Shopping with Pepper
Elective in AI / HRI + RBC Report

Vincenzo Crisà 2143080 - Stefano D'Urso 2143081 - Fabrizio Italia 2143104

All students have equally contributed to the project.

1 Introduction

The project involves a socially interactive robot navigating autonomously through a supermarket environment with the goal of assisting customers through natural, multilingual interaction, responding to questions related to product information (such as price or ingredients) and product location within the supermarket. A key emphasis is on creating rich, human-like interaction between the robot and customers, requiring it to be a social cognitive robot capable of expressing emotions through gestures and also handling unexpected situations.
The project involves human-robot interaction and includes aspects of robotic environment representation, perception and planning. To develop this level of interaction, the robot employs real-time face recognition to identify recurring customers and personalize interactions accordingly (e.g., it does not give information about alcohol to underage people). The system also integrates speech-to-text (STT) modules to process spoken input from customers and employs a GPT model exploiting the OpenAI Assistant API to generate coherent, context-aware and multilingual responses in natural language, enhancing social skills. The combination of these modules improves the fluidity of conversations and thus of the human-robot interactions, allowing the robot to interpret and execute appropriate social behaviors in response to what the customers ask. For its expressive design and social capabilities, we employed Pepper by SoftBank Robotics.
In addition to the deployment of these capabilities, the project integrates an RBC evaluation framework. Robot benchmarking enables a modular and standardized analysis of the robot's perception, language understanding and planning modules through prompt-based GPT agents. This benchmarking process quantifies the contribution of each module to the complete pipeline and ensures that spatial reasoning, object grounding and action planning are evaluated against human-annotated ground truth data.

1.1 Context and motivation

The integration of socially interactive robots into everyday environments has gained significant attention in recent years, in particular in domains such as hospitals, schools, airports and also shopping centers.
This project refers to a supermarket environment, where the robot has to assist customers during their shopping experience. Places like supermarkets are dynamic, complex spaces that require skills such as good social interaction, spatial reasoning and real-time adaptability. For these reasons, this domain is an ideal environment to test social robots' abilities to interact with humans.
The motivation for deploying this technology is multifaceted. Socially interactive robots can enhance the

accessibility and inclusivity of public spaces, for instance, assisting elderly customers, or those with disabilities, offering personalized support in navigating the store, finding products or understanding information such as ingredients and allergens of some products. The multilingual capability of the robot can further expand its applicability to any kind of cultural setting, breaking down the lingual barriers. From an economic point of view, employing social robots as assistants can increase operational efficiency by continuously updating its knowledge of product locations, guiding customers and also reducing the demand on human staff for routine requests. In this way the customer experience can be enhanced, but even the employees can reduce their workload.

1.2 Objectives

The objectives of the project are to prove the social ability of Pepper when dealing with human interaction and test semantic mapping ability through functional and task benchmarks, supported by a knowledge representation of its workplace.
Particularly, the core of the project is to demonstrate Pepper's skills to be involved in socially intelligent behavior when interacting with humans in a supermarket environment. This includes not only providing accurate product information and spatial guidance but also interpreting and accordingly answering to customers in multilingual natural language, supported by the awareness of the specific work context. A key focus is the personalization of the interactions based on user profile, carrying out a more human-like and adaptive communication style.
To rigorously evaluate these objectives, the project leverages an RBC evaluation framework, including both functionality benchmarks and scene-level coherence checks. Each module, such as environment semantic perception and symbolic planning, is independently scored using a weighted evaluation scheme. This allows to precisely access the robot's strengths and weaknesses in its perception-action pipeline. Furthermore, robot benchmarking allows to identify bottlenecks in the robot's work pipeline, such as failures in spatial understanding that impact the downstream planning, guiding to future improvements and ensuring a transparent evaluation methodology.

1.3 Summary of the results

In this project, we successfully demonstrated Pepper's capability to engage in socially intelligent behavior within a supermarket environment.
In nominal situations, the Pepper robot performs robustly across the complete interaction flow: identifying users through face recognition, engaging them via natural language dialogue, retrieving and presenting product-related information (price, location, description), maintaining personalized conversations, adapting its responses based on user identity, age and language and leveraging the OpenAI's GPT-4.1 Assistant.
In non-nominal situations, the robot exhibits advanced reasoning capabilities. It handles lost customer tracking through visual feedback and threshold counters, recovers gracefully from speech recognition failures with polite responses, manages age-restricted product requests based on customer profiles and language-specific legal thresholds and respects user privacy preferences by dynamically adapting data retention behavior. These situations highlight Pepper's ability to maintain socially acceptable, ethical and context-aware interaction even under challenging conditions. A higher level of personalization experience, such as history of interaction of recurrent customers, would have required a database to be included in the query to OpenAI Assistant, but for quota limitation this aspect was not implemented.
The RBC evaluation framework validates the system's reasoning pipeline using task-specific benchmarks across visual perception, scene description and symbolic planning. The modular, prompt-based architecture

achieved high scores in planning and scene description. However, the Environment Agent emerged as a performance bottleneck, primarily due to spatial relation detection and object localization challenges. Overall, the system maintained a success rate of 80% across tested scenarios with an average pipeline score of 0.764 and average end-to-end latency of ~ 15 seconds. More complex scenes revealed performance drops mainly tied to spatial reasoning.
Together, these results demonstrate that the Pepper robot can function effectively as a socially intelligent assistant in real-world retail environments while being rigorously evaluated using a transparent, modular benchmarking framework.

2 Related works

In contexts like robots in service in retail selling environments, recent studies highlight how social robots adoptions contributes to enhance the shopping experience. Khan and Erden [1] provide a systematic summary of social robots literature in shopping contexts, capturing that the main developments in the last two decades were focused on autonomous and mobile robots for navigation assistance and customer service in supermarkets and shopping centers. In a real-world experiment [2], two social robots employed in a supermarket successfully carried out customer guidance, product promotion and entertainment tasks, identifying several roles in this specific context.
From questionnaires it emerged that customers mainly perceives positively the robots' presence, even if a not negligible subset consider them unpleasant. The works of Niemelä et al. [3], Barnett et al. [4], but also [5] and [6], indeed assert that consumer users have been shown to positively perceive socially interactive service robots. Consumers value the unbiased, speedy and error-free behavior that a robot can provide, although many customers found that if the robot was perceived to be void of personality, it would negatively impact the service experience. These studies affirm that a humanoid social service robot can have a positive role in a shopping scenario, attracting customers due to its novelty and the capability of providing pleasant shopping experience.
In supermarkets is essential to implement a socially aware robotic behavior: Lewandowski et al. [7] show that integration of real-time people recognition and multiple communication channels (speech, video, LED lights) improve the robot acceptance and keep the appropriate distance, avoiding to bother customers in tight lane.
Thompson et al. [8] introduced Blueberry, an autonomous shopping assistant robot capable of escorting customers to desired products in grocery stores without requiring a pre-built map of the environment. Their HRI study demonstrated that customers found the robot helpful, enjoyable and easy to use. This solution is particularly valuable in light of evidence showing that crowding and queuing often induces stress and anxiety in shoppers under time pressures [9], as well as in elderly customers who may struggle to cope with such situations [10]. These stressors can lead to shoppers failing to purchase their intended products, ultimately causing customer dissatisfaction leading to the loss of client [11]. The approach proposed by Thompson et al. thus supports the potential for flexible, on-the-fly deployment of robotic assistants in diverse retail spaces.

2.1 HRI: Face Recognition, Speech-to-Text and Dialogues in Natural Language

Notable works explore the Pepper robot as a social-cognitive agent capable of recognizing users' faces and engage natural-language dialogues with them converting their responses to text [12]. For example, in [13], the authors developed an LLM-based agent for a robot trainer, introducing a memory system for facilitating reasoning based on knowledge built across different interactions. They employed YOLOv8 for face

detection and a Whisper Automatic Speech Recognition (ASR) model for speech transcription, feeding this multimodal input into a large language model (LLaMA2/GPT-like) dialogue manager.
Another notable work is proposed by Perera et al. [14], where Pepper was extended with a face recognition service via a front-camera and a web service, along with speech recognition, enabling the robot to identify users and execute spoken commands.
More recent works demonstrate integration of OpenAI models with the aim of making humanoid robots more relatable. In [15] the authors employ Large Language Models (LLMs) to show the potential to solve the communication barrier for humanoid robotics, comparing different ASR APIs, with the integration of Whisper ASR and ChatGPT with Pepper. The comparison result shows that 60% of the participants thought the performance of the Pepper-GPT was "excellent". Another approach is proposed by Sevilla-Salcedo et al. in [16], using transformer-based models to paraphrase or generate diverse social-robot utterances on-the-fly, reducing the repetitiveness of scripted dialogues.
A relevant work where human-robot interaction is handled with multimodal input processing to generate appropriate behaviors is presented by Kang et al. [17]. They proposed SoR-ReAct, an LLM-agent framework for social robots that incorporates episodic memory retrieval based on user recognition and simulates the emotional states of the Nadine social robot in response to human interactions.
Our world modeling strategy is based on retrieving information from product and user databases managed by GPT Assistant to support and enhance social reasoning.

2.2 RBC: Object Detection, Semantic Segmentation, Symbolic Planning

To improve robot capabilities in dealing with unknown domains, a key focus lies in world modeling and information management. This involves a structured separation between prior knowledge, dynamic environmental models, real-time sensory information and an object-oriented representation of the world. This project builds upon foundational concepts in robotic benchmarking and semantic scene understanding as presented in the RBC-2025 lectures and related seminars. In particular, Lecture 3 [18] introduced the principles of task and functionality benchmarking in robotics, emphasizing the need for standardized tasks, environments and metrics such as accuracy, robustness and efficiency. The YCB Object and Model Set, widely used for manipulation benchmarks, has informed our choice of evaluation objects and procedural protocols. Lecture 8 [19] formalized the notion of a semantic map as the triplet $\langle R, M, P \rangle$, where R is a global reference frame, M a set of geometric elements and P a set of predicates describing the environment. It also proposed quantitative metrics, namely the Object Reconstruction Index (ORI) and Object Predicates Index (OPI), for assessing reconstruction quality and predicate correctness in semantic mapping tasks. We take ispiration of this framework to evaluate our shelf-scene reconstructions and predicate extraction (e.g. `("item", "on", "shelf")`) and we measure performance both in terms of geometric fidelity and semantic accuracy. For the perception and grounding modules, we were inspired by the EMPOWER framework described in Lecture 12 [20], which integrates open-vocabulary grounding and task planning for embodied agents via large-language-model inference [21]. Similar to EMPOWER, our pipeline uses a YOLO-based detector for initial object proposals, the Segment Anything Model (SAM) for precise mask generation and a GPT-based reasoner to convert scene graphs into symbolic task plans. Unlike prior work that focuses on static vocabularies, we leverage GPT's open-vocabulary interface to handle novel product names and shelf arrangements. Building on this, we propose a modular multi-agent evaluation framework that decompose the vision-language pipeline into specialized agents. Each agent is scored independently. Our evaluation strategy parallels prior benchmarks but diverges from standard metrics like mean average precision (mAP) or Intersection over Union (IoU). Instead, we focus on symbolic and semantic consistency across agents, including spatial relation F1-score, object mention rate, planning action overlap and cross-module coherence. These

metrics are better aligned with our pipeline and we extended them to the grocery-shelf domain and integrated cross-module consistency checks and instrumenting latency.
Overall, our work focuses on its end-to-end integration and diagnostic benchmarking of perception (YOLO + SAM), open-vocabulary reasoning (GPT), task execution, agent-wise scoring and scene coherence, tailored to grocery-store shelf scenarios.

3 Integrated Solution

This section outlines the key features that an ideal robot should have to be effectively employed in our application, followed by the integrated solution we developed to approximate this ideal, enabling a socially interactive robot to operate autonomously within a supermarket environment.
The ideal robot would be designed not only to navigate efficiently in a dynamic environment, but also to establish personalized and meaningful interactions with customers. Physically, it should have human-like features, such as expressive gestures, approachable height and a friendly behavior, to encourage trust and engagement and make affordances explicit. Importantly, its appearance should leverage a balance in the human-likeness spectrum, remaining just before the Uncanny Valley [22]. Moreover, the robot should be able to build a semantic map of the store, including the products in the various aisles to dynamically update its database and to assist customers, for example by leading them to specific aisles or helping retrieve products. A customer-specific database would enable the robot to tailor responses based on past purchases and preferences. For instance, if a customer frequently buys a specific brand of a product that is currently unavailable, the robot could suggest alternatives and even propose recipes involving previous purchases, elevating it from a passive information provider into an active assistant. For example:

> ***Customer:*** *Where is tuna?*
> ***Robot:*** *Hi* `<Name>`*, the brand you usually buy isn't in stock, but I can suggest you a similar one you might like!*
> *[the next day]*
> ***Customer:*** *Where's pasta?*
> ***Robot:*** *Welcome back! Pasta's in aisle 5. Want a recipe with the tuna you picked up yesterday?*

To enable such complex interaction, the robot should rely on an embedded AI module capable of generating real-time, personalized dialogue based on the customer-specific and products databases, without being constrained by API token limits or quota restrictions.

Given these requirements, for HRI we selected Pepper by SoftBank Robotics as our hardware platform. Pepper is a widely employed social humanoid robot with 17 degrees of freedom, omnidirectional wheels and multiple sensors including microphones and RGB cameras [23]. It supports object and face detection and speech recognition. Its expressive design and social capabilities make it well-suited for human-facing environments and a strong foundation for integrating advanced AI modules.
The robot's functional architecture we employed for HRI is depicted in Figure 1 and it is characterized by the following modules:

- **RAIM (Rich Adaptive Interaction Manager):** manages communication among all system modules via a standardized protocol.
- **PepperBot Interface:** provides low-level control of Pepper's physical behaviors.

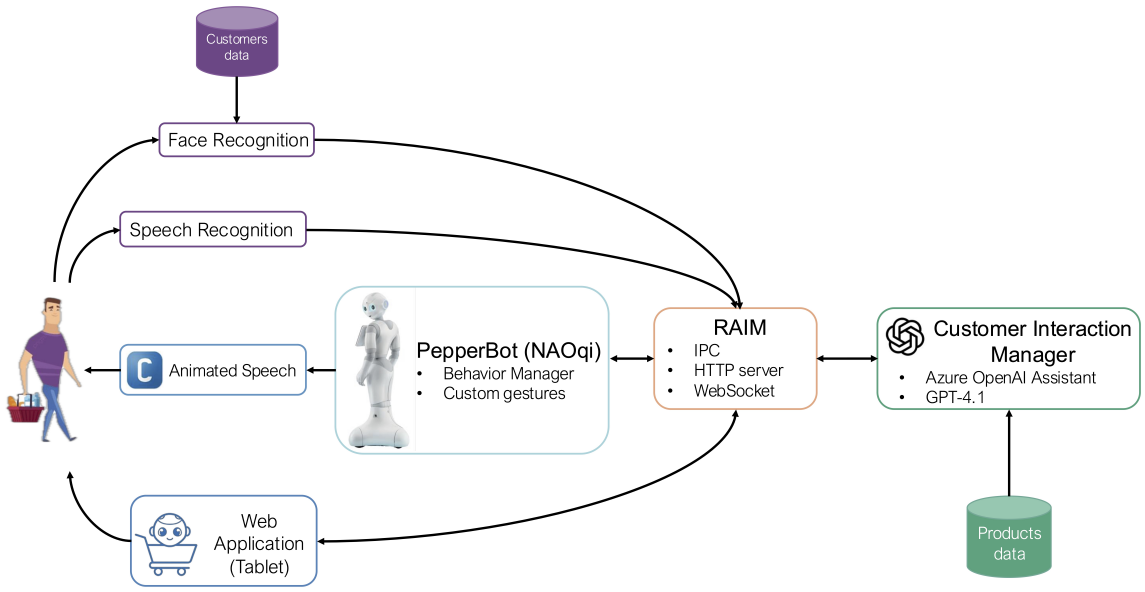


Figure 1: HRI system architecture. Social signals from customers are perceived through face and speech recognition, supported by a customer information database. The RAIM module acts as the central communication hub, connecting all components via IPC, HTTP and WebSocket protocols. Customer requests are processed by the Customer Interaction Manager, powered by an Azure OpenAI Assistant (GPT-4.1) and augmented with product data. The robot's physical behavior is controlled through the PepperBot interface (NAOqi), while the tablet interface is simulated by a web application. Social responses are generated through animated speech and multimodal interaction. Bidirectional arrows indicate client-server communication.

- **Face Recognition Module:** identifies and distinguishes between known and unknown users for personalized interaction.
- **Customer Interaction Module:** handles natural language understanding, multilingual responses and connects with GPT-4.1 for intelligent dialogue. The robot is designed to exhibit cognitive functions such as reasoning and learning and to behave according to social norms. To enable mental state attribution and context-aware interactions, we employed OpenAI's GPT to manage the dialogue phases.
- **Web Application Module:** offers a multimodal user interface on external devices.

The following subsections describe each of these modules in detail, followed by an explanation of how they are integrated into the interaction pipeline.
Since Pepper's physical features are not suitable for navigating and grasping tasks, we implemented semantic mapping developing a dedicated benchmarking and evaluation pipeline to assess the performance of the robot when executing perception-based and goal-directed tasks in a controlled environment. The RBC system focuses on testing the robot's capacity to perceive, describe and act upon complex visual scenes in a modular, interpretable and measurable way.
The RBC architecture is composed of a multi-agent system where each agent corresponds to a specific

cognitive or perceptual function. These agents are implemented as prompt-based GPT-4o calls, allowing diagnostics and performance scoring at each stage of the reasoning process. Each agent's output is evaluated against structured ground truth data, enabling objective and reproducible assessment across tasks and scene variations. The RBC evaluation framework includes Environment Agent, Description Agent, Planning Agent, Information Retrieval Module and Coherence Module. Due to hardware constraints, our evaluation pipeline currently stops after the symbolic plan is generated by the Planner Agent and the creation of the scene point cloud. As a result, no robotic simulation is performed in a custom environment (to be enriched with the obtained point clouds) and the assessment focuses only on cognitive and decision making stages.
Each module is independently evaluated using task-specific metrics and their weighted combination determines overall success. In this way, the RBC framework measures end-to-end pipeline performance.

3.1 HRI

The proposed solution is a modular client-server architecture, integrating several components that enable the robot to assist customers in a supermarket environment. All the modules communicate with each other through RAIM which acts as the central server. RAIM is designed to manage communication among heterogeneous modules, allowing the system to easily integrate components built with different technologies. Indeed, some modules can run on a machine using Python 3, while others can operate on Pepper's native Python 2 environment.

3.1.1 RAIM

In this project, RAIM serves as the central communication core that manages the interaction between Pepper and the modules that handle the various features required to support the robot in assisting customers, allowing real-time information exchange between server and client modules. Initially, we considered to use MODIM, which is a widely adopted interaction manager, but it did not suit our necessities. Indeed, we needed a more adaptable solution that could run on both Python 2 and Python 3, leveraging also modern web technologies like WebSockets.
A server module is an abstraction of the underlying communication channel or platform. The modules communicate using a standardized custom communication protocol, which is based on RAIM commands, thus the process of forwarding requests by server modules is a command dispatching. A RAIM command is a structured packet containing a unique identifier for the packet, the unique ID of the receiving client module, the unique ID of the sending client module, a boolean flag indicating whether the sender expects a response or if the command is standalone, the content of the command itself (as a JSON object) and a boolean flag to confirm the successful receipt of a command.
To support this architecture, the system is built on three different types of server modules: the HTTP server, the Inter-Process Communication (IPC) server and Websocket server.
The first module handles web-based communication, abstracting the HTTP interactions and allowing browsers on devices like tablets to easily connect with the system.
The second server module, the IPC server module, manages the communication between different python-based client modules, such as the face recognition server and the customer interaction server. These components, developed in Python 3, communicate via TCP sockets, allowing system compatibility with both Python 2 and Python 3 environments.
The WebSocket server facilitates communication between multiple javascript-based client modules, using WebSocket channels.
On the client side, each module is responsible for connecting a specific system feature to the RAIM infrastructure, exposing APIs that enable other components to access their specific functionalities. For example,

the client interaction manager exposes the user query that will be handled by GPT.
We developed a total of three client modules: **Customer Interaction Manager Client**, **Face Recognition Client** and **Pepper Bot Client**.

3.1.2 PepperBot Interface

To control the robot within our system, we used a dedicated communication interface called PepperBot, designed to integrate smoothly with our architecture. Establishing a connection with the robot is indeed straightforward: it is enough to provide the robot's IP address and port to connect through the NAOqi framework, giving access to a wide range of native services offered by the Pepper robot. Through this interface, it is possible to dynamically adjust Pepper's behavior to suit various interaction scenarios (e.g., it manages speech output, joint movements, eye color and the activation of different sensors). All these robot commands can be executed either sequentially or simultaneously, thanks to multi-threading, which makes it possible to combine different actions for richer interaction.
To make this system accessible via remote API calls over the network, we included the interface within a RAIM client module to bridge PepperBot with the RAIM communication framework. This server listens for incoming commands containing sequences of actions for the robot to execute. Each action is defined by an action type, which specifies the task the robot should perform and a set of action properties that provide any additional parameters required to carry out the action.

3.1.3 Face Recognition

A face recognition system is integrated in the system, to enable the robot to reason and distinguish between new users and known ones and thus behave accordingly.
Our solution is built using the Python face-recognition library, which leverages `dlib`'s facial recognition technology. This module is initialized with two key parameters: the resize value, which controls how much the input image is scaled down to improve processing speed and the unknown face threshold, defining how many consecutive frames an unidentified face must appear in before being confirmed as an unknown user. This threshold is helpful to prevent situations where known users are mistakenly classified as strangers due to brief recognition errors.
The face recognition pipeline starts processing a video frame, resizing the image according to the configured scale, detecting visible faces and computing an embedding for each one. The system checks whether each detected face has been seen before and if so, the face is labeled as known user and their associated information is retrieved. If the captured face is not recognized, it is temporarily marked as potentially unknown and, if it keeps on appearing in the following frames for at least the number of frames defined by the unknown face threshold, the system confirms it as an unknown user.
At the end of this process, the module provides updated information about both known and unknown users, which can then be used by the robot's decision-making and interaction modules. Additionally, the system includes functionality to save information about unknown users, allowing the robot to recognize them in future encounters as known users. This saving process starts only if the users give the consent to store their information, otherwise Pepper will not remember the customers the next time they ask for help.
This module is also wrapped as a RAIM client module, as already seen for PepperBot.

Figure 2: Sample products from the database. All the images are generated by ChatGPT.

3.1.4 Customer Interaction Module

The Customer Interaction Module is central in the management of the dialogue between the robot and the supermarket customers. This module is responsible for processing the natural language queries from the users and providing personalized assistance based on individual characteristics, such as age and language. Additionally, it accesses the supermarket's inventory database in JSON format, to offer real-time and context-aware information about the products.
An example of the employed database follows:

```
{
    'name': 'Whole Milk',
    'image_path': 'images/milk.png',
    'overage': 'False',
    'category': 'Dairy',
    'price': 1.49,
    'location': 'Aisle 5, Shelf B2',
    'description': 'Fresh whole milk from local farms',
    'ingredients': [
        'pasteurized whole milk'
    ],
    'allergens': [
        'milk'
    ],
    'keywords': [
        'milk',
        'dairy',
        'fresh'
    ]
}
```

Each product is characterized by the product's name, a path to a static image generated by GPT (an example is shown in Figure 2), a boolean field indicating whether the product is restricted to overage customers, the cost, the location in the supermarket, a brief description, a list of the ingredients, any allergens and a set of relevant keywords for easy retrieval. Through the RAIM infrastructure, the module can communicate with

```
"PEPPER_NEW_USER_INTRO": {
    "en-US": "So, %s, do you need help with something?",
    "it-IT": "%s hai bigogno di aiuto?"
},
"PEPPER_ASK": {
    "en-US": "%s, do you need help with something?",
    "it-IT": "%s, posso aiutarti con qualcosa?"
},

"PEPPER_ASK_AGAIN": {
    "en-US": "%s, do you need help with something else?",
    "it-IT": "%s, posso aiutarti con qualcos'altro?"
},

"PEPPER_EXPLAIN_0": {
    "en-US": "Hi! I'm PepperShop and I am here to help you with shopping.",
    "it-IT": "Ciao! Io sono PepperShop e sono qui per aiutarti con la tua spesa."
},
```

Figure 3: Examples of static responses that Pepper can give without employing Azure OpenAI Assistant.

other clients and respond to customers requests in real time.
The core of the module lays in the usage of the Azure OpenAI GPT-4.1 Assistant, configured specifically to understand and then answer the questions made by the customers. However, not all interactions are processed through the Azure OpenAI Assistant. This is because of the usage limitations and associated costs of the service, which imposes quotas and charges based on the number of requests and token processed. To optimize this use of resources, simpler and repetitive queries, such as common greetings, basic instructions or confirmation attempts, are handled through predefined responses available in two languages. An example of these predefined bilingual responses is provided in Figure 3.
The module exposes several functions within the RAIM architecture. The most important one handles customers' requests made in natural language (namely `natural_query`), allowing the user to ask about any product-related information. An example of how this function works is shown in Figure 4.

```
action_type == "natural_query"
# Handle natural language queries
user_query = action_properties.get("query", "")
user_name = action_properties.get("user_name", "")

response_action, is_successful = self._query_assistant(
    command.from_client_id,
    user_query,
    user_name,
    "natural_query"
)

self.send_response(command_in=command, action=response_action, is_successful=is_successful)
```

Figure 4: Snippet of code showing how a natural query response is sent.

When the natural query action is received, the Customer Interaction module processes the customer ques-

tions expressed in natural language, extracts the user's query and name from the incoming RAIM command and forwards this information to the Azure OpenAI Assistant for processing. The assistant then generates a structured response as defined in the system prompt based on the product database and returns it to the module. This, then, packages the response and sends it back to the originating client, enabling real-time, personalized robot-customer interactions.
Another function terminates the interaction (namely `end_interaction`) providing a personalized farewell message.
The GPT model is also instructed to respond automatically in the language used by the customer. Furthermore, when the customer's name is provided, it is incorporated into the response to personalize the interaction. All these instructions, along with the required context for GPT and the specific JSON response format in which it should respond, are clearly defined in the system prompt provided to the model. The system prompt is the following:

```
You are Pepper, a virtual assistant for a supermarket. You can help
customers with these:
1. Product location in the store
2. Price information
3. Product descriptions

You have an available product database in JSON format:
{PRODUCT_DATABASE_JSON}
Always respond in a polite and professional manner. If the client name
is provided, use it in the response but not to greet them.
When a customer asks for information:
- Use the data from the database to provide accurate information
- If a product is not found, suggest similar alternatives
- Always provide the price and location when relevant
- Maintain a friendly and helpful tone
- Do not put questions in your responses

Your responses must be in JSON format with the following structure:
{
    "action_type": "type_of_action",
    "message": "message for the customer",
    "data": {
        'location': product location from the database
        'name': product name from the database
        'price': product price from the database
        'image_path': product image_path from the database
        'overage': product overage from the database
    },
    "is_successful": true/false
}
Note that if the request is in Italian language you have to respond in
Italian language, otherwise in English language.
```

From a social intelligence perspective, the module is designed to differentiate between first-time and returning customers, thanks to its integration with the Face Recognition module, handle social signals such as addressing the customer by name and suggest alternative products when a specific request cannot be fulfilled.
The human-to-robot interaction modalities supported in this module include voice input. On the robot-to-

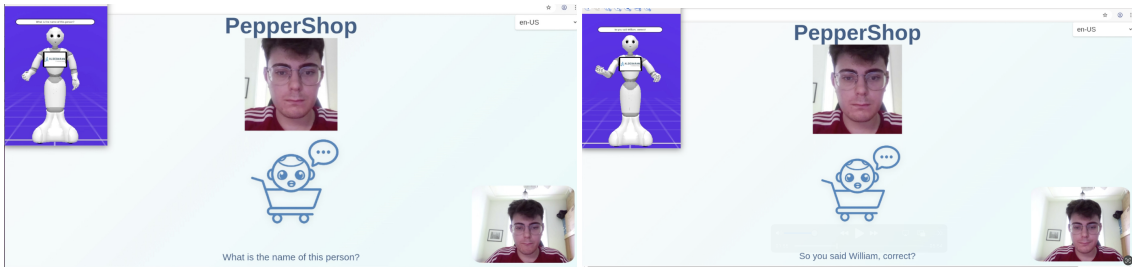
human side, Pepper responds in natural language responses, along with expressive movements and gestures to reflect its emotions. Indeed, to each Pepper's response is associated a gesture and a posture to have a major impact on the overall perception of social interactions. Additionally, visual feedback is provided through the tablet interface.
In this project, social reasoning is implemented to reflect realistic and polite customer service behavior. The robot should indeed always maintain a helpful, polite and professional tone. For example, if a customer asks for a product that is unavailable, the robot should not simply state that the product is missing, but it should comply with the request and find a similar alternative actually available in the store. This is possible thanks to the LLM's integration. This design aims to make the robot socially aware, customer-focused and adaptive to specific users, in order to also maintain coherent, contextually appropriate dialogues through the entire shopping experience.

3.1.5 Web Application

RAIM's HTTP web server, HTML, CSS and Javascript are employed to create a web application. To support multiple modalities of human-robot interaction and reduce uncertainties in communications, providing users with flexible feedback options, we used the SpeechRecognition API. This module is particularly useful in scenarios where the robot either lacks a built-in microphone or is unable to clearly hear the user. It is based on an external device (e.g., tablet, laptop), leveraging additional features like promise-based interactions and timeout management to guarantee fast and consistent responses. To interact with each module in the system, several client modules are structured, including the RAIM Client, Pepper Client, Face Recognition Client and Customer Interaction Client. Each client module exposes a set of functionalities as APIs, which can be accessed by all other modules through the RAIM communication protocol. These APIs correspond to the specific features previously described for each module and can be invoked remotely using standardized RAIM commands exploiting the async/await paradigm to allow multi-threading and avoid blocking code.

3.1.6 Complete Pipeline

The interaction pipeline is designed to integrate user recognition, natural language understanding and multimodal communication in a continuous flow.
The process begins with real-time video acquisition. If a customer comes close to the robot and a face is detected, Pepper starts explaining its functionalities and frames are captured and continuously sent to the Face Recognition Module, which analyzes each frame to detect both known and unknown users. If the customer disappears from the robot's field of view for a defined number of frames, the system automatically resets the session and prepares to identify a new user.
If an unknown face is detected, Pepper initiates a user identification process. The robot asks the user to provide their name and age through a voice-based interaction to personalize the future interactions and asking for confirmation of the provided information (Figures 5a, 5b). Pepper also requests explicit consent to store personal data in order to provide personalized assistance (Figure 6). If the user declines consent, their data will not be stored and will be deleted at the end of the session.



(a) Pepper asks the name of the unknown customer: *"What is the name of this person?"*

(b) Pepper checks if the heard name is correct: *"So you said William, correct?"*

Figure 5

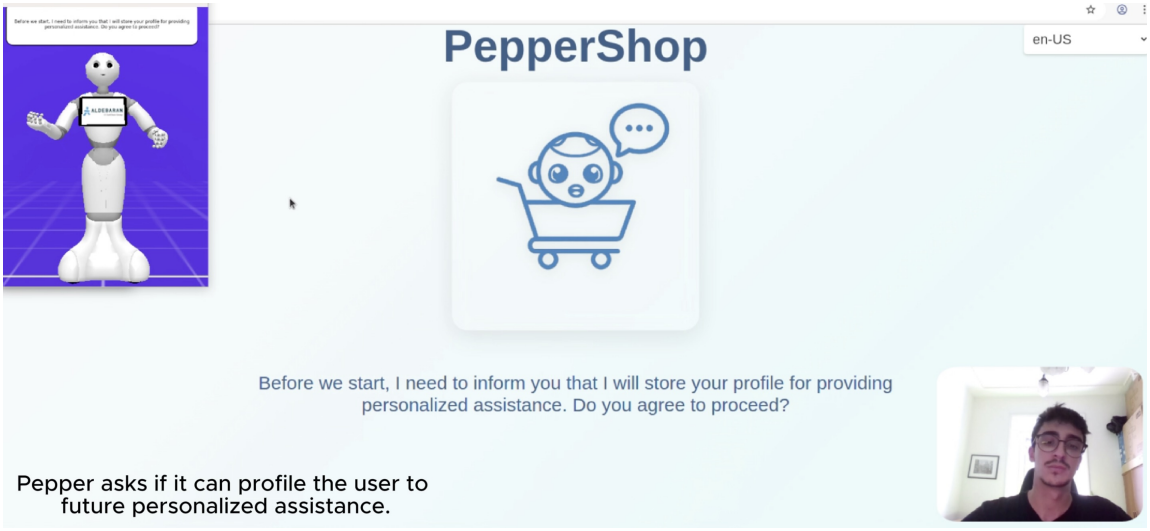


Figure 6: Pepper asks the consent to store the customer's information.

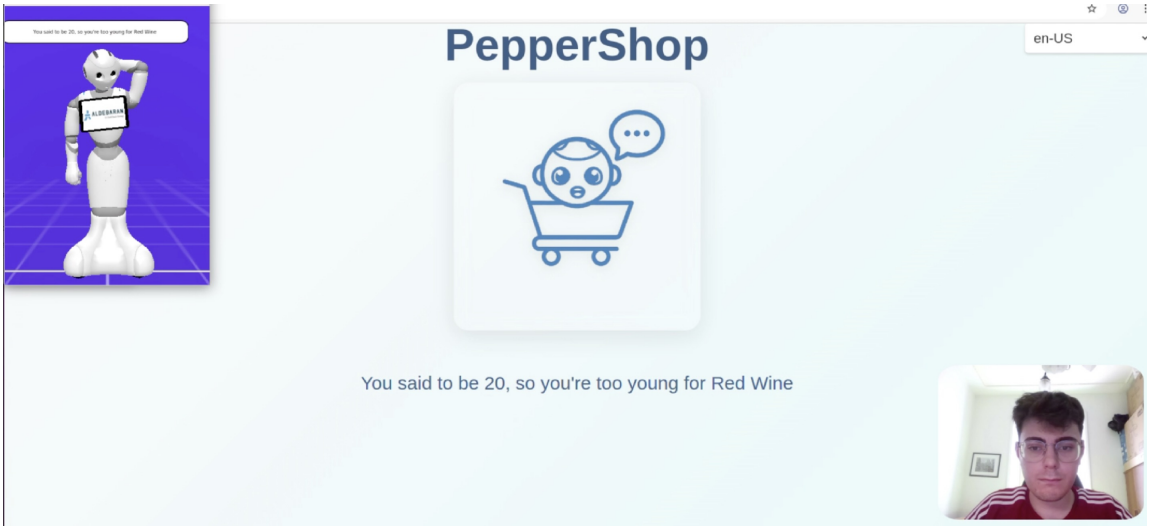


Figure 7: Pepper knows that the customer is too young, so it doesn't provide information about overage products.

However, some data may still be temporarily processed in a sort of cache memory to ensure appropriate use of the service, for example, to prevent inappropriate requests such as attempting to purchase alcohol if you are underage.
Once a customer is identified, Pepper engages in a conversational exchange. The robot asks whether the customer needs assistance and listens to the response using integrated speech recognition. The customer's request is then forwarded to the Customer Interaction module, which queries the Azure OpenAI Assistant to generate a response based on the supermarket product database. The response is provided in both spoken form via Pepper and visual form through the tablet interface, including product images.
The robot performs age-based reasoning, ensuring that product recommendations are appropriate for the user's age (e.g., alcohol restrictions). The age threshold is adapted based on the user's language and local regulations: for example, if the language is English, the minimum age is set to 21, while if the language is Italian, the threshold is set to 18 (Figure 7). The robot also handles situations where Pepper is not able to hear properly or where the customer may not respond, asking for repetitions in a socially acceptable manner and showing visual feedback when it fails to hear correctly. At the end of the session, Pepper asks whether the customer would like further assistance. If the customer declines, Pepper says a farewell message and, if consent was not given, deletes the user's data before resetting the system for the next interaction.
This pipeline allows Pepper to function as a socially aware assistant, capable of recognizing, understanding and adapting to individual users, providing practical information and demonstrating cognitive social robot interaction skills.

3.2 RBC

In the following, are described the functionality benchmarks of HRI modules.
Face Detection. The face detection module was evaluated using two complementary metrics: precision and recall to assess recognition accuracy and per-frame latency (in milliseconds) to quantify system responsiveness. Accuracy measurements were simulated under normal lighting conditions, while latency was recorded

via on-device profiling on an Intel CPU with the onboard camera.

Automatic Speech Recognition (ASR). The ASR component was benchmarked with Word Error Rate (WER) as the primary accuracy metric and end-to-end latency (in seconds) to evaluate promptness of transcription. WER simulation measures are computed first in a clean environment to establish a baseline, then under simulated "supermarket ambient" noise to test robustness to acoustic interference. Latency is due to online inference using the Web Speech API in a browser context. Measure it represents capturing the full pipeline delay from spoken input to returned transcript.

In both modules, results were obtained through a combination of AI-generated numeric simulations (with functionalities context) and small controlled simulations based on alternating group members faces and tone of voice. This approach allows for a comprehensive assessment of accuracy, latency and robustness across varied conditions.

This section describes the evaluation framework developed for RBC, a multi-agent system combining vision-language perception, structured scene understanding and symbolic task planning. The architecture is orchestrated through modular agents implemented as prompt-based GPT-4o calls and the benchmarking process measures each agent's contribution to the pipeline through structured ground truth comparisons and task-specific scoring. The system prompts of the agents are reported in Figure 8.

Description agent SP

```
You are an assistant which is able to accurately describe the content of an image.
In particular, you are able to accurately describe the content of the image to make one
understand
all the details of the image without seeing it.

Use the name in the following relations :
<ENVIRONMENT_INFO>
and nothing else without adding adjectives for the objects.

You should describe how the scene is made with high level description and precise instruction to
solve
the following task :
<TASK_DESCRIPTION>

The database containing information about the products in the scene is the following:
<PRODUCTS_DATASET>

Try to minimize the steps required and find the best plan to solve.
If the task contains ambiguity in the solution of the task, for example many objects of the same
type,
specify the position of the object in the image or in relation to other objects.

If the task involves questions about product costs or prices:
1. Identify the specific product mentioned in the query
2. Find its location based on the spatial relationships
3. Look up the product in the database and provide its price information
4. If multiple products of the same type exist, specify which one based on position

Format price information as: "PRODUCT: [name], LOCATION: [position], PRICE: [cost], DETAILS:
[details]"
```

Planning Agent SP + Prompt

```
You are a mobile robot with a base that allows you to move around the environment.
You have a robotic arm with a gripper that allows you to pick up and
place one object at time.
You know the detailed description of the scene and preliminary
instructions that can help you in the definition of the plan
with the following information :
<DESCRIPTION_AGENT>

Remember that you have to use this information just as a help.
For the task given in input you should plan a sequence of actions to solve the task.
Use univocal name given in the relations of the environment to specify the object.
Work as a Markovian agent, so you can only see the last action and the current state
of the environment.
After each step, update the state of the environment to elaborate the next step
executable in the updated environment.

If the task involves providing information (like answering questions about price,
location, or product details),
start your response with that information before planning any physical actions.

For information queries, use the RESPOND action to provide the answer to the user.
```

```
The task is <TASK_DESCRIPTION>

You must use only the following actions for the plan and nothing else:
    NAVIGATE : for the movement in the scene towards a point far from you, for example 'NAVIGATE
to the table'
    GRAB : for the action of picking up an object and specifying which object to grab, for
example 'GRAB bottle'
    DROP : for the action of placing an object, specifying where with respect to another object,
for example 'DROP bottle left to mug' or 'DROP mug right to bottle' or 'DROP pen into bag'
    PULL : for the action of pulling an object with the gripper
    PUSH: for the action of pushing an object on the ground with the base to free its trajectory
if necessary
    RESPOND: for providing information to the user, for example 'RESPOND The can behind the eggs
costs $2.50' or 'RESPOND I cannot find price information for that product'

If the task is asking for information (like price, cost, or product details), start with a
RESPOND action.
If the task involves both answering a question AND performing an action (like "How much does it
cost and can you grab it?"),
provide the RESPOND action first, then the physical actions.

Write only the actions for the plan and nothing else
```

```
You are an assistant which is able to accurately describe the content of an image.
In particular, you are able to capture the main objects present in the image and to provide all
the relations that exist between them.
These relations are described in the form of a triple (subject, relation, object)
and when you answer you are only expected to answer with triples and nothing else.

Write just the triples that are essential to solve the following task:
    <TASK_DESCRIPTION>

Use specific relation to describe the position of the objects in the scene. Do not use 'next to'
but you must
use 'right to', 'left to', 'behind', 'beside to', 'on', 'below', 'above', 'in front of'

For example, if in a scene there is a door, a table in front of the door and a book on the table
with a pen right to it, your answer should be:
    1) (table, in front of, door)
    2) (book, on, table)
    3) (pen, on, table)
    4) (pen, right to, book)
    5) (paint, behind, book)
    6) (book, left to, pen)

If the task involves identifying specific products or asking about costs, make sure to identify
all visible products and their brand names or types clearly.

Answer with only the relations and nothing else
```

Environment Agent SP (one-shot)

Figure 8: System prompts of the agents employed in the pipeline.

3.2.1 Functionality Benchmarks and Task Evaluation

The framework evaluates each agent's environment, description and planning through a weighted scoring scheme. The evaluation is structured around five key modules:

- **Environment Agent**: Extracts spatial relations and objects from the scene. Triplets of the form `(subject, relation, object)` are normalized through synonym and inverse mappings (e.g., `"beside"` → `"next to"`, `"left of"` ↔ `"right of"`), enabling robust comparison with the ground truth. Precision, recall and F1-score are computed for spatial relation extraction, while object detection accuracy is calculated separately. The score is computed as follows:

```
object_accuracy =
    len(detected_objects & ground_truth_objects) / len(ground_truth_objects)
Environment = (f1_score + object_accuracy) / 2
```

- **Description Agent**: Assesses scene descriptions generated by GPT-4o. Metrics include object mention rate, spatial keyword density and text completeness (normalized on expected length). The description score is computed as follows:

```
object_mention_rate = len(mentioned_objects) / len(ground_truth.objects)
spatial_keywords = ['left', 'right', 'behind', ..., 'near', 'beside']
spatial_mentions = sum(1 for keyword in spatial_keywords
                       if keyword in description.lower())
completeness_score = min(1.0, (len(description.split()) / 50))
Description = (object_mention_rate + min(1.0, spatial_mentions / 3)
              + completeness_score) / 3
```

- **Planning Agent**: Analyzes generated actions using keyword-matching. Action overlap with the expected ground truth is computed and missing essential primitives (e.g., `NAVIGATE`, `GRAB`) are penalized in the computation of the feasibility score. The planning score is computed as:

```
action_accuracy = action_overlap / len(expected_actions)
Planning = (action_accuracy + feasibility_score) / 2
```

- **Information Retrieval Module**: Activated only if the task includes a known product price. It compares the extracted price from the pipeline to the ground truth. It is not used for the final pipeline score, but it is considered to assess the correct data extraction from the product database by the GPT Assistant.

- **Coherence Module**: Measures internal consistency between agents. It evaluates object agreement across environment and description outputs and semantic alignment between planning and perceptual inputs. The consistency score (`object_consistency`) is computed measuring the overlap between the set of the detected object in the environment and the objects mentioned in the agent verbal description. The flag `planning_mentions_objects` checks whether any of the objects detected in the environment are mentioned in the agent's planning output. If at least one object is mentioned, it considers the planning to be semantically aligned, computing the score as:

```
Coherence = (object_consistency +
             (1 if planning_mentions_objects else 0)) / 2
```

The final pipeline score is a weighted sum of the individual modules scores:

$$\texttt{Score} = 0.3 \cdot \texttt{Environment} + 0.3 \cdot \texttt{Description} + 0.3 \cdot \texttt{Planning} + 0.1 \cdot \texttt{Coherence}$$

A task is marked as successful if this score exceeds a threshold (default: 0.7). **Partial achievements** are recognized by scoring each agent independently. This modular breakdown allows for diagnosis of performance bottlenecks: for example, a task may succeed in planning but fail in accurate environment understanding, which would result in high planning score but low overall score.

3.2.2 Ground Truth Construction and Scene Variability

Ground truth is derived through a scene parsing routine which annotates:

- a list of expected objects
- normalized spatial relations
- expected action sequences
- correct product information (e.g., price) if relevant

Object names and relations are normalized using domain-specific rules (e.g., `egg carton` vs. `carton`) and spatial relations are expanded to include both synonyms and inverses, enabling a more tolerant matching.

3.2.3 World Modeling and Agent Coherence

Scene knowledge is encoded as a dynamic set of spatial triplets, which are propagated through the pipeline to maintain referential consistency. The planner receives both a structured prompt and updated scene context, which it uses to generate high-level symbolic actions.
The evaluation loop includes timing measurements to calculate **end-to-end latency**, defined as the duration between image acquisition and plan generation, capturing system responsiveness. Evaluation results are saved in structured logs and plain-text summaries for later inspection. An example of evaluation scores from the pipeline is shown in Table 1.

Module	**Score**
Environment Agent	0.564
Description Agent	0.944
Planning Agent	1.000
Overall Coherence	0.750
Latency (seconds)	15.740
Final Pipeline Score	0.828
Success	True

Table 1: Example of scores from Multi-Agent Pipeline

4 Implementation and results

4.1 HRI

IPC Communication IPC plays a key role in enabling processes or applications to exchange data and collaborate. In this project, we implement IPC using sockets, which serve as standardized communication endpoints. This approach enables smooth and modular communication between multiple Python scripts running on different machines.

Flask HTTP Server To handle web-based communication, abstracting the HTTP interactions and allowing browsers on devices to easily connect with the system we employed Flask, a web framework for Python that enables rapid development of web applications without enforcing rigid structure dependencies. Flask simplifies URL routing and is particularly well-suited for building RESTful APIs. For its simplicity and flexibility we chose Flask to develop the HTTP Server module within the RAIM infrastructure and the Web UI to support HRI.

WebSocketsServer Library WebSockets are a communication protocol that enables real-time, bidirectional data exchange between a client and a server over a single, persistent, connection. They allow both ends to send messages asynchronously, unlike traditional HTTP request-response model, making them ideal for low-latency, interactive applications, such as sending commands to a talking robot. In our project, we used the WebSocketServer Python library, which is a specific implementation of a WebSocket server that simplifies the process of creating WebSocket-based applications in Python.

NGINX Reverse Proxy Many web-based technologies, such as the Web Speech API, require the use of a reverse proxy and Transport Layer Security (TLS) to ensure data confidentiality and integrity during Internet communications. To meet these requirements, we employed Nginx, an open source web. In our project, Nginx is employed as proxy server, acting as an intermediary between client devices and backend servers and used to implement TLS, manage port routing, serve static files efficiently and host the web application on a publicly accessible domain.

Web Speech and MediaDevices API The Web Speech API captures and transcribes spoken input directly from the browser's audio stream. To access the device's microphone and other media inputs, like cameras and microphones, we also leverage the MediaDevices API. This combination enables users and developers to interact with the robot through the web interface or via the PepperBot library. For security reasons, these web features can only be used over secure connections (i.e., using TLS). However, this does not pose a limitation in our setup, as the system is already deployed within a TLS-compliant environment through Nginx.

NAOqi APIs and Choregraphe At the core of the PepperBot module lies the NAOqi API, which provides a comprehensive set of tools for programming, controlling, interacting with and customizing humanoid robots such as Pepper and NAO. NAOqi simplifies the development of applications that enable the robot to perform a broad range of tasks, from the basic ones to more complex, interactive behaviors. This framework offers an high level of customization, including motion control, LED animations, sensor readings and more. The NAOqi architecture supports both synchronous and asynchronous interactions, making it suitable for developing interactive behaviors that react to environmental stimuli or user input in real time.
In addition to the Python SDK, NAOqi is also closely integrated with Choregraphe, a visual programming

environment that allows developers to create complex robot behaviors. Choregraphe is built on top of the same NAOqi framework and allows the combination of motions, speech and logic blocks. The combination of these two frameworks is used in the HRI scenarios, in particular to make the robot speak and express through gestures. A customized set of gestures is built exploiting these frameworks, each of them to handle different situations. For instance, in the following snippet of code it is shown the call to the PepperBot Client to make the robot say a specific message while looking confused.

```
await this.pepper.sayMove (
    message,
    PepperClient.MOVE_NAMES.confused,
    true
);
```

Dlib's Face Recognition The face recognition system is built using the open-source Dlib library, which provides high-quality facial recognition based on deep learning. Each detected face is encoded into a unique vector representation, which can then be compared against a database of known faces to either identify or verify individuals. For all of these reasons, Dlib is well-suited for real-time applications and aligns with the needs of our project, where efficiency in processing images and video streams is crucial for real-time facial feature extraction.

Azure OpenAI GPT-4.1 Assistant The project strictly relies on the integration of the Azure OpenAI GPT-4.1 Assistant, specifically configured to manage natural interactions with supermarket customers. This assistant is responsible for interpreting user queries related to product prices, descriptions and locations within the store, providing accurate and context-aware answers.
Relying on Azure OpenAI for the natural queries provides access to more powerful computational resources and allows the system to benefit from continuous model updates and improvements, minimizing response latency and maintaining real-time conversational fluidity.
Moreover, the adoption of Azure's Assistant API proved to be the most effective solution for integrating advanced conversational capabilities into Pepper. Unlike traditional Natural Language Understanding (NLU), which required extensive manual configuration and explicit state tracking, the Assistant API leverages a powerful pre-trained transformer model (GPT-4.1) capable of understanding both open-ended and structured queries with minimal setup. This approach significantly reduced development time while ensuring a more flexible and scalable conversational experience.
To use the Azure Assistant, the user must first create an Azure account and register to the Azure OpenAI service. After this, it is necessary to deploy a model instance, such as GPT-4.1 and obtain the corresponding deployment name, endpoint URL and an API key. The initialization of the Azure OpenAI client requires these credentials. The client handles communication with the custom assistant, which is then created embedding the previously described system prompt as follows:

```
client = AzureOpenAI(
    api_key=<YOUR_AZURE_API_KEY>,
    azure_endpoint=<YOUR_AZURE_ENDPOINT>,
    api_version=<YOUR_API_VERSION>)
```

```
assistant = client.beta.assistants.create(
    name="Pepper Store Assistant",
    instructions=system_instructions,
    model="gpt-4.1")
```

Once the assistant is instantiated, a conversational thread is initiated and the assistant is instructed to handle the user's query. The system enters a loop where it continuously checks the status of the assistant's run. If the status indicates completion, the system retrieves the assistant's reply.

```
run = self.client.beta.threads.runs.create(
    assistant_id=self.assistant.id,
    thread_id=thread.id)
```

```
while run.status in ['queued', 'in_progress', 'cancelling']:
    time.sleep(1)
    run = self.client.beta.threads.runs.retrieve(
        thread_id=thread.id,
        run_id=run.id
)
```

Upon successful completion, the latest assistant message is extracted and cleaned and an attempt is made to parse the reply as JSON. If successful, it is returned to the system in structured form, otherwise a fallback plain-text message is returned. This ensures robustness even in the presence of parsing issues.
Overall, the assistant serves as the intelligent backend for the HRI module, providing natural, fluent and context-aware responses while maintaining robustness and continuity across customer interactions.

4.2 RBC

The RBC system is implemented as a modular pipeline integrating perception, spatial reasoning and planning. The system relies on structured Python components and statically typed data containers through `@dataclass` definitions. This architecture emphasizes modularity, interpretability and transparency. Each module is designed to expose internal state and outputs in a form readable by humans. Vision-language interactions are structured through prompt engineering. The pipeline supports both symbolic reasoning and perception, enabling robust integration between machine learning and classical robotics principles.

Visual Perception The visual perception module forms the input layer of the system and is responsible for detecting and segmenting all relevant objects in the scene. It combines the **YOLOWorld** architecture for object detection with **EfficientViT-SAM** for instance segmentation. YOLOWorld is configured with dynamic class instantiation, which allows the system to adjust its object vocabulary at runtime based on the semantic requirements of a given task description. This capability ensures that the system remains flexible across multiple task domains.
The detection output for each object is maintained in a Python dictionary, where each key corresponds to a normalized object label and each value is a nested dictionary containing:

- `bbox:` the bounding box coordinates
- `confidence:` the object detection confidence score, ranging from 0 to 1
- `mask:` a binary mask indicating the object's segmented region in the image

After initial detection, object names are normalized using mappings to unify variations across descriptions and datasets (e.g., mapping "cans" or "tuna cans" to "tuna can"). These normalized names are propagated to all downstream modules to maintain semantic consistency.

Spatial Reasoning The spatial reasoning module transforms visual detections into symbolic knowledge by grounding the relationships between objects in structured triplets. These are expressed as `(subject, relation, object)` tuples, such as `("tuna can", "left of", "water bottle")` and are used to form a high-level symbolic scene graph.
Relations are computed using geometric heuristics based on the centroids of the detected bounding boxes. The module calculates relative positions using axis-aligned comparisons and Euclidean distances. For example, `left of` is inferred when the x-coordinate of the subject's centroid is significantly less than that of

the object, by a tunable threshold.
The resulting relations are stored in a list of tuples and enriched using synonym expansion and inverse relation inference. These expansions are implemented through dictionaries such as `RELATION_EQUIVALENCE` and `INVERSE_RELATIONS`, enabling robust matching during evaluation. Such dictionaries are defined as follows:

```
RELATION_EQUIVALENCE = {
    'beside': 'next to',
    'beside to': 'next to',
    'next to': 'next to',
    'left to': 'left of',
    'right to': 'right of',
    'at the back of': 'behind',
    'ahead of': 'in front of',
    'near': 'next to',
    'on': 'on',
    'in front of': 'in front of',
    'behind': 'behind',
}
```

```
INVERSE_RELATIONS = {
    'left of': 'right of',
    'right of': 'left of',
    'in front of': 'behind',
    'behind': 'in front of',
    'on': 'under',
    'under': 'on',
    'next to': 'next to'
}
```

Multi-Agent Pipeline The orchestration module coordinates interactions among three specialized agents: the environment agent, the description agent and the planning agent. Each of these agents is prompted using structured templates with dynamic field replacement, creating a modular and adaptive interface.
The full pipeline is executed via a single function call:

```
vlm_answer, description_answer, planner_answer =
    multi_agent_planning(image, task_description, products_dataset)
```

Here, image data is first encoded and passed to a visual-language model (GPT-4o) via the environment agent. Its output is then embedded within the prompt sent to the description agent, which generates natural language descriptions of the scene and relevant product details. These are further consumed by the planning agent to generate actionable robot instructions.
All agents communicate via structured text and image buffers. Outputs are cached using the Python `pickle` module to support traceability, reproducibility and offline evaluation.

Libraries The implementation leverages both open-source libraries and custom-developed modules for advanced task orchestration and perception:

- **Computer Vision:** OpenCV is used for low-level image manipulation (resizing, cropping, format conversion), while PyTorch serves as the inference backend for neural models. YOLOWorld (Ultralytics) handles object detection and EfficientViT-SAM provides high-resolution image segmentation.
- **Language Processing:** spaCy performs tokenization and part-of-speech tagging, enabling post-processing of generated text. Word2Vec (via Gensim) is employed to compute semantic similarity and perform class extraction from object relations detected in the image.
- **3D Integration:** Open3D enables mapping of 2D segmentation masks to 3D point clouds.

Data Structures and Serialization Structured data is managed using Python `@dataclass` constructs. The two primary classes are:

- `GroundTruth`: encapsulates the reference scenario with ground-truth objects, spatial relationships, target actions and pricing information.
- `PipelineOutput`: represents the agents' output, including environment triplets, detected objects, natural language descriptions and action plans.

5 Results

In this section we report a description of the results, showing a few situations and the corresponding behavior of the robot. Refer to the objectives in Section 1.2 to show how they have been achieved.

5.1 HRI

In nominal operational conditions, the PepperShop system follows a well-defined execution flow that demonstrates successful human-robot interaction.
Initialization and Face Recognition Phase: The system begins with camera initialization and face recognition services. When a customer approaches, the face recognition module processes video frames at regular intervals (minimum 400ms) to detect faces. The assistant starts explaining its functionalities and classifying faces. If a known face is detected, the system immediately identifies the customer and retrieves their stored profile information (name, age and eventual interaction history).
Unknown Face Registration: When encountering unrecognized faces, the system transitions to a complex multi-modal registration process involving face cropping, name collection, age verification and confirmation loops. Each step requires error handling, repetition capabilities and graceful fallback mechanisms when registration fails.

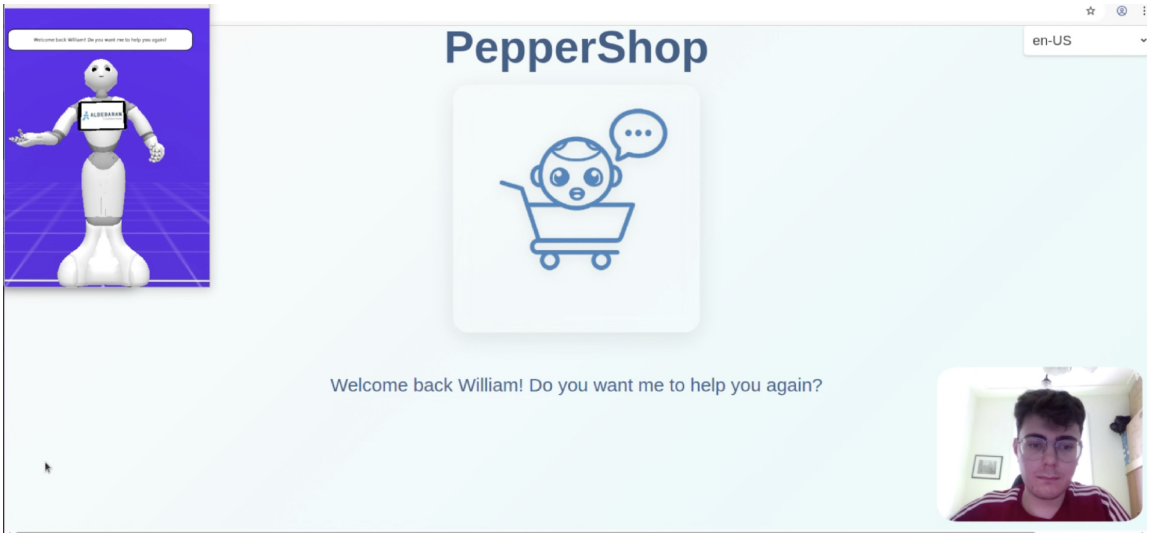
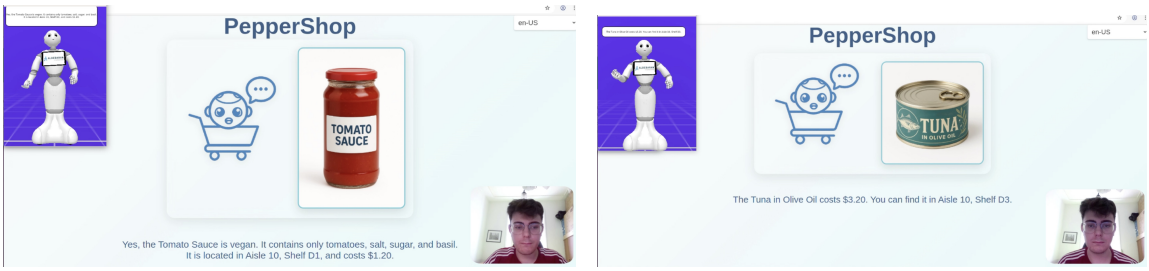


Figure 9: Pepper welcomes back the known user, asking if any help is needed.

Customer Engagement and Service Phase: Upon successful face recognition, Pepper greets the customer using personalized language patterns. For returning customers, the system uses welcoming phrases like *"Welcome back [Name]! Do you want me to help you again?"* (Figure 9), while a profiling phase starts for new customers. The speech-to-text module captures customer responses, enabling natural language interaction.
Product Assistance and Information Retrieval: When customers request product information, the system processes natural language queries through the Azure OpenAI Assistant. The assistant accesses the product database and provides contextually appropriate responses including product location, pricing, detailed descriptions and image (Figures 10a, 10b).
Interaction Conclusion: The system gracefully terminates interactions by asking if additional assistance is needed. Upon completion, it thanks the customer and resets its state to await new customers while maintaining learned customer profiles for future interactions.



(a) Pepper answers whether tomato sauce is vegan.

(b) Pepper answers with price and location of tuna.

Figure 10

The system's robustness is tested in non-nominal situations that make the reasoning abilities of the robot challenging.
Lost Customer Tracking: It is not obvious that the customer ends the interaction in a "proper way" (e.g., negative answer to additional assistance questions). Indeed they can just go away from the robot field of view. When the chosen customer moves out of camera range for more than 8 consecutive frames, the system must handle the loss gracefully. The reasoning challenge involves determining whether the customer has left or temporarily stepped away. Indeed, it works along with a counter-based approach that escalates responses from simple repetition requests to complete interaction termination after three consecutive failures. Therefore, the system implements a threshold-based approach to start a "lost customer" announcement: *"Oh no, I lost you. Come back whenever you want!"*. This requires the robot to maintain situational awareness and transition smoothly between interaction states.
Speech Recognition Failures: Audio processing failures represent a significant challenge that requires error recovery mechanisms. During all interaction phases, if the robot cannot clearly hear or understand the response (Figure 11), it will politely ask the user to repeat it. In addition, the system implements a process to allow customers to correct the responses to ensure accurate and effective communication (Figure 5b).
Age Verification and Ethical Constraints: A particularly complex reasoning scenario occurs when customers request age-restricted products. The system must cross-reference stored customer age data with product restrictions, implementing different age thresholds based on language settings (21 for English, 18 for Italian). This requires multi-modal reasoning combining customer profile data, product database information and regulatory compliance logic.

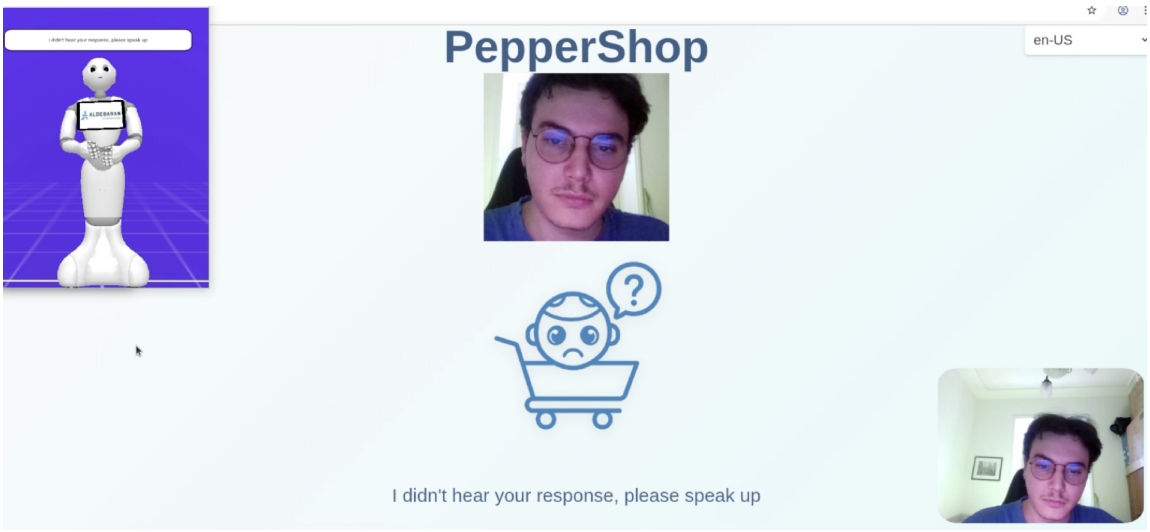


Figure 11: Pepper asks to repeat the answer, because it didn't understand the response.

Privacy Consent Management: The system handles dynamic privacy preferences, allowing customers to decline data storage while maintaining functional interaction capabilities. When customers refuse to consent, the system must provide equivalent service quality without persistent data storage and then automatically delete any collected information post-interaction. This represents a balance between personalization and privacy protection.

5.2 HRI: User study

Following the evaluation strategies defined in [24], the HRI experimental factors are **subjects**, **variables**, **hypothesis** and **protocol**.

5.2.1 Subjects

The selection of subjects is a crucial factor for HRI evaluation. A total of 80 participants were recruited, selected through stratified sampling across three age groups (18–35, 36–50 and 51–65 years old), with balanced male-female representation within each group. The sample included individuals with varying educational backgrounds. Additionally, the selection ensured a diverse level of prior technology experience, particularly in the use of smartphones and digital assistants.
Eligibility was limited to individuals aged between 18 and 65 years who possessed normal or corrected-to-normal vision and demonstrated at least B1-level English language proficiency. Participants with severe social anxiety or phobia toward robots were excluded, as were those unable to provide informed consent.
Exclusion criteria also included any prior direct experience with the Pepper robot, severe hearing impairments that could hinder speech-based interaction. Furthermore, individuals currently employed in the fields of robotics or artificial intelligence, as well as those who had participated in HRI studies within the past six months, were not considered.

5.2.2 Variables

Variables are the quantities to control and measure in the experiments. We distinguish between independent and dependent variables: the former are controlled during the experiments, while the latter are measured as outcomes.

Independent Variables The experimental design includes four independent variables.
The first is the **Personalization Level**, which varies across three conditions: (i) no personalization, where the interaction is anonymous with only generic greetings; (ii) basic personalization, involving name recognition and personalized greetings; and (iii) full personalization, which includes not only the participant's name but also stored preferences and interaction history.
The second variable is the **Privacy Consent Mode**, with two levels: in the (i) explicit consent condition, users must actively agree to data collection; in the (ii) automatic collection condition, data is collected by default.
The third variable is the **User Experience Level**, also with two conditions: (i) first-time users, who have had no prior interaction with the system and (ii) returning customers.
The fourth variable is the **Error Recovery Condition**, which includes either (i) no induced errors (i.e., a standard interaction without failure) or (ii) a scenario in which a controlled speech recognition failure occurs, followed by a demonstration of the robot's recovery capabilities.

Dependent Variables Dependent variables include objective performance metrics and subjective perception scales.
Objective performance measures include the *task completion time* (measured in seconds), the *task success rate* (expressed as the percentage of successfully completed tasks), the *number of interaction* turns required per task and the *error recovery success rate* (measured as a percentage).
Subjective perceptions are captured using several validated scales. The *Godspeed Questionnaire* [25] includes five subscales: anthropomorphism, animacy, likeability, perceived intelligence and perceived safety. The scales consist of five-point semantic differentials. The *RoSAS* (Robotic Social Attribute Scale) questionnaire [26] evaluates competence, warmth and discomfort, rated on 18-item scale to measure people's judgments of the social attributes of robots on a nine-point semantic differential.
Additionally, a custom *Privacy Comfort Scale* (7-point Likert) was developed for this study, assessing comfort with face recognition (3 items), trust in data handling (4 items) and willingness to share personal information (3 items).

5.2.3 Hypotheses

- **H1:** *Godspeed perceived intelligence* and *likeability* scores are affected by **Personalization Level**, with full personalization producing higher scores than basic personalization, which produces higher scores than no personalization.
- **H2:** *Task completion time* and *Interaction turns* are affected by **User Experience Level**, with returning customers completing tasks faster than first-time users.
- **H3:** *Privacy Comfort Scale* scores are affected by **Privacy Consent Mode**, with explicit consent producing higher comfort scores than automatic data collection.
- **H4:** *RoSAS competence* is affected by **Error Recovery Condition**, with error recovery demonstrations producing higher safety and intelligence perception scores than no error conditions.

- **H5:** *RoSAS warmth* ratings are affected by **Personalization Level**, with full personalization resulting in higher warmth ratings than basic or no personalization.

Null Hypotheses

- $\mathbf{H0_1}$: **Personalization Level** (no personalization, basic personalization, full personalization) do not affect *Godspeed perceived intelligence* or *likeability* scores.
- $\mathbf{H0_2}$: **User Experience Level** (first-time users, returning customers) does not affect *task completion time* and *interaction turns*.
- $\mathbf{H0_3}$: **Privacy Consent Mode** (explicit consent, automatic collection) does not affect *Privacy Comfort Scale* scores.
- $\mathbf{H0_4}$: **Error Recovery Condition** (no errors, induced errors) does not affect *RoSAS competence*.
- $\mathbf{H0_5}$: **Personalization Level** does not affect *RoSAS warmth* ratings.

5.2.4 Experimental Protocol

To evaluate the research questions outlined in this study, we propose a mixed factorial design that incorporates both Between-subjects and Within-subjects approaches. These factors allow to examine the complex interplay between personalization, privacy concerns and user experience.
Following a Between-subjects approach, the designed protocol tests the **Privacy Consent Mode** (2 levels: *explicit consent*, *automatic collection*), while the **Personalization Level** (3 levels: *no personalization*, *basic personalization*, *full personalization*) is tested following a Within-subjects approach. Additionally, the **User Experience Level** (2 levels: *first-time*, *returning*) is examined considering two subsequents phases of the session, with a brief interval, to simulate the realistic return customer behavior.
Given the mixed factorial design, participants will be randomly assigned to one of two experimental groups:

- Group **A** (n=40): *Explicit consent* × (*First-time customers* → *Returning customers*)
- Group **B** (n=40): *Automatic collection* × (*First-time customers* → *Returning customers*)

Once completed the profilation phase, each participant will experience all three levels of the **Personalization Level** (*no personalization*, *basic personalization*, *full personalization*) in a counterbalanced order across participants using a Latin square design. Within each personalization condition, task order will be randomized and error trial placement will be balanced across participants and conditions to prevent systematic bias and a brief interval will be included between each personalization condition to ensure ecological validity. The **Error Recovery Condition** will be randomly introduced during one of the three personalization sessions for each participant, ensuring that approximately half of the interactions across all conditions include the controlled speech recognition failure scenario.
Therefore, the experimental sessions for each group can be conducted in parallel, since the only difference lies in the profile-collection approach. This grouping is based on the idea that, once the profiling phase is completed, customers who have given explicit consent enjoy direct and seamless interactions on subsequent visits (considering the personalization levels too), whereas those who refused consent must repeat that phase. In contrast, with automatic data collection there is no need for a profiling step at any future visit, speeding up both the initial (no consent request) and all returning interaction, but this may cause discomfort regarding privacy. An experimental session of each group will last approximately 75 minutes, providing sufficient time for comprehensive data collection and will be structured in four distinct phases described in Table 2.

Phase 1: Pre-Interaction Assessment (10 min)

Participants complete the informed consent procedure and provide demographic information. This initial phase serves in assessing technology acceptance and familiarity levels. A brief anxiety and expectation assessment can be also administered to capture participants' initial emotional state and anticipations regarding the upcoming interaction. Participants then receive explanations of experimental tasks and procedures and have opportunities to ask questions or seek clarifications.

Phase 2: *First-time customers* Experimental Trials (25 min)

- Profiling phase
- 12 total interactions (3 personalization conditions × 4 tasks)
- Tasks:
  1. Product location
  2. Price comparison
  3. Nutritional info
  4. Recommendation
- One induced speech failure
- 30-second inter-trial breaks

Phase 3: *Returning customers* Experimental Trials (25 min)

- Welcome back protocol (or Profiling phase if consent was rejected in Phase 2)
- 12 total interactions (3 personalization conditions × 4 tasks)
- Tasks:
  1. Product location
  2. Price comparison
  3. Nutritional info
  4. Recommendation
- One induced speech failure
- 30-second inter-trial breaks

Phase 4: Post-Interaction Assessment (15 min)

- Godspeed + RoSAS questionnaires
- Privacy Comfort Scale
- Semi-structured interview
- Debriefing + eventual compensation

Table 2: Overview of Experimental Procedure.

Data Collection The study will employ multiple data collection approaches to ensure comprehensive measurement of both objective performance metrics and subjective user experiences. Main performance measures to collect are the *task completion time*, *task success rate*, *number of interaction* turns required per task and the *error recovery success rate*. These can be collected thought **Automated System Logging** to capture

detailed interaction metrics including timestamp recording for all interactions, speech recognition confidence scores and processing times, face recognition success/failure rates and recognition confidence, system response latencies and error occurrences and task completion status with interaction turn counts. Additionally, a **Video Recording Protocol** can be implemented using a fixed camera setup for consistent participant framing. This will enable behavioral coding for approach distance and gesture frequency, optional facial expression analysis (with additional consent) and audio recording for speech pattern analysis.

Statistical Analysis Plan Before any hypothesis testing, all dependent variables will be examined for missing data, outliers and distributional properties to ensure model validity. A three-way mixed-effects ANOVA (ANalysis Of VAriance) [27] will test the main effects of **Personalization Level** (Within-subjects), **Privacy Consent Mode** (Between-subjects) and **User Experience Level**. Degrees of freedom, F-values and exact p-values will be computed with appropriate tools, with statistical significance set at $p < 0.05$ [24]. Possible statistical analysis methods to use to test each hypothesis are in Table 3.

Hypothesis	Statistical Method	Dependent Variables	Effect Size
H1	One-way repeated measures ANOVA	*Godspeed intelligence, Godspeed likeability*	Partial eta-squared (η_p^2)
H2	2×2 mixed ANOVA (Privacy Consent Mode × User Experience Level)	*Task completion time, Interaction turns*	Partial eta-squared (η_p^2)
H3	Independent samples t-test	*Privacy Comfort Scale*	Cohen's *d*
H4	Paired samples t-test	*RoSAS competence*	Cohen's *d*
H5	One-way repeated measures ANOVA	*RoSAS warmth*	Partial eta-squared (η_p^2)

Table 3: Statistical analysis methods used to test each hypothesis. **Statistical Method** indicates the type of test applied. **Dependent Variables** are the outcome measures analyzed. **Effect Size** quantifies the magnitude of observed differences: while the p-value tells us whether a difference exists (e.g., "Is there a difference?"), the effect size tells us how large this difference is ("How big is the effect?"). For example, Cohen's *d* is used for t-tests to measure the difference between two groups, while partial eta-squared (η_p^2) is used for ANOVA tests to measure the proportion of variance explained.

Expected Outcomes and Success Criteria For HRI analysis, each hypothesis will be judged against both statistical significance ($p_{value} < 0.05$). For instance:

H1: Personalization Level effects on perception

- Success criteria: Significant main effect of **Personalization** on *Godspeed perceived intelligence* and *likeability* (F, $p < 0.05$, $\eta_p^2 \geq 0.06$) with a linear trend: full > basic > none.

H2: User Level Experience

- Success criteria: Significant main effect of returning customers faster than first-time users (F, $p < 0.05$, $\eta_p^2 \geq 0.06$).
- Practical significance: At least a 15% reduction in mean task completion time for returning versus first-time users, when previous accepted data profiling.

H3: Privacy Consent Mode

- Success criteria: Explicit consent group shows higher Privacy Comfort Scale scores (two-sample t-test, $p < 0.05$, $d \geq 0.5$).

H4: Error Recovery on RoSAS Competence

- Success criteria: Higher RoSAS competence ratings in the recovery versus no-error condition (two-sample t-test, $p < 0.05$, $d \geq 0.5$).

H5: Personalization on RoSAS Warmth

- Success criteria: Significant effect of Personalization on RoSAS warmth (F, $p < 0.05$, $\eta_p^2 \geq 0.06$) with the ordering full $>$ basic $>$ none.

This experimental design addresses the core research questions while maintaining methodological rigor and ethical standards required for HRI evaluation research.

5.3 RBC

The experimental evaluation of the multi-agent robotic pipeline reveals comprehensive performance characteristics across diverse scene configurations and task complexities, as demonstrated in the supermarket-inspired scenarios in Figure 12.An example of the final result of the pipeline is shown in Figure 13. The system demonstrates robust overall functionality with a mean pipeline score of 0.764 and an 80% success rate (8/10 runs), where success is defined as achieving a score threshold of ≥ 0.7.

Figure 12: Test scenes used for the RBC pipeline evaluation. **Left**: First scene containing a structured arrangement of everyday objects including pasta packaging, tuna cans and eggs in a carton, representing a basic spatial relationship scenario. **Right**: Second scene featuring a more complex arrangement with a water bottle, tuna cans, eggs in a carton and a tomato sauce bottle, designed to test the system's ability to handle varied object heights, occlusions and spatial relationships in a more cluttered environment.

```
Task: What is behind the eggs? Can you grab it?
Environment agent info: 1) (eggs, in, carton)
2) (carton, on, table)
3) (pasta bag, behind, eggs)
4) (pasta bag, on, table)
5) (tuna cans, left to, eggs)
6) (tuna cans, on, table)
Description agent info: The scene contains the following arrangement:

1. A carton of eggs is placed on the table.
2. Behind the carton of eggs, there is a bag of pasta (labelled "La Molisana").
3. To the left of the carton of eggs, there are two tuna cans stacked on top of each other
(labelled "Mareblu").
4. Both the pasta bag and tuna cans are also on the table.

### Task: What is behind the eggs? Can you grab it?

**Answer:**
Behind the eggs is the **pasta bag**. Yes, you can grab it as it is accessible and placed on the
table.

### Product Information:
**PRODUCT:** Pasta Fusilli
**LOCATION:** Behind the eggs, on the table
**PRICE:** $1.50
**DETAILS:** Traditional Italian fusilli made from 100% durum wheat semolina.
Planning agent info: RESPOND Behind the eggs is the pasta bag. Yes, I can grab it.
NAVIGATE to the table
GRAB pasta bag
Planning latency: 13.79 seconds
```

Figure 13: Example of the final result of the pipeline. The task is: *"What is behind the eggs? Can you grab it?"*. The Environment Agent returns the spatial information of recognized products, while the Description Agent a description in natural language of the scene and the product information retrieved by the JSON database. The final symbolic plan is provided by the Planning Agent.

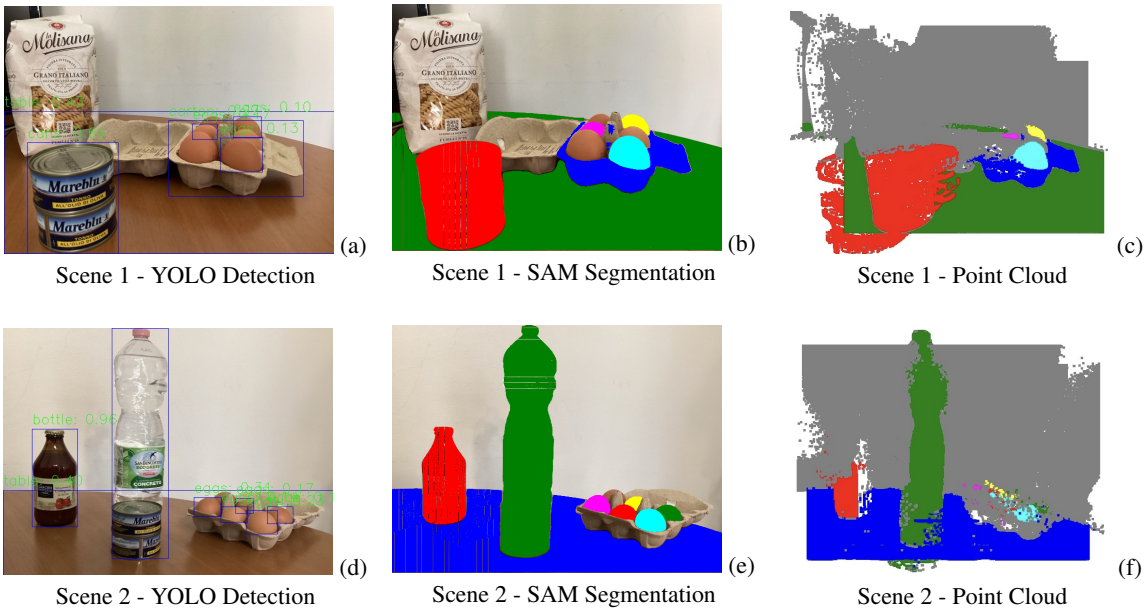
Module	**Mean Score 1^{st} scene**	**Mean Score 2^{nd} scene**	**Mean Score**
Environment Agent	0.454	0.429	0.442
Description Agent	0.960	0.855	0.908
Planning Agent	1.000	0.937	0.969
Overall Coherence	0.630	0.735	0.683
Latency (seconds)	16.236	14.374	15.305
Final Pipeline Score	0.788	0.740	0.764
Success rate	100% | 5/5	60% | 3/5	80% | 8/10

Table 4: Quantitative results of the multi-agent pipeline evaluated on two custom scenes. Each agent is scored based on task-specific metrics: the environment agent is evaluated on spatial relation F1-score and object detection accuracy; the description agent on object mentions, spatial awareness and description completeness; and the planning agent on action accuracy and plan feasibility. The final pipeline score is computed as a weighted average across these modules, plus a coherence score assessing consistency between agents. Latency measures the end-to-end time from image acquisition to the generation of the final plan. Each score is averaged over 5 runs per scene. Success rate indicates how many runs exceeded the success threshold (≥ 0.7).

The Planning Agent achieves remarkable performance with a mean score of 0.969, evaluated using task-specific action sequence accuracy metrics that measure the correctness of generated robot actions (e.g. NAVIGATE, GRAB, RESPOND) against ground-truth expected behaviors, alongside plan feasibility assessment that verifies logical action ordering and spatial constraints, for example before grabbing a product the robot

needs to navigate towards it. The Description Agent demonstrates strong scene comprehension capabilities with a mean score of 0.908, obtained through object mention completeness rates, spatial relationship awareness metrics and semantic description quality scores that represents agent's ability to generate descriptions for robotic task planning. However, the Environment Agent presents the most significant performance bottleneck with a mean score of 0.442, reflecting challenges in spatial relation detection accuracy and object localization precision. Figure 14 shows detailed performance visualizations provided in the bounding box annotations and point cloud colorization outputs.

As reported in Table 4, the computational efficiency analysis reveals a mean end-to-end latency of 15.305 seconds from RGB-D image acquisition to final plan generation, with the complete robotic perception-to-action pipeline including YOLO-based object detection with custom class assignment, SAM-based mask generation for precise object segmentation, spatial relationship grounding using geometric constraint analysis and 3D point cloud colorization for enhanced spatial understanding. The functional benchmarks employed include spatial relation F1-scores computed using precision and recall metrics for relationships such as "left of," "on," and "behind"; object detection accuracy based on overlap between detected and expected object names; action sequence accuracy evaluated by comparing generated robot commands against expected task-specific action sequences; overall coherence scoring that measures inter-agent consistency and the quality of information flow within the multi-agent architecture. Scene complexity significantly impacts system performance, with the first scene achieving perfect success (100%, 5/5 runs) and a pipeline score of 0.788, while the second scene, featuring more complex spatial arrangements of objects including



(a) Scene 1 - YOLO Detection (b) Scene 1 - SAM Segmentation (c) Scene 1 - Point Cloud

(d) Scene 2 - YOLO Detection (e) Scene 2 - SAM Segmentation (f) Scene 2 - Point Cloud

Figure 14: Outputs of the RBC pipeline for both test scenes. **Left column**: YOLO-based object detection with bounding boxes and confidence scores, showing identification of products with their respective labels. **Center column**: SAM-generated segmentation masks overlaid on the original RGB images. **Right column**: 3D point cloud visualizations with object-specific color mapping, enabling spatial reasoning and manipulation planning in the robot's coordinate system. The first row shows results for the simpler scene configuration, while the second row demonstrates performance on the more complex spatial arrangement.

water bottles, egg cartons and products positioned in challenging geometric configurations (tuna can right in front of the water bottle), resulted in 60% success (3/5 runs) and a reduced pipeline score of 0.740.
The latency measurements (16.236 seconds for scene 1, 14.374 seconds for scene 2) indicate stable computational performance across varying scene complexities, demonstrating the system's scalability for real-time robotic applications. However, the variation in success rates suggests that spatial relationship complexity, particularly in scenes involving occluded objects, represents the primary limiting factor for system reliability in real-world deployment scenarios. These findings highlight the need for enhanced spatial reasoning capabilities in the Environment Agent, while maintaining the strong performance demonstrated by the Planning and Description agents in generating feasible robot action sequences and contextually appropriate scene descriptions.
Additional benchmarking information is represented in Table 5, with a comparison of functionalities and tasks of the proposed solution. Table 6 describes the simulated results of estimated functional benchmark for perception and interaction modules. These values are AI generated estimations and do not derive from real-world experiments.

Functionality	**Product Location**	**Product Price**	**Product Description**	**Relative Position**	**Navigate & Grab**
Face Recognition	✓	✓	✓		
ASR	✓	✓	✓	✓	✓
GPT Interaction Handler	✓	✓	✓	✓	
Social Cognition	✓	✓	✓		
Object Recognition				✓	✓
Semantic Segmentation				✓	✓
Symbolic Planning				✓	✓

Table 5: Functional vs Task Benchmarking.

Module	**Metric**	**Result**	**Condition**
Face Detection	Precision / Recall	0.88 / 0.84	Normal lighting
Face Detection	Latency	120 ms/frame	Intel CPU + onboard camera
ASR	WER	15.6%	Clean environment
ASR	WER	32.4%	With supermarket ambient noise
ASR	Latency	$\sim$0.7 seconds	Online inference using Web Speech API

Table 6: Functional benchmark results for perception and interaction modules (AI-generated estimates)

6 Conclusion

Working on this integrated HRI and RBC project with Pepper has been both stimulating and rewarding. The opportunity to design, implement and evaluate a socially interactive robot in the complex setting of a su-

permarket challenged us to think holistically about perception, dialogue, planning and user experience. We particularly enjoyed working on the visual design of the user interface and the customer interaction design. In particular, we built this project with the clear idea that AI-integrated technologies are now at the forefront of Human–Robot Interaction development.

Through this project, we learned the critical importance of modular, transparent architectures. The RAIM communication core showed us how to integrate heterogeneous modules (Python 2/3, WebSockets, HTTP, NAOqi) in a scalable way. Our face-recognition and consent pipeline underscored the practical challenges of real-time identity verification and privacy management. It was also interesting to observe that not all group members shared the same views on privacy concerns. This led to the integration of a consent request feature and the design of a user testing process to evaluate the **Privacy Consent Mode**. On the RBC side, although the hardware limitations were disappointing, the prompt-based multi-agent approach, which separates perception, description and planning into independently measurable components, was particularly interesting from a benchmarking perspective. Indeed, designing an appropriate method to evaluate the pipeline was itself a challenging task.

Looking forward, several avenues for improvement emerge. From a technical standpoint, integrating a spatial SLAM module would enhance map accuracy and allow the robot to navigate through the store. On the HRI front, incorporating emotion recognition via facial expression analysis would enable truly adaptive, empathetic dialogues. For the RBC pipeline, leveraging continual learning in the description agent could reduce performance bottlenecks in novel scene configurations.

Beyond these enhancements, other technical challenges remain to be addressed. First, robust multi-robot coordination could enable swarms of assistants to collaborate on stocking or guiding multiple customers simultaneously. Second, dynamic updating of product databases in real time (e.g., detecting out-of-stock items) would improve reliability. Third, handling adversarial or ambiguous user queries calls for advances in safe LLM deployment, perhaps via structured fallback strategies.

Finally, several non-technical issues deserve careful consideration. Ethically, long-term data retention and its impact on customer trust must be balanced against personalization benefits. Philosophically, the increasing autonomy of social robots raises questions about human responsibility and agency in mixed societies. Psychologically, prolonged exposure to humanoid assistants may affect social norms and customer expectations. Economically, cost–benefit analyses are needed to evaluate deployment at scale, especially for small retailers. Societally, we must consider accessibility and inclusivity, ensuring that voice and multimodal interfaces serve users with disabilities or language barriers without bias.

In summary, this project has deepened our understanding of both the promises and difficulties of integrating HRI and RBC frameworks in real-world settings. We look forward to these next steps, confident that the lessons learned here will guide more sophisticated, responsible and human-centered robotics in the supermarket of the future.

References

[1] Umer Khan and Zuhal Erden. A systematic review of social robots in shopping environments. *International Journal of Human-Computer Interaction*, 11 2024.

[2] Niklas Eriksson, Kristoffer Adolfsson, Christa Tigerstedt, and Minna Stenius. Customer-facing social robots in the grocery store: Experiences from a field trial. pages 509–516, 04 2025.

[3] Marketta Niemelä, Päivi Heikkilä, and Hanna Lammi. A social service robot in a shopping mall: Expectations of the management, retailers and consumers. pages 227–228, 03 2017.

[4] Willy Barnett, Adrienne Foos, Thorsten Gruber, Debbie Keeling, Kathy Keeling, and Linda Alkire. Consumer perceptions of interactive service robots: A value- dominant logic perspective. volume 2014, 08 2014.

[5] Takayuki Kanda, Masahiro Shiomi, Zenta Miyashita, Hiroshi Ishiguro, and Norihiro Hagita. A communication robot in a shopping mall. *IEEE Transactions on Robotics*, 26:897–913, 10 2010.

[6] Horst-Michael Gross, H. Boehme, Christof Schröter, Steffen Müller, A. Koenig, Erik Einhorn, Ch Martin, Matthias Merten, and A. Bley. Toomas: Interactive shopping guide robots in everyday use - final implementation and experiences from long-term field trials. pages 2005 – 2012, 11 2009.

[7] Benjamin Lewandowski, Tim Wengefeld, Sabine Müller, Mathias Jenny, Sebastian Glende, Christof Schröter, Andreas Bley, and Horst-Michael Gross. Socially compliant human-robot interaction for autonomous scanning tasks in supermarket environments. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 363–370, 2020.

[8] Christopher Thompson, Haris Khan, Daniel Dworakowski, Kobe Harrigan, and Goldie Nejat. An autonomous shopping assistance robot for grocery stores. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[9] R. I. Aylott and Vince Mitchell. An exploratory study of grocery shopping stressors. *International Journal of Retail & Distribution Management*, 26:362–373, 1998.

[10] Rabi S. Bhagat. Effects of stressful life events on individual performance effectiveness and work adjustment processes within organizational settings: A research model. *The Academy of Management Review*, 8(4):660–671, 1983.

[11] Cheol Park, Easwar Iyer, and Daniel Smith. The effects of situational factors on in-store grocery shopping behavior. *Journal of Consumer Research*, 15:422–33, 03 1989.

[12] Aly Khalifa, Ahmed A. Abdelrahman, Dominykas Strazdas, Jan Hintz, Thorsten Hempel, and Ayoub Al-Hamadi. Face recognition and tracking framework for human–robot interaction. *Applied Sciences*, 12(11), 2022.

[13] Luca Garello, Giulia Belgiovine, Gabriele Russo, Francesco Rea, and Alessandra Sciutti. Building knowledge from interactions: An llm-based architecture for adaptive tutoring and social reasoning, 2025.

[14] Vittorio Perera, Tiago Pereira, Jonathan Connell, and Manuela Veloso. Setting up pepper for autonomous navigation and personalized interaction with users, 2017.

[15] Xiaohui Chen, Katherine Luo, Trevor Gee, and Mahla Nejati. Does chatgpt and whisper make humanoid robots more relatable?, 2024.

[16] Javier Sevilla-Salcedo, Enrique Fernádez-Rodicio, Laura Martín-Galván, Álvaro Castro-González, José C. Castillo, and Miguel A. Salichs. Using large language models to shape social robots' speech. *International Journal of Interactive Multimedia and Artificial Intelligence*, 8(3):6–20, 09/2023 2023.

[17] Hangyeol Kang, Maher Ben Moussa, and Nadia Thalmann. Nadine: An llm-driven intelligent social robot with affective capabilities and human-like memory, 05 2024.

[18] Vincenzo Suriani. Benchmarking. In *RBC 2025 Lecture Series, Lecture 3*, 2025.

[19] Vincenzo Suriani. Geometrical and semantical representation. In *RBC 2025 Lecture Series, Lecture 8*, 2025.

[20] Michele Brienza and Vincenzo Suriani. Using the tiago and the nao robot for projects. In *RBC 2025 Lecture Series, Lecture 12*, 2025.

[21] Francesco Argenziano, Michele Brienza, Vincenzo Suriani, Daniele Nardi, and Domenico D. Bloisi. Empower: Embodied multi-role open-vocabulary planning with online grounding and execution, 2024.

[22] Masahiro Mori, Karl MacDorman, and Norri Kageki. The uncanny valley [from the field]. *IEEE Robotics & Automation Magazine*, 19:98–100, 06 2012.

[23] Deepti Mishra, Guillermo Arroyo Romero, Akshara Pande, Bhavana Nachenahalli Bhuthegowda, Dimitrios Chaskopoulos, and Bhanu Shrestha. An exploration of the pepper robot's capabilities: Unveiling its potential. *Applied Sciences*, 14(1), 2024.

[24] Luca Iocchi. Evaluation. In *HRI 2025 Lecture Series, Lecture 7*, 2025.

[25] Astrid Weiss and Christoph Bartneck. Meta analysis of the usage of the godspeed questionnaire series. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 381–388, 2015.

[26] Colleen M. Carpinella, Alisa B. Wyman, Michael A. Perez, and Steven J. Stroessner. The robotic social attributes scale (rosas): Development and validation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, page 254–262, New York, NY, USA, 2017. Association for Computing Machinery.

[27] Philip Stoker, Guang Tian, and Ja Young Kim. *Analysis of variance (Anova)*, pages 197–219. Taylor and Francis, January 2020. Publisher Copyright: © 2020 selection and editorial matter, Reid Ewing and Keunhyun Park; individual chapters, the contributors.