]. In this context, autonomous intelligent agents perfectly fit into these paradigms, abstracting such smart devices as agents, and resulting in the conceptualization of systems based on multi-agent systems [3].

Thus, embedded intelligent agents, or embedded agents, refer to embedded computer systems with some capacity for reasoning, planning, and learning [14, 17]. Embedded agents typically have a network connection, enabling communication and cooperation with other embedded agents [14].

The literature has different views on the integration of embedded agents with resource-limited IoT devices: (i) decoupling agents from IoT devices and then creating a representative agent responsible to interact with such device using a M2M protocol such as MQTT, examples are [18, 19]; (ii) embedded agents executing directly in the IoT devices, normally using an agent platform that facilitates the agent interactions, examples are [20, 21].

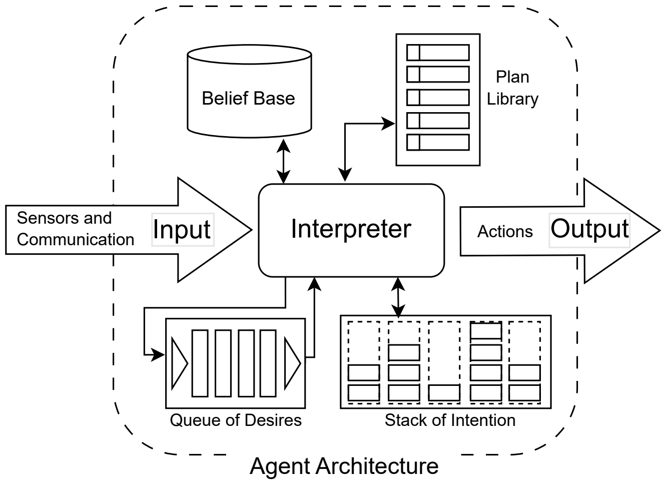

Figure 1: PRS-BDI-Based Agent Architecture.

3 Embedded Intelligent Agents Enhanced by Computer Vision

In this work, we propose an approach for embedded agents, in which agents are directly integrated into 'smart' devices, making them autonomous, reactive entities capable of proactively achieving design goals. The proposed agent architecture incorporates dynamic components from the BDI (Belief-Desire-Intention) framework [22] and the PRS (Procedural Reasoning System) [23], including a plan library that allows for dynamic addition of new plans, thereby expanding the range of tasks the agent can perform. In Section 3.1, we will detail the proposed architecture.

To demonstrate our approach for embedded intelligent agents, we implement a device enhanced by a computer vision interface. The intelligent agent can extract semantic information from the video stream and use this data in its decision-making processes. In Section 3.2, we will describe the implemented scenario.

We follow the same direction pointed out by the authors in [15], who assert that the integration of agents into devices ushers in a new generation of smarter, more collaborative, and flexible devices. Additionally, this integration enhances their adaptability and responsiveness to the surrounding environment and to the presence of other embedded agents within the same network.

3.1 Proposed Agent Architecture

We propose an adapted architecture based on the BDI model [22] and the PRS [23] to embedded intelligent agents. This architecture keeps the main components of BDI-PRS-based agents, but it simplifies some components to run on devices with less computational power. An overview of the proposed architecture is shown in Figure 1.

While the BDI architecture offers an elegant orchestration of dynamic mental components for intelligent agents, the PRS provides a framework for constructing real-time reasoning systems capable of performing complex tasks in dynamic environments. It combines the mental components from BDI with a well-established process of practical reasoning, incorporating reactive capabilities (such as events triggered by sensors or communication) into the agents.

The formal components that have been considered and integrated into the proposed agent architecture are outlined as follows:

- E: a set of events $\{e_i, \ldots, e_n\}$, which represents the input for the agent. Events aggregate both sensor input and receiving messages, implementing a generic input interface for agents;
- B: a set of beliefs $\{b_i, \ldots, b_n\}$ that represents the knowledge of the agent. Beliefs reflect the information the agents has about the world and other agents.
- D: a set of desires (or goals) the agent is motivated to pursuit.
- I: a set of intentions, representing those goals an agent has committed to pursuit.
- A: a set of actions $\{a_i, \ldots, a_n\}$ available to the agent execute in the environment. These actions are implemented by the action interface component, representing the actions available at the devices the agent is embedded. When relevant to the application, the set of actions will include actions for communication, which follow the literature of speech acts [24]. Actions represent the output interface to the agent.
- ps: a plan library, containing a set of plans $\{p_1, \ldots, p_n\}$ available to the agent. Plans represent the know haw of the agent, and they can be dynamically updated in the proposed architecture.

The pseudo-code in Algorithm 1 outlines the proposed reasoning cycle for embedded agents. The proposed reasoning cycle is based on the idea that embedded agents will operate in dynamic environments, and they will require a continuously updating of knowledge, then acting based on its beliefs, desires, and intentions.

In the pseudo-code in Algorithm 1, at the initialization phase, the agent starts its execution by creating an empty queue of events E, considering no event has been perceived or communicated yet; a predefined plan library ps, which represents a set of plans that the agent can use to achieve its goals; an initial belief base B, containing the agent's knowledge about itself and about the world; a queue of desires D, representing the initial goals or objectives the agent aims to achieve; and an empty stack I of actions and sub-goals representing the agent's intentions, considering the agent has not yet planned how to achieve its goals and therefore has no intentions at this stage.

After, the agent begins the continuous reasoning cycle, which is divided into event perception, updating beliefs, handling intentions, updating desires, selecting the next goals, and planning.

- *Event Perception*: at the event perception, the agent gathers new information through the $perception()$ function, which serves as an interface to the external world, enabling it to sense its environment, and the $communication()$ function, which provides a communication interface that agents can use to exchange messages.
- *Belief Update*: the agent updates its belief base B to reflect the new events E, according to some belief update function $update()$. This process ensures the agent's knowledge stays consistent with the latest observations or communications.
- *Handling Intentions*: when the agent has ongoing intention, i.e., $I \neq \emptyset$, the agent retrieves the next action to be executed from the intention stack, executing it through the function $execute()$, which implements an interface to external components, such as actuators or communication interfaces. After executing that action, the agent updates the intention queue removing the executed action.
- *Updating Desires*: the agent update its desires D based on the new events E according to some desire update function $update()$. This allows the implementation of methods to ensure the agents goals are still relevant, as well as creating new goals according the events perceived.
- *Selecting Next Desire*: the agent selects the next goal τ from the updated desires D. The function $next()$ allows us to implement mechanism in which an agent verify the priority of its goals, selecting those with higher priority to pursuit.
- *Planning*: The agent selects a plan from the plan library ps to achieve the goal it has decided to pursue, i.e., τ. The agent evaluates the applicability of plans based on their context (i.e., preconditions for executing the plan) by comparing the

```
/* initialisation                                        */
$E \leftarrow \emptyset$ ;                        /* empty events */
$ps \leftarrow \{p_1, \ldots, p_n\}$ ;            /* plan library */
$B \leftarrow \{b_1, \ldots, b_n\}$ ;             /* belief base */
$D \leftarrow \{d_1, \ldots, d_n\}$ ;                 /* desires */
$I \leftarrow \emptyset$ ;                    /* empty intentions */
while true do
    $E \leftarrow perception() \cup communication()$ ;    /* update events */
    $B \leftarrow update(B, E)$ ;              /* update beliefs */
    if $I \neq \emptyset$ then
        $\alpha \leftarrow next(I)$ ;    /* retrieve the next action from the intention */
        $execute(\alpha)$ ; /* execute the next action from the intention */
        $I \leftarrow I \setminus \alpha$ ;    /* remove the action from the intention */
    end
    $D \leftarrow update(D, E)$ ;  /* update desires according the events */
    $\tau \leftarrow next(D)$ ;    /* retrieve the next desire */
    $I \leftarrow I \cup plan(\tau, ps, B)$ ;  /* retrieve the sequence of actions for $\tau$ */
    $D \leftarrow D \setminus \tau$ ;             /* remove desire $\tau$ */
end
```

Algorithm 1: Pseudo Algorithm for the Reasoning Cycle of the Embedded Agent.

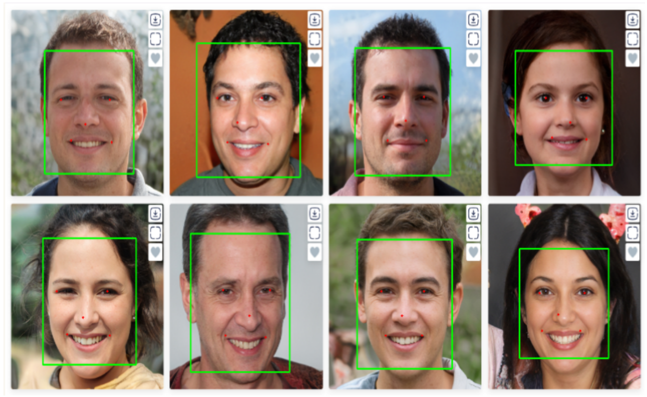

Figure 2: Face Detection using Yunet.

plans' contextual requirements with its belief base *B*. Once an applicable plan is identified, the sequence of actions and sub-goals associated with that plan are added to the intention stack. At this point, the reasoning cycle concludes.

Some relevant characteristics are inherent in the proposed reasoning cycle. First, the agent continuously revises its beliefs, desires, and intentions in response to new events, demonstrating adaptability to dynamic environments. Second, the agent prioritizes existing intentions by addressing them before selecting new desires, thereby maintaining a sense of commitment and continuity in its execution. Additionally, by incorporating plans from the plan library, agents can effectively exchange their know-how (plans), enhancing their collaborative and knowledge-sharing capabilities, making them highly adaptable.

3.2 Integrating Computer Vision

One of the objectives of this work is to demonstrate the incorporation of computer vision with embedded agents. In this context, the field of computer vision involves equipping computers with the ability to recognize images and the environment around them.

For the computer vision component, several face detection models can be used. In our experiments, we evaluated two models. The first was the Viola-Jones algorithm [25], a classical computer vision technique that was highly efficient for its time. The second model evaluated was the Yunet [26], developed in 2018, which is a neural network trained to detect faces more efficiently.

After some experiments, we chose Yunet over Viola-Jones for two reasons: it is lighter in computational processing and demonstrates significantly higher effectiveness compared to the traditional model. The Yunet model was developed with the intention of being used on IoT devices with low computational power, while still achieving performance similar to larger models that require more processing resources[1]. Figure 2 shows an example of the output provided by the Yunet face detection algorithm.

The computer vision interface developed uses an ESP32-Cam to capture the images to be processed. This camera was chosen due to its low cost and ease of integration into larger systems. Then, the images are transmitted via a local Wi-Fi network to a Raspberry

[1]Yunet is known for achieving an accuracy of 81% in WIDER FACE validation tests [26].

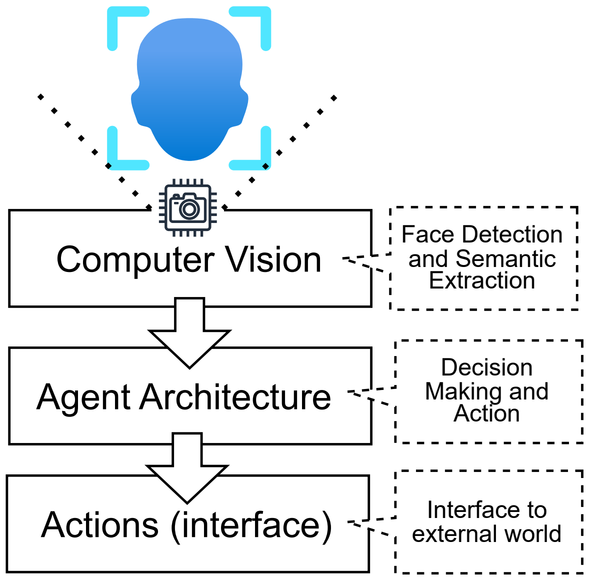
Computer Vision
Face Detection and Semantic Extraction
Agent Architecture
Decision Making and Action
Actions (interface)
Interface to external world

Figure 3: System Workflow.

Pi 3B, which functions as the central server of the operation. Real-time images are received by this computer, which uses Python code along with the OpenCV library to run the Yunet model on each frame and the agent architecture to make decisions about those inputs.

With the face detection performed by Yunet implementation[2], we developed a simple semantic extraction (sub)component which extracts the position of the faces in the image, providing such information to the embedded agent. This workflow is illustrated in Figure 3.

3.3 Implementation

The proposed agent architecture was implemented in Python[3], running within a Raspberry Pi 3B. The mental components of the agent were implemented using straightforward and effective data structures to represent its internal reasoning processes. The belief base was implemented as a simple list, which stores the agent's current knowledge about the environment. This structure allows for efficient addition and retrieval of beliefs, enabling the agent to dynamically update its understanding as new information is acquired. The agent's desires, representing its potential goals, were organized using a queue. This approach ensures that desires are managed in a first-in, first-out manner, reflecting a natural priority of objectives.

To manage the agent's intentions, a stack data structure was employed, enabling a last-in, first-out execution order. This design facilitates an elegant handling of goal decomposition into sequences of actions. When a goal or desire is selected from the intention stack for processing, the agent searches for

[2]The Younet implementation is available in https://github.com/LAIA-UFSC/SD2-Project/tree/main/raspb_3.
[3]An illustrative implementation of the agent reasoning cycle is available in https://github.com/LAIA-UFSC/SD2-Project.

an appropriate plan to achieve it. Once a plan is identified, the original goal is replaced on the stack with the set of actions (and any subgoals) specified by that plan. Finally, the agent's plan library, which details the know-how of the agent, i.e., plans defining the actions required to achieve their corresponding goals, is implemented as a list of tuples. This plan library maps goals to their corresponding step-by-step plans, facilitating efficient lookup and execution. The agent plan library has the following format:

$$
\begin{aligned}
&[\ \mathtt{goal}_1\mathtt{:\{context:\{b}_1,\ldots,\mathtt{b}_n\mathtt{\},plan:[a}_1,\ldots,\mathtt{a}_n\mathtt{]\},}\\
&\quad \ldots\\
&\quad \mathtt{goal}_n\mathtt{:\{context:\{b}_a,\ldots,\mathtt{b}_z\mathtt{\},plan:[a}_a,\ldots,\mathtt{a}_z\mathtt{]\}}\ ]
\end{aligned}
$$

The plan library defines a set of plans to achieve the agents' goals, for example, $\mathtt{goal}_1$ and $\mathtt{goal}_n$. To establish the conditions under which a particular plan can be used by the agents, each plan has a context that specifies the preconditions for its execution. The context essentially defines a set of beliefs that must be present in the agent's belief base. Finally, the plan itself is defined by a set of actions and/or subgoals, outlining a sequence of steps the agent can use to achieve the specified goal.

This data structure enables agents to dynamically update their know-how during execution, which is a critical feature of the proposed agent architecture. It is analogous to code that can modify itself (or be updated remotely) at runtime. Moreover, this structure supports the implementation of different classes of goals, such as *achievement* and *maintenance* goals.

Achievement goals are objectives in which the agent formulates a plan to accomplish the goal, executes the plan, and considers the goal achieved upon completion. This concept can be illustrated with the following abstract example: $[\ \mathtt{g}_1\mathtt{:\{context:\{b}_1\mathtt{,\ b}_2\mathtt{\},\ plan:[a}_1\mathtt{,\ a}_2\mathtt{,\ a}_3\mathtt{]\}}\ ]$, which represents a plan for the goal $\mathtt{g}_1$. The plan includes a context defined by $\mathtt{\{b}_1\mathtt{,\ b}_2\mathtt{\}}$ and a sequence of actions (and/or subgoals) $\mathtt{[a}_1\mathtt{,\ a}_2\mathtt{,\ a}_3\mathtt{]}$. When the context is satisfied, and the agent selects this plan, it stacks the actions in reverse order —- $\mathtt{a}_3$, $\mathtt{a}_2$, and $\mathtt{a}_1$ —- and then executes them sequentially in the order established by the plan. The agent achieves the specified goal upon completing the execution of $\mathtt{a}_3$, the final action in the stack.

Maintenance goals are objectives that the agent continually strives to maintain. These goals often represent desired or undesired states of the world that the agent seeks to preserve or avoid, respectively. They can be implemented using the following plan structure: $[\ \mathtt{g}_2\mathtt{:\{context:\{b}_3\mathtt{,\ b}_4\mathtt{\},\ plan:[a}_4\mathtt{,\ a}_5\mathtt{,\ g}_2\mathtt{]\}}\ ]$, which represents a plan for the goal $\mathtt{g}_2$. The plan is defined by the context $\mathtt{\{b}_3\mathtt{,\ b}_4\mathtt{\}}$ and a sequence of actions and subgoals $\mathtt{[a}_4\mathtt{,\ a}_5\mathtt{,\ g}_2\mathtt{]}$. In this structure, the agent recursively stacks the goal $\mathtt{g}_2$, so that when it is selected, a new plan of the same type is added to the intention stack. Agents can terminate the pursuit of such goals by selecting an alternative plan for the same goal. For instance, $[\ \mathtt{g}_2\mathtt{:\{context:\{b}_5\mathtt{\},\ plan:[a}_6\mathtt{]\}}\ ]$ represents a plan that, upon execution, concludes the maintenance of $\mathtt{g}_2$. This mechanism allows the agent to adapt its behaviour dynamically in response to changing contexts or priorities.

4 Case Study

We are interested in scenarios where embedded intelligent agents are capable of determining the spatial position of individuals based on their location within the input image captured by a computer vision interface. Specifically, we aim to extract the relative position of the individual with respect to the agent. In practical terms, this means that if an individual appears within the camera's capture area as processed by the computer vision interface, the system should identify the person's relative position, such as whether they are on the right or left side of the agent's field of vision. The agent can then make informed decisions on how to act based on this input, for example, tracking the individual with the camera or communicating the person's direction of movement to another agent.

The computer vision interface, as described in Section 3, utilizes the Yunet model for face detection and outputs a discrete representation of the face bounding box center's position. This position is mapped to one of nine quadrants within the camera's capture area, thereby identifying the individual's location within the captured image. Figure 4 shows an overview of the discretization and semantic extraction process. In this figure, the Yonet implementation identifies the person's face, which appears in the top-left quadrant of the camera's capture area. This information is discretized into a simple matrix, which is then used for semantic extraction, generating an event for the agent with the information `position(top_left)`.

To evaluate the proposed computer vision interface, we conducted a series of experiments by positioning our faces in different quadrants of the camera's capture area and comparing the resulting outputs. Figures 5, 6, and 7 illustrate examples of the experiments performed. For all experiment, our implementation returned the correct semantic extraction. Note that our approach for semantic extraction uses the central point of Yunet face recognition, which is positioned at the nose of the person, that why the experiment illustrated in Figure 6 returns `position(top_left)` when almost half of the face is at the `middle_left` quadrant.

After validating the proposed computer vision interface, the predicates generated by the semantic extraction component can be used to provide context to the agent. Specifically, the agent's belief base is updated with this information, triggering an event. The agent can then react to this event by creating a new goal and planning how to achieve it. This is accomplished by selecting an applicable plan from its plan library, i.e., a plan designed to achieve that particular goal, where the context (precondition) of the plan is satisfied by the agent's current beliefs.

To evaluate the proposed integration of the computer vision interface and the embedded agent, we implemented a ludic scenario with different agent profiles, which can be dynamically assigned to the agent by adding the predicate `my_profile(<profile>)` to its belief base. We considered three distinct profiles: the *confident* agent, which always looks directly into a person's eyes; the *thoughtful* agent, which always looks straight at the center of the person's face position; and the *shy* agent, which always looks downward toward the person's face position.

In the proposed scenario, whenever a face appears in the camera's capture area, the computer vision interface extracts the face's position and updates the agent's belief base by generating an event of the type `position(<quadrant>)`. Within the agent's reasoning

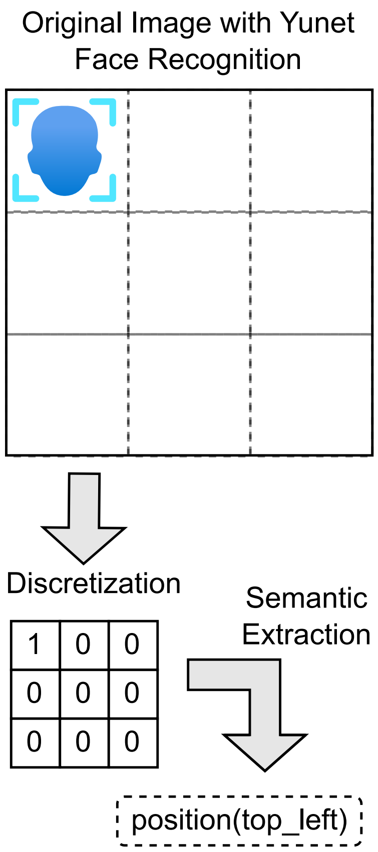

Figure 4: Discretization and Semantic Extraction.

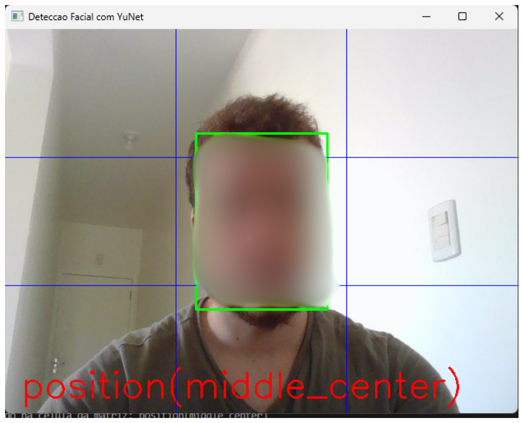

Figure 5: Semantic Extraction - Experiment 1.

cycle, this event is mapped to the goal `adjust_vision`. The agent then selects one of the relevant plans from its plan library to achieve this goal. All plans for `adjust_vision` require the current face's position, `position(<quadrant>)`, as part of their context, and they

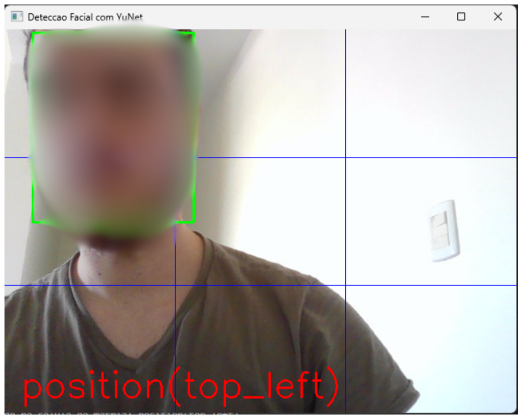

Figure 6: Semantic Extraction - Experiment 2.

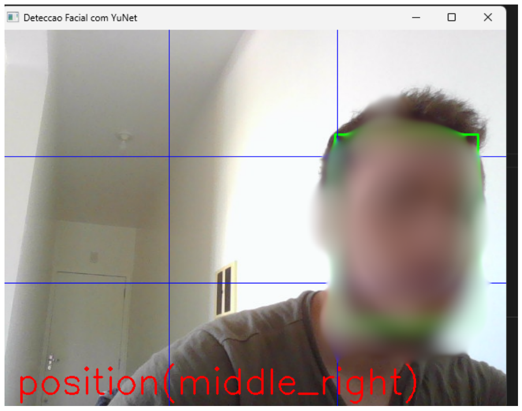

Figure 7: Semantic Extraction - Experiment 3.

differ based on the agent's profile. For instance, the first plan for `adjust_vision` is applicable only if the agent believes it has the *confident* profile, i.e., `my_profile(confident)`. If this condition is not met, the plan is deemed inapplicable, and the agent evaluates the next plan in the plan library.

```
[
  adjust_vision, {context:{position(X),
     my_profile(confident)}, plan:[look_to(X)]},
  adjust_vision, {context:{position(X),
     my_profile(thoughtful)}, plan:[look_strait(X)]},
  adjust_vision, {context:{position(X),
     my_profile(shy)}, plan:[look_down(X)]},
]
```

After the planning process — i.e., selecting an applicable plan and stacking it in the intention stack — the agent proceeds to execute the intention. In the proposed scenario, each plan corresponds to a single action. All actions are implemented as functions adhering to predefined action interfaces, and these functions are invoked by the agent architecture when selected for execution.

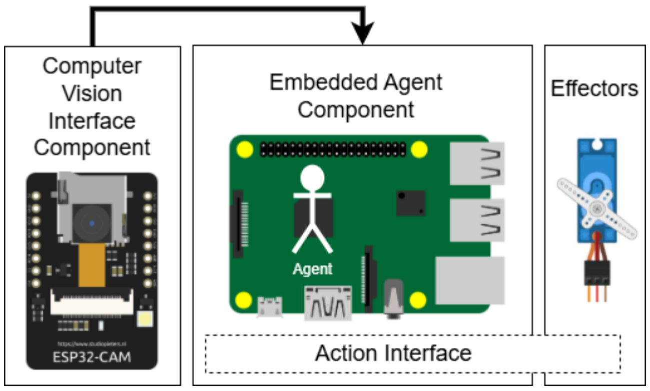

Figure 8: Case Study Components.

An overview of the main hardware components is presented in Figure 8. The system includes an ESP32-CAM microcontroller that captures images and transmits them to a Raspberry Pi 3, which performs the following tasks: (i) implements the computer vision interface by processing the video stream, applying the Yunet face recognition algorithm, discretizing the detection, performing semantic extraction, and updating the agent's belief base with the information extracted from the camera; (ii) hosts the interpreter for the proposed agent architecture, in which the agent keeps running according to the pseudo-algorithm 1, receiving inputs as events and planning and executing actions to achieve its goals; and (iii) integrates the action interface component, which implements all actions available to the agent.

5 Related Work

There are a few work on embedding intelligent agents, proposing agent architectures and frameworks. In [15], the authors propose (what they call) a new class of smart agent-based IoT devices, which is based on the implementation of software embedded agents. Agents execute within an unaware IoT environment (e.g., a smart home environment), in which IoT devices (agents) and resources are not known a priori. Their work focused on an embedded agent model and the architecture of an agent-based IoT device. While their work aligns with ours in conceptualizing embedded intelligence, our work considers more flexible interfaces, incorporating, for example, computer vision. This flexible interface is a central difference in our approach, and it allows us to incorporate different AI techniques according to the application need. In [27], the authors examine the challenges of embedding intelligence in compact, distributed computational artifacts for ubiquitous computing in domestic settings, raising important questions about communication and association among multiple smart artifacts — a key consideration for the scalability of our method. The authors in [28] also emphasize the integration of agents to create more intelligent, collaborative, and flexible devices, their case study focuses on a playful application.

The field of cloud and edge computing has also contributed to advancements in IoT intelligence. In [29], the authors explore computational offloading with multi-agent IoT systems supported by edge computing, leveraging federated learning and deep reinforcement learning. While this work shares a common interest in enhancing IoT devices' intelligence, its primary focus lies in resource optimization and task offloading, contrasting with our emphasis on integrating computer vision for perception and decision-making. In [30], the authors introduce Cresco, a distributed agent-based edge computing architecture, which enables efficient orchestration and management of edge resources. It emphasizes scalability, resource management, and agent communication, aligning with key challenges in IoT systems. Additionally, the discussions on IoT challenges and opportunities for smart cities, as well as resource allocation in distributed clouds, provide valuable context for our research, highlighting areas where our work can contribute to advancing these technologies.

In the domain of multi-agent systems and distributed systems, work such as [31] investigates multi-agent coordination in embedded systems, exemplified by a search-and-rescue scenario using drones, demonstrating the potential of agent collaboration. The work [31] proposes an architecture for programming intelligent robots also based on the cognitive concepts of BDI, which aligns with our approach of using intelligent agents to control IoT devices. In [32], the authors explore dynamic task offloading in multi-agent mobile environments, involving multiple mobile and edge devices. It employs reinforcement learning techniques to optimize task offloading decisions while accounting for energy and resource constraints. Although resource optimization is relevant to our work, our primary focus lies in integrating computer vision to enable enhanced perception and action within intelligent systems, which sets our approach apart. In [33], the authors focused on the development of a lightweight agent platform specifically designed for fieldbus systems, addressing their resource constraints to enable the deployment of distributed intelligence techniques. In [33], the objective was to extend the advantages of multi-agent systems to the fieldbus domain, enhancing control, monitoring, and automation in industrial processes. This approach highlights the potential for integrating intelligent agents into resource-constrained environments to improve operational efficiency and adaptability.

6 Conclusion

In this work, we proposed an approach for implementing embedded agents, which includes: (i) an agent architecture based on the BDI (Belief-Desire-Intention) model and the PRS (Procedural Reasoning System); (ii) a pseudo-algorithm specifying the agent's reasoning cycle; and (iii) well-defined interfaces based on the concepts of events and actions, which can be implemented according to the application domain.

In our approach, agents can dynamically update all their mental components, including their know-how, such as their plan library. This capability is particularly significant, as it allows new interfaces for sensors (events) or actuators (actions) to be integrated without requiring modifications to the agent's reasoning cycle. Such updates necessitate only the implementation and subsequent update of the agent's plan library.

As a case study, we demonstrated the implementation of an embedded intelligent agent enhanced by computer vision. The computer vision interface streams video capture of the person, performs face detection, discretizes the face position into a 9x9 grid, and executes semantic extraction to determine the face's position within the camera's capture area. This information is then updated in the agent's belief base, generating an event that corresponds to a specific agent goal. Subsequently, the agent formulates a plan to achieve the goal, with the planning process influenced by the agent's profile, which can be dynamically adjusted. Once the plan is established, the agent executes the corresponding actions.

Although the implemented case study is a ludic scenario, it effectively illustrates how our approach to embedded agents can seamlessly integrate AI techniques by well-defined interfaces based on events. Additionally, it demonstrates the flexibility of modifying the behaviour of embedded agents by assigning profiles to them or even updating the agent's plan library remotely. This flexibility is enabled by the plan library's design as a data structure that can be easily updated externally.

There are numerous potential applications for the proposed approach, particularly in intelligent environments, such as smart buildings, smart homes, and smart cities. In these contexts, embedded agents could adapt and personalize their behaviour in response to human presence and actions [14]. We are also interested in developing intelligent embedded agents capable of operating on highly resource-constrained devices with very low power consumption. Such agents could be utilized for monitoring remote areas, including forests or oceans, utilizing radio frequency communication with chain communication protocols.

Future research will enhance these agents with sophisticated interaction mechanisms, such as argumentation-based dialogue protocols, enabling them to negotiate and reason over optimal decisions in dynamic environments [34, 35]. We aim to develop embedded agents capable of collaborative resource allocation and autonomous coordination in decentralized networks with real-time constraints. Additionally, we will explore optimized communication strategies based on enthymemes [36], improving efficiency in agent interactions and decision-making.

References

[1] Wayne Wolf. Cyber-physical systems. *Computer*, 42(03):88–89, 2009.
[2] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & information systems engineering*, 6:239–242, 2014.
[3] Michael Wooldridge. *An introduction to multiagent systems.* John Wiley & Sons, 2009.
[4] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.
[5] D. D. Guinard and V. M. Trifa. What is the web of things? apachecon europe presentation, 2008. URL https://webofthings.org/2017/04/08/what-is-the-web-of-things/.
[6] Juan Carlos Augusto, Hideyuki Nakashima, and Hamid Aghajan. Ambient intelligence and smart environments: A state of the art. *Handbook of ambient intelligence and smart environments*, pages 3–31, 2010.
[7] Daniel Situnayake and Jenny Plunkett. *AI at the Edge.* " O'Reilly Media, Inc.", 2023.
[8] Hélio Pedrini and William Robson Schwartz. *Análise de imagens digitais: princípios, algoritmos e aplicações.* Cengage Learning, 2008.
[9] Richard Szeliski. *Computer vision: algorithms and applications.* Springer Nature, 2022.
[10] Rafael C Gonzalez and Richard E Woods. Digital image processing 4th edition, global edition, 2018.
[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[12] Guangxu Zhu, Dongzhu Liu, Yuqing Du, Changsheng You, Jun Zhang, and Kaibin Huang. Toward an intelligent edge: Wireless communication meets machine learning. *IEEE communications magazine*, 58(1):19–25, 2020.
[13] Mithun Mukherjee, Rakesh Matam, Constandinos X Mavromoustakis, Hao Jiang, George Mastorakis, and Mian Guo. Intelligent edge computing: Security and privacy challenges. *IEEE Communications Magazine*, 58(9):26–31, 2020.
[14] Hani Hagras, Victor Callaghan, Martin Colley, Graham Clarke, Anthony Pounds-Cornish, and Hakan Duman. Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Systems*, 19(6):12–20, 2004.
[15] Juan A Holgado-Terriza, José Caicedo-Ortiz, and Pablo Pico-Valencia. Embedded agents for the development of smart devices on the internet of things. In *Workshops at 18th International Conference on Intelligent Environments (IE2022)*, pages 233–242. IOS Press, 2022.
[16] Shihao Shen, Yiwen Han, Xiaofei Wang, and Yan Wang. Computation offloading with multiple agents in edge-computing–supported iot. *ACM Transactions on Sensor Networks (TOSN)*, 16(1):1–27, 2019.
[17] Leslie Pack Kaelbling and Stanley J Rosenschein. Action and planning in embedded agents. *Robotics and autonomous systems*, 6(1-2):35–48, 1990.
[18] Takumi Kato, Ryo Chiba, Hideyuki Takahashi, and Tetsuo Kinoshita. Agent-oriented cooperation of iot devices towards advanced logistics. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 3, pages 223–227. IEEE, 2015.
[19] Daniel H. De La Iglesia, Gabriel Villarrubia González, André Sales Mendes, Diego M Jiménez-Bravo, and Alberto L. Barriuso. Architecture to embed software agents in resource constrained internet of things devices. *Sensors*, 19(1):100, 2018.
[20] Meftah Zouai, Okba Kazar, Guadalupe Ortiz Bellot, Belgacem Haba, Nadia Kabachi, and M Krishnamurhty. Ambiance intelligence approach using iot and multi-agent system. *International Journal of Distributed Systems and Technologies (IJDST)*, 10(1):37–55, 2019.
[21] Luis Alberto Cruz Salazar, Felix Mayer, Daniel Schütz, and Birgit Vogel-Heuser. Platform independent multi-agent system for robust networks of production systems. *IFAC-PapersOnLine*, 51(11):1261–1268, 2018.
[22] Michael Bratman. Intention, plans, and practical reason. 1987.
[23] Jaeho Lee, Marcus J Huber, Edmund H Durfee, and Patrick G Kenny. Um-prs: An implementation of the procedural reasoning system for multirobot applications. *NASA. Johnson Space Center, Conference on Intelligent Robotics in Field, Factory, Service and Space (CIRFFSS 1994), Volume 2*, (AIAA PAPER 94-1297-CP), 1994.
[24] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, 1969.
[25] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57:137–154, 2004.
[26] Wei Wu, Hanyang Peng, and Shiqi Yu. Yunet: A tiny millisecond-level face detector. *Machine Intelligence Research*, 20(5):656–665, 2023.
[27] Vic Callaghan, Graham Clarke, Martin Colley, and Hani Hagras. Embedding intelligence: Research issues for ubiquitous computing. In *Proceedings of the 1st Equator IRC Workshop on Ubiquitous Computing Environments*, pages 110–130, 2001.
[28] Tim Finin, Anupam Joshi, Lalana Kagal, Olga Ratsimor, Sasikanth Avancha, Vlad Korolev, Harry Chen, Filip Perich, and R Scott Cost. Intelligent agents for mobile and embedded devices. *International Journal of Cooperative Information Systems*, 11(03n04):205–230, 2002.
[29] Shihao Shen, Yiwen Han, Xiaofei Wang, and Yan Wang. Computation offloading with multiple agents in edge-computing–supported iot. *ACM Transactions on Sensor Networks (TOSN)*, 16(1):1–27, 2019.
[30] VK Cody Bumgardner, Victor W Marek, and Caylin D Hickey. Cresco: A distributed agent-based edge computing framework. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 400–405. IEEE, 2016.
[31] Gustavo R Silva, Leandro B Becker, and Jomi F Hübner. Embedded architecture composed of cognitive agents and ros for programming intelligent robots. *IFAC-PapersOnLine*, 53(2):10000–10005, 2020.
[32] Javad Heydari, Viswanath Ganapathy, and Mohak Shah. Dynamic task offloading in multi-agent mobile edge computing networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.
[33] Yoseba K Penya, Stefan Mahlknecht, and Peter Rössler. *Lightweight agent platform for high-performance fieldbus nodes.* Citeseer, 2002.
[34] Alison R Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Towards practical argumentation-based dialogues in multi-agent systems. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 151–158. IEEE, 2015.
[35] Alison R Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordin. Towards practical argumentation in multi-agent systems. In *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, pages 98–103. IEEE, 2015.
[36] Alison R Panisson and Rafael Heitor Bordini. Uttering only what is needed: Enthymemes in multi-agent systems. In *Proceedings of the 16th International Conference on Autonomous Agents & Multiagent Systems (AAMAS-2017), 2017, Brasil.*, 2017.