RAI: Flexible Agent Framework for Embodied AI

Kajetan Rachwał[1][0000−0003−1524−7877], Maciej Majek[1][0009−0009−9541−8461], Bartłomiej Boczek[1][0009−0006−9097−9655], Kacper Dąbrowski[1][0009−0001−5661−9801], Paweł Liberadzki[1][0000−0002−4072−7605], Adam Dąbrowski[1][0000−0002−2130−0577], and Maria Ganzha[2][0000−0001−7714−4844]

[1] Robotec.AI, Warsaw, Poland
{name.surname}@robotec.ai
[2] Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland

Abstract. With an increase in the capabilities of generative language models, a growing interest in embodied AI has followed. This contribution introduces RAI – a framework for creating embodied Multi Agent Systems for robotics. The proposed framework implements tools for Agents' integration with robotic stacks, Large Language Models, and simulations. It provides out-of-the-box integration with state-of-the-art systems like ROS 2. It also comes with dedicated mechanisms for the embodiment of Agents. These mechanisms have been tested on a physical robot, Husarion ROSBot XL, which was coupled with its digital twin, for rapid prototyping. Furthermore, these mechanisms have been deployed in two simulations: (1) robot arm manipulator and (2) tractor controller. All of these deployments have been evaluated in terms of their control capabilities, effectiveness of embodiment, and perception ability. The proposed framework has been used successfully to build systems with multiple agents. It has demonstrated effectiveness in all the aforementioned tasks. It also enabled identifying and addressing the shortcomings of the generative models used for embodied AI.

Keywords: Embodied AI · Multi-Agent Systems · Robotics · Generative AI · Digital Twin · Simulations

1 Introduction

Since the advent of Large Language Models (LLMs), there has been a growing interest in their applications in embodied AI agents [10] [17]. This effort has accelerated with the introduction of tool or function calling mechanisms, leading to the creation of more complex embodied AI systems [22]. These systems often rely heavily on custom implementations that enable integration with foreign APIs and AI models.

Some existing solutions facilitate the development of tools for API integration, such as ModelScope [11]. Systems such as ROSA [20] are tightly coupled with particular solutions (in this case ROS and ROS 2[15]), but do not offer solutions for integration with other communication stacks. Frameworks such as

AgentGYM [23] offer agents a standardized environment and evaluate LLM-based agents within that environment. These LLM-based solutions are generally designed for single-agent deployments. However, all of the above inherit common limitations of LLMs, including hallucinations [9], a lack of interpretability and explainability [26], and susceptibility to jailbreak attacks [24]. These constraints often lead to serious functional and ethical concerns.

It is worth mentioning that before the rise in popularity of LLMs, several frameworks were already developed for building robotic agents. For example, FABRIC [21] has been introduced to develop robotic agents that utilize reusable components for seamless integration with ROS. Likewise, MARC [4] has been designed for Multi-Agent Systems (MAS) based on reinforcement learning. In addition to these, some frameworks have been proposed for specific robotic applications [27,12,19], ranging from designs based on multi-robot cooperation to human-robot interaction with multiple people. An important contribution toward creating a generalized framework applicable across various robotic domains is the G++ architecture [6]. All of the aforementioned approaches lack the flexibility and potential for human-robot interaction offered by native integration with LLMs.

In this context, this paper presents RAI, a framework designed for creating Multi-Agent Systems (MAS) with built-in capabilities for integration with LLMs, robotic stacks, external APIs, and human-robot interaction mechanisms. It also comes with components which support embodiment, such as tools for understanding Agent's physical form. Generality is achieved through a modular and scalable architecture, which is based on the M-Agent model [1]. This model defines an agent through its reasoning system, sensors, and actuators.

In RAI, these abstractions are streamlined into the agent's core logic and connectors – utility components that allow the agent to both observe and interact with a broadly defined environment. RAI agents can be virtual (interacting solely with a digital environment), physical, or exist within a virtual-physical continuum, seamlessly integrating both domains. By being in a single MAS, the virtual and physical agents can communicate with each other to better achieve their goals. The details of these mechanisms are described in Section 2.

This paper presents three deployments in different environments to validate the viability of RAI. The performance of the system in an edge computing setup within a physical environment has been tested using ROSBotXL, a ROS 2 autonomous mobile robot platform. RAI agents have been deployed on the platform to observe the environment, control robot movement, and enable human-robot interaction (HRI) capabilities. Details of this experiment are presented in Section 3.1. In addition, two pure virtual (simulation-based) experiments have been performed. These RAI agents have been utilized for manipulation tasks with a simulated robotic arm and controlling a tractor. The details are described in Section 3.

2 Architecture

RAI architecture defines high level abstract classes, that allow for concise and flexible Agent definition. The current section focuses on details of these interfaces and describes example use cases.

2.1 Component definitions

At the highest level, RAI uses three abstractions – *Agents*, *Connectors*, and *Tools*.
Agents are the basic building blocks of a MAS. RAI *Agent* implements a simple interface containing two methods, `run()` and `stop()`. *Agents* are also guaranteed to possess *Connectors* – mechanisms for observing and actuating their environment. It is up to the user to define the internal model of the environment and the decision-making process for selecting actions. Some RAI *Agents* use simple rules-based systems, while others can utilize mechanisms such as LLMs for the decision-making process.
Connectors are an abstraction that represents the sensors and actuators of the *Agents*. They enable three modes of communication: publish-subscribe, service-based, and action-based. Previous experiences [8] have shown that relying only on one mode of communication can be limiting, so with RAI, we decided on an approach similar to ROS 2 communication. The publish-subscribe mode provides utilities for publishing and receiving messages from other hosts. In most implementations (e.g., ROS 2, MQTT, XMPP), this method of communication is efficient and performant. However, its reliability is limited as there is no guarantee that a sent message will be received. The service-based mode addresses this limitation by providing the API caller with information on the message delivery and the result of the call. This type of communication is usually less performant than the publish-subscribe mode (given the need to receive a response), but provides more reliability. Finally, the action-based mode is modeled after the ROS 2 Actions API, which informs users about the acceptance and outcome of actions. In addition, it provides continuous feedback on the execution state.

A specific RAI *Connector* can implement all of these modes or combine any of them depending on the particular communication mechanism it encapsulates.
Tools are an abstraction inspired by LLMs and their tool-calling mechanisms. A specific *Tool* implements a way to create or parse data sent or received through a *Connector*. These implementations are compatible with langchain [3], enabling seamless integration with tool-calling-enabled LLMs. They can also be used by *Agents* utilizing other decision-making mechanisms.

2.2 Embodiment

Robotic embodiment lacks a clear and broadly accepted definition. Some researchers [7] argue that simply having information on ways to interact with

[3] `https://www.langchain.com/`