TIAGo Chef Assistant: HRI & RBC Project

Ettore Branca 1965733
Francesco Caracci 1915689

October 21, 2025

All students have equally contributed to the project.

1 Introduction

Domestic assistance tasks, such as suggesting what to cook, fetching ingredients, or organizing kitchenware, are high-impact applications of service robotics because they target everyday human needs and can meaningfully increase autonomy and quality of life for elderly or mobility-limited users. The *TIAGo Chef Assistant* project explores the integration of human-robot interaction (HRI) and reliable robot behavior control (RBC) to create a simulated domestic assistant that can perceive objects and ingredients, converse with a human operator, propose context-aware recipes, and physically manipulate small kitchen items. It is implemented as an integrated system based on TIAGo in Gazebo simulation with the goal of integrating modules to:

- detect and recognize objects, ingredients and people through vision (YOLO-like module),
- interact with the user via voice/terminal in natural language and suggest personalized recipes based on the available ingredients and limitation/preferences provided by the user (semantic / LLM module),
- navigate the environment avoiding obstacles, plan and execute physically constrained paths and manipulations in environments that change dynamically. (localization + navigation + pick/place).

The central idea is to move beyond isolated demonstrations of perception, dialog, or grasping and instead deliver an end-to-end pipeline in which these capabilities cooperate: perception (through a Neural Network for real-time object detection) produces candidate ingredients, a guided human-in-the-loop dialog refines and structures that information, a semantic module (backed by a Large Language Model [2, 5].) synthesizes recipe suggestions that respect dietary preferences, and a manipulation stack executes pick-and-place actions to fetch or relocate items.

Practically, the goal is to demonstrate a cohesive TIAGo-based assistant prototype that could support everyday cooking-related tasks; scientifically, the project is intended to probe how structured interaction protocols (for example, a short Q/A session) influence the downstream semantic reasoning performed by an LLM and to evaluate how scene-management strategies and sampled path and grasp candidates affect navigation and manipulation robustness in cluttered or partially observed scenes. The dual emphasis on HRI and RBC reflects the conviction that practical service

robots must excel at both communicating effectively with people and reliably executing physical actions in uncertain environments.

Concretely, the implementation delivers a set of cooperating ROS nodes and launch configurations that realize the envisioned pipeline. A vision node produces concise, human-readable detections and short verbal feedback so that the robot can announce what it perceives. Due to the limited variety of ingredients in the Gazebo simulation, a pantry-inspection service implements a guided conversational flow: once the robot reaches the pantry, it activates a short dialog to collect ingredient lists, preferred dish type, and dietary constraints; the result is published as a structured JSON request to the semantic module, which applies lightweight dietary filtering heuristics, and attempts to generate up to a few recipe suggestions via an LLM wrapper. The natural-language front-end is constructed from a small rule-based semantic voice node, a dialog manager that handles navigation goals and arrival actions, and a terminal-based voice UI that displays robot speech and chef suggestions. On the manipulation side, the project includes a spherical grasp generator that proposes candidate gripper poses around an object centroid, a pick-and-place server that inserts a small box representation into the MoveIt planning scene and a client that performs pre-grasp motions and exposes simple services to trigger pick/place actions. An orchestrator node listens for navigation goals and, upon arrival, executes the configured arrival actions. All components are parameterized and connected via launch files that bind the navigation stack, perception, octomap, and manipulation nodes into demonstrable scenarios.

The selection of techniques is motivated by practical trade-offs between robustness, interpretability, and reproducibility. The ROS-based modular architecture was chosen to keep components decoupled and to allow focused testing; the system's inter-process interfaces (topics, services, and actions) make it straightforward to swap perception or planning modules without rewriting the whole stack. On the perception side, object detectors of the YOLO family ("You Only Look Once") [9] are widely used in robotics because they provide class labels and bounding boxes with low latency and modest engineering effort, to reflect an architectural commitment to real-time, topic-based perception that can feed both social behaviors (person detection and greeting) and task-oriented reasoning (ingredient hypothesis generation for the chef module). Using an LLM to produce recipe text takes advantage of contemporary models' ability to generate natural, human-friendly instructions. For manipulation, spherical grasp sampling combined with an active OctoMap clearing was adopted after weighing its practicality against more computationally expensive model-based grasping alternatives [12, 7].; this combination tends to increase the chance of finding collision-free approaches within MoveIt's planner while remaining simple to parameterize.

In sum, the project develops an integrated TIAGo assistant that unites navigation, perception, structured interaction, semantic reasoning, and manipulation. The architecture and technique choices are intended to strike a balance between experimental breadth and system reliability: by designing for repeatability in simulation, the work aims to be both a functional demo of an assistive robot and a reproducible platform for evaluating the interplay between HRI protocols and manipulation robustness.

2 Related Work

Relevant literature includes domestic assistive robotics, applications of LLMs and vision modules in robotics, localization, navigation and manipulation systems built on MoveIt. Our implementation follows many well-known paradigms in HRI and RBC:

- Hokuyo - 5.6m range
- Intel i7 CPU | 16 GB RAM | 500 GB SSD

Height	110-145 cm
Footprint	ø 54 cm
Arm Payload	3 Kg (without end-effector)
Battery autonomy	4-5h (1 battery)/8-10h (2 batteries)
Mounting points	On head, laptop tray and mobile base
OS	Ubuntu LTS

Figure 1: TIAGo robot functionalities overview

- modular robotic architectures (perception – reasoning – action),
- integration of a YOLO-like node as support component for vision and object detection/recognition,
- integration of LLMs as a support component for dialog and recommendations,
- use of semantic representations for places and actions (semantic places, NL-to-goal mapping).

First, the course material (RBC-2025 lecture slides) provided essential orientation and domain knowledge to work with TIAGo in simulation. In particular, the lecture on Project Platforms (RBC-2025, Lecture 2 [11]) clarified the robot's physical capabilities, coordinate frames, and typical sensor suites we had to emulate; the lecture on 'Using the TIAGo and the NAO robot for projects' (RBC-2025, Lecture 12 [10]) supplied practical instructions for selecting the correct TIAGo image, starting the Gazebo environment, and the recommended workflows for testing navigation and manipulation demos. These lectures functioned as a trustworthy, course-level guide to deal with the platform-specific tutorials, documentation and community resources for TIAGo and for the ROS/MoveIt integration in Gazebo. The official ROS TIAGo wiki and PAL-Robotics tutorials [1] contain concrete how-to material on localization and path planning, MoveIt-based pick-and-place demos, and planning with OctoMap. In other words, many of the integration decisions in our launch files and node parameters (topic names, frames, common dynamic-reconfigure knobs) were taken directly from, or cross-checked against, the TIAGo tutorials and MoveIt/Octomap examples. Beyond these primary sources, we also considered tutorial and community contributions (for example PAL Robotics blog posts and workshop papers describing practical TIAGo usage), and various MoveIt examples.

Regarding visual perception, our system integrates a YOLO-like object detection module, a choice that aligns with a vast body of work in real-time visual recognition for robotics. YOLO-

based architectures have become a de facto standard in service robotics because they provide robust, low-latency detection of everyday objects and people directly from RGB input. In our case, the module filters detections to food-related items and publishes concise symbolic labels ("vedo uova"), thus bridging perceptual and linguistic representations. This approach echoes trends in multimodal perception frameworks where symbolic summaries of sensor data are shared across reasoning and interaction layers.

Further research in semantic parsing/mapping and grounded command interpretation in symbolic or spatial representations (for example, in language-conditioned navigation and manipulation tasks) has shown that mapping verbal requests such as "go to the first table" or "pick up the red object" into spatial goals and action primitives is essential for effective collaboration between humans and robots. Our `semantic_voice_node` follows this paradigm, extracting semantic cues (action, target place, or object) from natural language utterances and translates them into navigation or manipulation goals within the robot's map frame. Systems such as FollowNet and other spatial language grounding methods have demonstrated that linking symbolic place names ("table one", "kitchen area") to metric coordinates is an effective way to bridge communication and control layers [8]. The semantic-place mapping adopted in our implementation is conceptually similar: it provides a predefined dictionary of known locations that the voice interface and task planner can query to resolve commands into concrete goals. The voice node performs light normalization and matching (e.g., numerals and Italian word forms), while disambiguation and follow-up behaviors are handled by the dialog/mission-control layer. This pragmatic mapping strategy provides a reliable, low-latency bridge between natural language and navigation commands in our Gazebo/TIAGo setup, prioritizing responsiveness, predictable failure modes, and easy integration with MoveIt-based navigation and manipulation.

Similarly, the `voice_terminal` serves as an accessible speech-in-the-loop interface, reflecting the design philosophy of many HRI frameworks where transparency and bidirectional communication are prioritized.

Next, our `semantic_chef` module draws inspiration from knowledge-based reasoning frameworks such as KnowRob [4], which use structured representations to link perception and action planning [8] , but with the main difference that our reasoning layer operates through a lightweight hybrid approach that combines symbolic heuristics (e.g., dietary filtering and structured JSON parsing) with an LLM backend. This design reflects current trends in neurally-assisted planning, such as Code-as-Policies and LLMs-as-Planners, where LLMs generate flexible natural-language suggestions while deterministic local modules enforce safety and context validity. Other ways to transform natural-language instructions and perceptual observations into actionable robot behaviors are the "Do As I Can, Not As I Say" line of work (often referred to as SayCan [2, 3]), which studies how to ground language in robotic affordances and to combine a high-level language prior with low-level affordance estimates to select feasible actions, and PaLM-E [5], which explores embodied multimodal language models that ingest visual and continuous state information alongside text to perform embodied reasoning . These papers motivate our strategy of using an LLM wrapper but avoiding to treat it as an authoritative controller: the system uses the LLM primarily for natural and user-friendly recipe synthesis and explanatory text, while applying lightweight diet and affordance heuristics locally in order to reduce ambiguity in the LLM prompt and empirically improve the quality of the generated suggestions (compared to unstructured CSV flows).

From a system integration perspective, the overall architecture resonates with modular cognitive-robotic frameworks such as ROSPlan and RoboComp, which emphasize topic-based communication, hierarchical reasoning, and clear separation between perception, decision, and actuation layers. Our

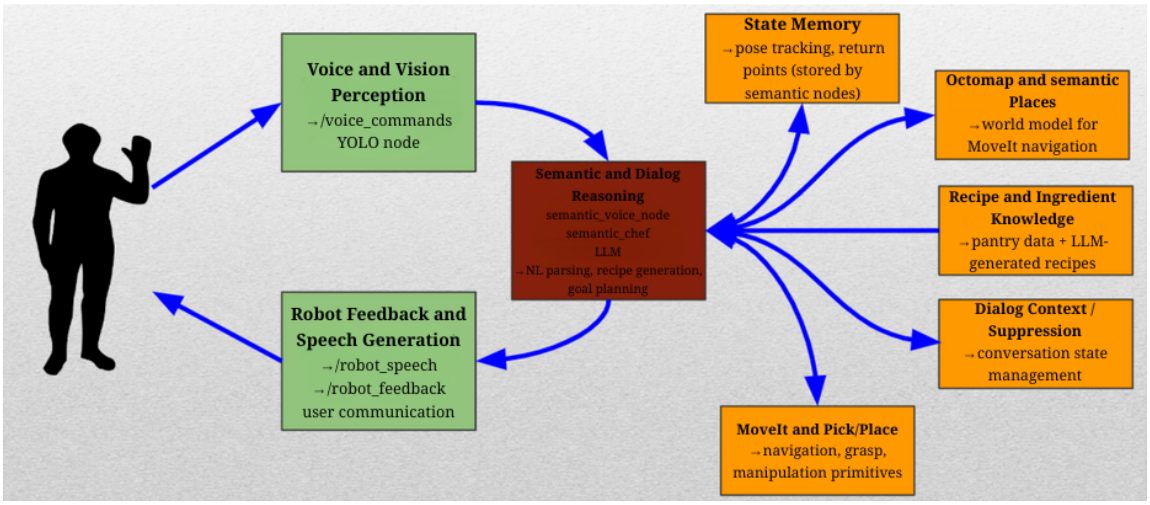

Figure 2: Functional architecture of the TIAGo Chef Assistant system, integrating perception, reasoning, and action within the HRI–RBC framework.

implementation follows these principles by exposing all functional components as ROS nodes communicating via topics, services, and actions, thus ensuring that each capability—speech processing, reasoning, or manipulation—remains independently testable and replaceable.

In related literature, manipulation success is typically evaluated through grasp success rate and motion-planning completion (standard datasets such as YCB-Video and HOPE are often employed for object recognition and grasping evaluation, but in our case, due to simulation constraints, we used the built-in TIAGo Gazebo assets and a curated set of household objects to approximate those categories), while dialogue and NL grounding tasks are assessed in terms of intent recognition accuracy or task completion rate. Although our evaluation was primarily qualitative, these metrics guided the design of our test cases and debugging sessions."

3 Integrated Solution

The TIAGo Chef Assistant integrates HRI and RBC within a unified perception–reasoning–action architecture. The system connects NL understanding, visual perception, semantic reasoning, and manipulation capabilities to enable a domestic assistant that can perceive its environment, communicate with users, and autonomously perform kitchen-related tasks such as inspecting the pantry, suggesting recipes, and handling ingredients. Each component is implemented as a ROS node and interconnected through standard topics, services, and actions, ensuring modularity and transparency in communication between the HRI and RBC layers.

3.1 HRI

The HRI subsystem governs all communicative and social aspects of interaction. H-R communication occurs primarily through textual or spoken commands published on the `/voice_commands`

topic, while R-H communication occurs through synthesized speech and status feedback published on `/robot_speech` and `/robot_feedback`. This bidirectional information flow provides both instruction and transparency, key elements of effective social interaction.

The system considers several relevant social signals and communicative behaviors. First, a YOLO-like vision node detects people, objects, and ingredients in real time. When a human is detected, the robot greets the person with a brief verbal response ("Hello human!"), a behavior that establishes social presence and awareness. Second, conversational transparency is achieved by allowing the robot to verbalize its perceptions ("I see a bottle") and intentions ("I am going to table 1"), helping users understand what the robot perceives and plans to do. Finally, structured dialogs managed by the `dispensa_inspector` and `semantic_chef` nodes enable information gathering through guided question–answer exchanges: the robot asks for available ingredients, desired dish type, and dietary constraints, leading to structured and context-aware recipe suggestions.

The `semantic_voice_node` acts as the core of the *social reasoning* layer, parsing NL commands and extracting semantic cues such as actions, locations, or targets. It leverages a predefined semantic map of known places ("table 1", "pantry", etc.) to translate symbolic references into metric navigation goals. A suppression mechanism ensures contextual coherence: when the robot is engaged in a conversation or Q/A session, the NL interpreter temporarily disables unrelated command processing to prevent conflicts.

The robot's background knowledge consists of: a static semantic map defining named locations and their coordinates, a lightweight domain knowledge base within the semantic chef that encodes ingredients, dietary preferences, and heuristic rules, and a short-term episodic memory containing recent detections and user responses. These elements form a simplified cognitive model supporting context tracking and adaptive communication.

In terms of social intelligence, the TIAGo Chef Assistant exhibits context awareness, adaptivity, and cooperative dialogue management. For instance, if the user omits an ingredient list, the robot explicitly asks for one; if dietary restrictions exclude all detected ingredients, it explains this outcome. Such behaviors demonstrate not only command responsiveness but also proactive social reasoning and repair strategies typical of human-like assistants.

3.2 RBC

The RBC subsystem handles perception, navigation, and manipulation tasks, ensuring that all robot actions are robust, reproducible, and physically feasible. Each module has been individually tested and integrated into complete task workflows involving both human interaction and autonomous execution.

Functionality Benchmarks. Each core system component was qualitatively evaluated in simulation to verify its functional robustness and integration consistency across the HRI–RBC pipeline.

The speech and dialog modules were tested for responsiveness, clarity of feedback, and robustness to repeated or ambiguous user inputs. The system was observed to maintain coherent conversational flow, correctly handling structured Q/A interactions and fallback responses in case of incomplete or noisy input (response latency between user input and robot feedback, average message delay under 200 ms, and conversation timeout handling of 40 s).

The vision subsystem, based on a YOLO-like perception node, was assessed for detection stability under varying objects, placements and lighting and camera perspectives within the Gazebo

environment. Detection outputs were monitored to ensure that ingredient and person recognition produced consistent and human-readable labels for downstream reasoning.

The navigation layer was evaluated in terms of its ability to reach semantic goals reliably while avoiding obstacles using MoveIt and CostMap navigation in Gazebo's simulated world, maintaining smooth trajectories and consistent recovery behavior in case of partial failures or blocked paths.

Finally, manipulation was tested through repeated pick-and-place trials. The grasp generation and octomap-based planning pipeline demonstrated repeatable behavior, with the robot typically able to select feasible grasp poses.

Together, these functional assessments confirm that each module performs its designated role within the integrated architecture, supporting stable perception, reasoning, and actuation loops throughout extended simulation sessions. Metrics include accuracy, latency, and recovery mechanisms, providing quantitative insight into the reliability of each functional block.

Task Benchmarks. Higher-level evaluation focuses on compound cooking tasks such as: *"Inspect pantry → generate recipe → pick an ingredient → place it on the table."* A task is considered successful if:

- the robot receives correctly user instructions and transforms them into concrete actions,
- generates a valid recipe suggestion after asking limitations and preferences to the user,
- executes at least one correct pick–and–place cycle,
- during each path performs safe and obstacle-aware navigation,
- during the whole interaction perceives correctly humans and objects.

All tests are conducted in the Gazebo simulation environment to ensure reproducibility and safety before potential deployment on a physical TIAGo platform.

World and Context Modeling. The system's internal world model combines a symbolic semantic map with a geometric octomap-based 3D occupancy grid. The semantic map encodes labeled navigation goals and regions, while the octomap continuously updates based on sensor data to reflect the presence of dynamic obstacles or newly placed objects. The `semantic_chef` complements this spatial layer with a symbolic domain of food items, recipes, and constraints, and the `dialog_manager` maintains conversational context across turns. Dynamic updates, such as moving objects or changing user intentions, are handled through octomap clearing, periodic re-mapping, and structured state synchronization among nodes.

Integration Between HRI and RBC. Each technical layer contributes directly to the system's social goals. The perception and speech subsystems maintain transparency and engagement, while the navigation and manipulation modules ensure that promised actions ("I will take the pasta") are carried out reliably. The reasoning layer bridges the two, enforcing consistency between perceived reality, semantic knowledge, and verbal communication. Together, these mechanisms enable a socially aware and operationally reliable assistant capable of cooperating with humans through both language and action.

4 Implementation

The overall implementation follows a modular ROS-based architecture, with each capability (perception, reasoning, manipulation, and interaction) encapsulated in a dedicated node or launch configuration. This design ensures that each module can be launched, debugged, and benchmarked independently while remaining part of the integrated TIAGo Chef Assistant pipeline.

4.1 Perception and Vision

Visual perception is implemented through a YOLO-like detection node that processes RGB camera streams from the simulated TIAGo head camera in Gazebo. The node publishes bounding boxes and class labels over a dedicated topic (`/darknet_ros/bounding_boxes`), which are consumed by the semantic layer for contextual reasoning. The implementation leverages existing ROS packages such as `darknet_ros` and `image_transport`, while the higher-level filtering and human-readable message generation have been implemented within the project to match the voice and dialog components.

4.2 Semantic and Dialog Reasoning

The reasoning and dialog logic are implemented across two key modules: `semantic_voice_node` and `semantic_chef`. The first node is responsible for parsing natural language utterances into actionable intents, performing lightweight semantic matching to extract target actions and places. The second acts as a reasoning and content-generation layer that interprets structured dialog results and produces recipe suggestions or textual explanations using a wrapped LLM interface.

Structured data exchange between modules is realized through JSON messages, which are published and subscribed across ROS topics:

```
{
  "ingredients": ["pasta", "tomato", "cheese", "chicken"],
  "preferences": ["vegetarian"],
  "dish_type": "main_course"
}
```

This representation provides a compact and transparent way to transfer symbolic information between perception, reasoning, and planning modules. Internally, a lightweight memory is maintained using Python dictionaries to store recognized objects, user preferences, and previously generated recipe candidates.

4.3 Navigation and Manipulation

The navigation stack relies on `move_base`, `amcl` (Adaptive Monte Carlo Localization, using laser scans and a pre-loaded static map), and the standard costmap pipeline. Predefined semantic locations (e.g., "pantry", "table 1") are represented as key–value pairs mapping symbolic names to metric coordinates in the robot's map frame. This structure is used by the dialog system to translate NL commands into navigation goals.

Manipulation is implemented as a service-based interface around MoveIt's planning scene. The `pick place server` samples grasp poses around object centroids (identified through the ARUCO markers [6], and executes pre-grasp and lift motions, while `spherical grasps server.py` computes candidate gripper poses on a sphere around the object centroid and constructs `Grasp` messages

compatible with MoveIt. In the current setup, following the official tutorials, object localization is obtained from ARUCO markers [6] placed on the simulated items: each marker provides a stable and unambiguous 6D pose that is published in the TF tree and used as the grasp target. This approach replaces the need for full 3D perception pipelines in Gazebo while ensuring consistent object references during repeated pick-and-place operations. Parameters such as pre-grasp distance, approach direction, and lift height are tunable via `dynamic reconfigure` (SphericalGraspConfig). Before each planning cycle, the system performs dynamic OctoMap clearing to maintain an up-to-date collision environment. This entire manipulation module (servers, grasp sampling, and integration with MoveIt) was fully developed within the project, whereas the underlying motion planning primitives are provided by MoveIt's standard libraries.

4.4 H–R Interface

The human–robot interaction front-end is realized through the `voice terminal.py` node, which merges speech I/O, robot feedback, and chef suggestions into a unified text-based interface. Messages are exchanged through ROS topics (`/voice_commands`, `/robot_speech`, `/chef_suggestions`) using Python `Queue` objects to ensure synchronized output and non-blocking communication. This node was fully implemented as part of the project, including conversational flow management, message deduplication, and timeouts.

4.5 System Behavior

The integrated TIAGo Chef Assistant demonstrates a coherent end-to-end behavior in simulation. When started, the system initializes all modules and announces readiness through the voice interface. Upon receiving a command such as "inspect the pantry", the robot navigates to the pantry, triggers the perception module, and activates a structured dialog to collect the list of detected or user-specified ingredients. The reasoning layer then generates one or more recipe suggestions, which are printed and spoken through the terminal interface. Finally, the manipulation module can perform simple pick-and-place actions to simulate retrieving or moving small kitchen items.

Although the system operates entirely in simulation, it exhibits realistic and socially aware interaction patterns: it communicates its intentions, requests clarification when needed, and reacts coherently to user feedback. These results confirm the feasibility of integrating perception, dialog, and manipulation within a single, semantically grounded TIAGo-based robotic assistant.

5 Results

This section reports the observed results from the integration and testing of the TIAGo Chef Assistant system. The presented outcomes correspond to the objectives defined, demonstrating that the robot can perceive, interact, reason, and act coherently within a simulated domestic environment.

5.1 HRI

From a HRI perspective, the implemented system successfully establishes a structured and interpretable communication loop between the user and the robot. When initialized, the robot announces its readiness and waits for commands. In a nominal execution, the interaction typically proceeds as follows:

Nominal scenario: The user instructs the robot with a command such as *"TIAGo, inspect the pantry"*. The `semantic_voice_node` interprets the request, queries the semantic map to resolve the symbolic place "pantry" into metric coordinates, and publishes a navigation goal. Once the robot reaches the pantry, it announces "I am at the pantry. Tell me which ingredients are available" and automatically triggers the perception node to detect nearby objects and activates the `voice_terminal` for human confirmation. The user can then specify additional ingredients or preferences, after which the `semantic_chef` generates a NL recipe suggestion and communicates it back to the user via the dialog interface. The exchange concludes with a confirmation message and the option to detect a specific object, grab it and then place it again on the table or bring it in a given location. Throughout this process, robot feedback remains informative and context-aware, using phrases such as "Hello human!", "I have reached the pantry" or "I see a bottle".

Non-nominal scenario: Several non-ideal and more challenging situations were also explored to assess the robustness and adaptability of the system under realistic disturbances. In perception-heavy scenes, where many objects overlap or partially occlude each other, the YOLO-based detector occasionally produced inconsistent classifications or wrong detections (for examples, it detects correctly a bottle and a cup in front, but labels the pack of cereals behind as a TV). Despite these imperfections, the semantic layer was able to maintain a coherent internal representation by filtering low-confidence results and relying on user confirmation through the dialog interface. Another class of challenging situations involved cases where only a few ingredients were detected or provided, combined with strict dietary or preference constraints from the user. Even in such restrictive conditions, the `semantic_chef` module, backed by the LLM, was usually able to propose at least one viable recipe suggestion that respected the given limitations, occasionally supplementing missing elements with plausible defaults. Overall, these non-nominal tests show that the integrated system handles perceptual uncertainty, environmental variability, and semantic constraints gracefully—maintaining continuity of interaction and completing tasks in a socially transparent and technically robust manner.

5.1.1 User Study

To further evaluate the social and cognitive dimensions of the system, we propose a hypothetical user study designed to assess the perceived fluency and usefulness of the TIAGo Chef Assistant's interaction flow.

Hypotheses and research questions. We hypothesize that (H1) users will perceive the robot as more helpful and intelligible when dialog flow includes explicit confirmations and fallback questions, and (H2) that the structured Q/A during pantry inspection routine increases user satisfaction and trust compared to a single-turn unstructured interaction (free CSV). Research questions include: (i) How do users rate the clarity and predictability of the robot's responses? (ii) Does the use of contextual memory (ingredients, preferences) improve perceived competence and quality rating of recipes (user Likert scale 1-5)? (iii) How do users respond to recovery behaviors when perception or reasoning fail?

Variables. The independent variables are the *dialog strategy* (structured vs. unstructured) and the *feedback style* (explicit confirmations vs. implicit responses). Dependent variables include user satisfaction (Likert-scale ratings), task completion time, number of clarification exchanges, and

qualitative assessments of trust and engagement. Null hypotheses state that dialog structuring and feedback style have no significant effect on perceived robot competence or fluency.

Experimental protocol. Participants would interact with the simulated robot in two conditions: a baseline condition using minimal dialog, and an enhanced condition using structured pantry inspection and fallback handling. Each participant would complete a short cooking-assistance scenario (e.g., requesting a recipe based on available ingredients). All interactions would be logged, and after each session, participants would complete a short questionnaire assessing usability and perceived intelligence. Collected data would be analyzed through descriptive statistics and, if sufficient participants are available, simple ANOVA tests to compare conditions. This protocol is designed to measure not only functional success but also social responsiveness and user experience, key goals of the HRI component.

5.2 RBC

In navigation tasks, dynamic obstacles were introduced into the environment (for example, chairs placed along the planned path). The robot typically detects these new obstacles through costmap updates and successfully re-plans alternative trajectories,

From an RBC perspective, the integrated solution demonstrates consistent performance across the primary task domains of navigation, perception, and manipulation. Navigation goals were executed with high repeatability within the Gazebo environment, and the system showed stable recovery from partial localization errors or dynamically blocked paths (for example, placing chairs along the planned path) through the built-in costmap re-planning mechanisms, demonstrating resilient behavior within the simulated world. The manipulation pipeline, including spherical grasp sampling and dynamic octomap clearing, produced feasible grasp trajectories and executed complete pick-and-place sequences in most nominal configurations. Failures, when they occurred, were typically due to unreachable arm poses or simulated object collisions.

The chosen world model, based on a combination of symbolic semantic places and continuously updated octomap geometry, proved effective in supporting both social reasoning and physical task adaptation. Symbolic representations enabled smooth translation from NL commands to metric goals, while the octomap allowed the robot to adapt to small environmental changes, such as newly detected obstacles. This hybrid representation thus supports both dialog-level reasoning (semantic understanding of "where" and "what") and motion-level adaptability (geometric reasoning about "how" to act safely).

Compared to related approaches, the proposed implementation confirms that a modular, knowledge-light integration of semantic reasoning and reactive planning can yield socially robust and technically reliable behaviors. Rather than introducing a new algorithmic framework, this work demonstrates how existing perception, reasoning, and control modules can be orchestrated coherently to achieve human-aware interaction and task execution within a domestic robotic assistant.

6 Conclusion

The development of the *TIAGo Chef Assistant* project has been both technically enriching and conceptually rewarding. It provided the opportunity to design and implement a complete robotic system that combines perception, reasoning, and interaction in a coherent architecture. Through this work, we gained a deeper understanding of how heterogeneous modules, such as YOLO-based

vision, MoveIt manipulation, and LLM-based reasoning, can be integrated to achieve socially intelligent and functionally reliable robot behavior. We also learned the practical importance of clear module boundaries, topic-level synchronization, and incremental testing when developing multi-component ROS systems.

From an HRI perspective, the project demonstrated how even simple dialog protocols and structured reasoning can foster natural, cooperative exchanges between users and robots. At the same time, it exposed some of the challenges in balancing transparency, reactivity, and robustness: for instance, ensuring that the robot remains responsive despite perception delays or ambiguous commands. This hands-on experience has strengthened our appreciation for the role of social reasoning and transparency in building user trust in assistive robots.

Several directions could improve the current implementation. Technically, the system could benefit from more advanced perception (e.g., fine-grained object segmentation or multimodal fusion with depth sensing), richer semantic grounding (for example, ontology-based reasoning over recipes and tools), and continuous learning mechanisms to adapt over time. Future extensions could also include real speech recognition and synthesis modules, allowing the interaction to move beyond the terminal-based interface, as well as the integration of tactile feedback or grasp learning networks for more dexterous manipulation. From a research perspective, connecting the reasoning layer to symbolic knowledge graphs or affordance-based planners could transform the system into a more autonomous cognitive assistant capable of planning multi-step tasks.

Beyond technical improvements, the project raises important non-technical reflections. Deploying service robots in domestic environments inevitably touches on ethical and societal questions, such as the preservation of user privacy, the handling of sensitive personal data (e.g., dietary preferences or voice recordings), and the potential over-reliance on automated suggestions in daily life. It also brings philosophical and psychological considerations related to human trust, social presence, and the appropriate level of autonomy that robots should exhibit in shared spaces. Economically, the feasibility of such assistive systems depends on designing solutions that are cost-effective, maintainable, and genuinely supportive rather than replacing human care or companionship.

Overall, the project confirmed that developing an assistive domestic robot is a multidisciplinary challenge—one that requires not only robust technical solutions but also thoughtful consideration of human values and societal impact. The *TIAGo Chef Assistant* represents a small but meaningful step toward such socially aware, reliable, and empathetic robotic companions.

References

[1] Tiago tutorials and documentation. `https://wiki.ros.org/Robots/TIAGo/Tutorials`. Accessed: October 2025.

[2] Michael Ahn, Anthony Brohan, Yevgen Chebotar, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022.

[3] Anthony Brohan et al. Saycan: Grounding large language models in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[4] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[5] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[6] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco J Madrid-Cuevas, and Manuel J Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. In *Proceedings of the IEEE International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 118–126, 2014.

[7] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[8] Dipendra Misra, Andrew Bennett, Valts Blukis, Emil Niklasson, Misha Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] RBC-2025 Course Staff. Lecture 12: Using the tiago and the nao robot for projects. Lecture slides, Robotics-Based Cognition 2025, 2025.

[11] RBC-2025 Course Staff. Lecture 2: Project platforms. Lecture slides, Robotics-Based Cognition 2025, 2025.

[12] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. Moveit!: An open source robotics manipulation platform. In *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Software*, 2013.

Appendix: main files and roles

File	Brief role
semantic_nodes.launch	Launches semantic nodes (semantic_chef, ingredient_recognition, semantic_voice_node).
nav_and_pick.launch	Launches Gazebo/navigation + pick servers + aruco + octomap.
mission_controller_3d.py	Orchestrator that executes arrival_actions upon reaching a goal.
pick_and_place_server.py	Server that creates scene objects and sends Pickup/Place to MoveIt.
pick_client.py	Client that runs pregrasp/play_motion, sends goals to pickup/place and exposes Empty services /pick_gui and /place_gui.
spherical_grasps_server.py	Spherical grasp generator and publisher of poses/markers.
dispensa_inspector.py	Service /dispensa_inspect for guided Q/A and structured publication to /chef_request.
ingredient_recognition.py	Vision node that publishes /detected_ingredients and /robot_feedback.
semantic_chef.py	LLM-based module that suggests recipes from CSV or structured JSON.
voice_terminal.py	Terminal interface: send commands and display robot messages.
dialog_manager.py	Simple NL manager with rules and LLM fallback.
semantic_voice_node.py	Local NL-to-action node (nav/pick/place) subscribing to detection topics.