POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

A Configuration-Driven Multi-Agent Framework for Natural Language-Based Human-Robot Collaboration

Laurea Magistrale in Mechanical Engineering - Ingegneria Meccanica

Author: Matteo Filippi

Advisor: Prof. Hermes Giberti

Academic year: 2024-2025

1. Introduction

Industrial robotics has long been dominated by rigid and deterministic automation paradigms designed for stable, highly structured environments. While this approach has proven extremely effective in traditional manufacturing, it becomes increasingly inadequate as production systems evolve toward higher variability, customization, and human involvement.

The transition formalized by Industry 5.0 promotes sustainable, resilient, and human-centered industrial ecosystems [1], where transparency, usability, and interaction quality are elevated to system-level requirements [2]. In this context, robots are no longer conceived merely as programmable machines, but as intelligent systems capable of interpreting human intent and operating in dynamic settings.

Natural language emerges as a promising interaction modality within this paradigm, as it enables users to express goals at a high level of abstraction without engaging with low-level programming details. However, linguistic instructions are inherently ambiguous, underspecified, and context-dependent. Their correct interpretation cannot rely on language processing alone, but requires explicit grounding in perception and structured internal representations of the environment. This challenge is closely related to the symbol grounding problem [3], which highlights the necessity of linking abstract symbols to perceptual and actionable entities. Consequently, effective language-driven robotic systems must rely on scene representations that encode objects, properties, and spatial relations in a structured form suitable for reasoning and action [4]. At the same time, advances in Large Language Models (LLMs) have dramatically expanded the capabilities of language-based reasoning. Transformer architectures [5], combined with empirically validated scaling laws [6], have enabled models that can perform high-level task decomposition and generate structured plans from natural language descriptions. These developments have stimulated significant interest in leveraging LLMs for robotic planning and decision-making. Nevertheless, purely language-driven or tightly coupled end-to-end architectures often exhibit weaknesses in grounding, reliability, and interpretability, particularly in embodied and safety-critical domains [7].

To address these limitations, recent research has increasingly emphasized modular and multi-agent architectures inspired by separation-of-concerns principles [8]. Instead of centralizing perception, reasoning, and execution within a

single model, such designs assign clearly defined roles to specialized components. This architectural clarity improves transparency, facilitates validation, and supports extensibility across diverse operational settings.
Despite these advances, important gaps remain in terms of explicit semantic grounding, architectural clarity, and configurability. Motivated by these challenges, this thesis proposes a configuration-driven multi-agent framework that explicitly separates language-based planning, structured scene representation, and deterministic execution. Large Language Models are confined to high-level reasoning, while grounding and control are handled through explicit representations and modular execution mechanisms. The objective is to balance expressive language-driven interaction with reliability and structural transparency in intelligent robotic systems.

2. Proposed Multi-Agent Framework Architecture

This section presents the structural design of the proposed framework, outlining its guiding principles, internal organization, and execution logic. The architecture is articulated as a coordinated multi-agent system in which planning, parameter inference, scene management, and execution are distributed across interacting components. The focus here is on how these elements are arranged, how information flows between them, and how the overall system behavior emerges from their controlled interaction.

2.1. Design Principles

The framework is grounded on four primary design principles.

- **Separation of concerns**.
  Language understanding, scene modeling, planning, parameter extraction, and physical execution are treated as distinct responsibilities. This decomposition reduces architectural opacity and facilitates debugging, validation, and future extensions.
- **Explicit grounding through structured representations**.
  Natural language instructions are never mapped directly to control commands. Instead, all linguistic references are interpreted with respect to a dynamically maintained scene representation that encodes objects, attributes, and spatial relations. Grounding is performed through structured intermediate representations that are subsequently resolved via deterministic matching against the internal world model.
- **Local LLM-based reasoning**.
  High-level reasoning is performed through locally deployed Large Language Models. While this ensures data control and reduced external dependencies, it imposes computational constraints that require structured prompting strategies and explicit validation mechanisms to preserve reliability.
- **Configuration-driven extensibility**.
  The system behavior is defined through an external configuration file specifying perception models, object attributes, action definitions, robot parameters, and LLM-related configuration settings. This design allows the framework to be adapted to different tasks and hardware platforms without requiring architectural redesign.

Together, these principles ensure architectural extensibility, data control, a clear separation between high-level reasoning and low-level control, and precise semantic grounding within a structured and verifiable framework.

2.2. System Architecture and Execution Flow

The execution pipeline, illustrated in Figure 1, begins with the loading of the configuration file, which defines the system setup and operational constraints. These specifications guide the Scene Representation and Management Agent in acquiring the current workspace state through the perception module and constructing a structured internal representation of the environment. This representation is then converted into a textual scene description that summarizes the objects and relations relevant to language-driven reasoning. The Planning Agent receives the user instruction together with the corresponding textual scene description and generates a complete operative plan expressed as an ordered sequence of primitive actions, formatted according to the set of executable capabilities defined in the configuration file. Before any physical execution is initiated, the complete plan is subjected to a user validation phase and a full-sequence simu-

lation procedure to ensure internal consistency and feasibility. In this step, the system processes the entire action sequence to ensure consistent parameter derivation. It also verifies that the internal scene representation remains coherent with the simulated evolution of the environment. During this stage, each primitive action is processed sequentially. The Action Selection Agent maps the current action description to the corresponding executable module, while the Action Parameter Extraction Agent produces the parameters required for its execution. After the action is simulated, the Scene Representation and Management Agent updates the internal scene state to reflect the expected outcome, ensuring that subsequent actions are evaluated against a consistent and updated representation. Physical execution is initiated only after the entire plan has been simulated and all relevant parameters have been successfully extracted, ensuring that no unresolved symbolic dependencies reach the control layer.

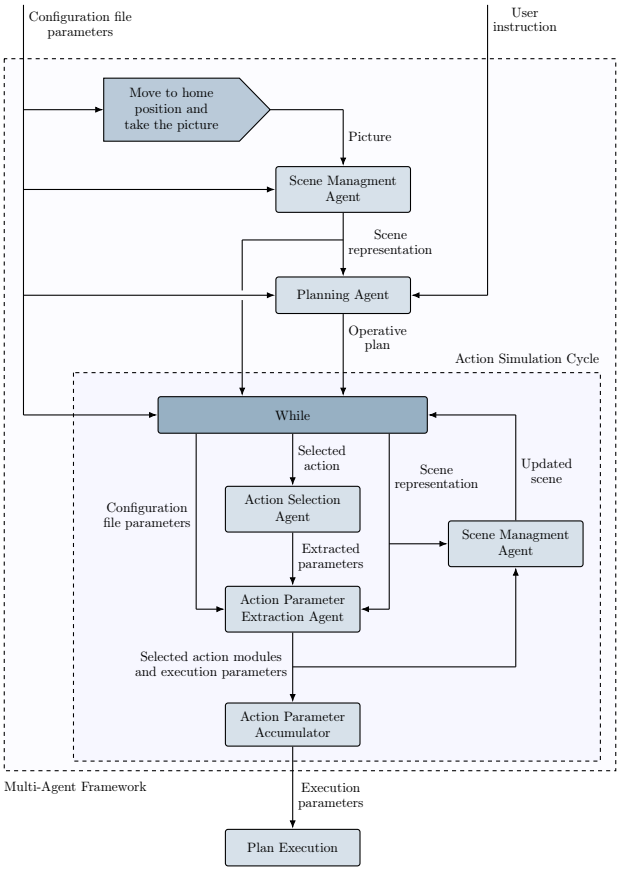

Figure 1: System architecture and execution flow of the proposed multi-agent framework, highlighting the structured flow of configuration and task-related information across the pipeline.

2.3. Multi-Agent System Design

The overall architecture is structured around four specialized agents.

Scene Representation and Management Agent
This module constructs and maintains the internal world model from perception outputs and updates it after each action simulation. It also provides the structured and textual scene representations required for planning and grounding.

Planning Agent
The Planning Agent receives the user instruction together with the corresponding textual scene description and the set of available action primitives defined in the configuration file, and produces a structured operative plan. Its internal operation is organized into two sequential stages. The first stage, illustrated in Figure 2, implements a proposal–validation–correction mechanism based on three coordinated LLM calls. An initial call generates a candidate explicit plan, a subsequent call verifies its consistency with both the instruction and the scene context, and, if necessary, a corrective call refines the output. This step ensures that all required primitive actions are explicitly identified before execution. In the second stage, the internally validated plan is converted into a canonical sequence of primitive actions aligned with the predefined action set and configuration-defined format.

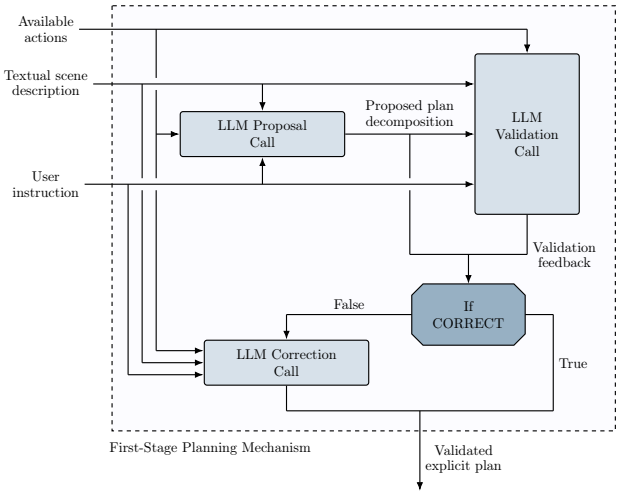

Figure 2: First-stage planning mechanism implementing the proposal–validation–correction loop based on user instruction, textual scene description, and available actions.

Action Selection Agent
The Action Selection Agent maps a primitive action to its related executable module. Selection is primarily deterministic, with constrained LLM support only in ambiguous cases, ensuring that only predefined primitives can be invoked.

Action Parameter Extraction Agent
Given a selected action, this module extracts the parameters required for its execution. The extraction process is clearly application-dependent and may vary across primitives according to the structural requirements of the execution layer.

3. Experimental Setup and System Implementation

The proposed framework was subsequently validated in a structured tabletop manipulation scenario to assess its effective operation in a practical robotic system and to verify the correct coordination among the different agents.

3.1. Hardware and Perception Setup

The experimental platform consists of a Techman TM5-700 collaborative manipulator, controlled through the TMFlow environment, which converts high-level action commands into executable trajectories. The robot is equipped with an OnRobot VGC10 electric vacuum gripper for object manipulation and an Intel RealSense D435i RGB-D camera providing aligned color and depth data for scene perception.
During the perception phase, fine-tuned YOLO segmentation models detect the object categories used in the setup (cubes and slots). Detected instances are then instantiated in the previously mentioned structured scene representation, where unique identifiers and task-relevant attributes are deterministically assigned to maintain a consistent world model.

3.2. System Configuration and Action Set

The behavior of the system is defined through an external configuration file, enabling a flexible configuration-driven architecture. The file specifies the perception models and related parameters, the selected local LLM, robot settings, object classes, and the set of admissible action primitives. For the considered application, four primitives were implemented: `pick`, `place`, `pour`, and `mix`, each linked to a predefined execution routine and a canonical textual structure enforced during the second planning stage. Each action is also associated with a deterministic update rule defining how the internal scene representation is modified after execution.

4. Experimental Results

The framework was evaluated at three distinct levels: a comparative analysis of locally deployed LLMs, planning performance assessment, and end-to-end system reliability evaluation.

4.1. Comparative Evaluation of Local LLMs

Four locally deployed instruction-tuned LLMs were evaluated as reasoning backends:

- `Qwen2.5-7B-Instruct-AWQ`
- `Mistral-7B-Instruct-v0.2-AWQ`
- `Meta-Llama-3.1-8B-Instruct-AWQ-INT4`
- `DeepSeek-Coder-6.7B-Instruct-AWQ`

A total of 500 tests were conducted on the Planning Agent and 500 on the Action Parameter Extraction Agent (350 object grounding and 150 mixing speed extraction cases), under identical prompting and configuration settings, to account for the inherent non-deterministic behavior of these LLM-based components. `Qwen2.5-7B-Instruct-AWQ` achieved the best overall performance. Its corresponding results are reported in Table 1.

	Estimated Accuracy	Inference Time [s]
Planning Agent	87.00%	2.38
Object Grounding	92.88%	0.75
Mixing Speed Extraction	100%	0.06

Table 1: `Qwen2.5-7B-Instruct-AWQ` model performance across the evaluated agents. The illustrated estimated accuracies are associated with the following Wilson CIs: [0.835, 0.899], [0.897, 0.951], and [0.975, 1.000], respectively.

A scaling analysis further examined how agent performance varies with model size. The selected 7B model was compared with its smaller variants, `Qwen2.5-3B-Instruct`,

`Qwen2.5-1.5B-Instruct`, and `Qwen2.5-0.5B-Instruct`. Selected results of this comparison are reported in Figures 3 and 4.

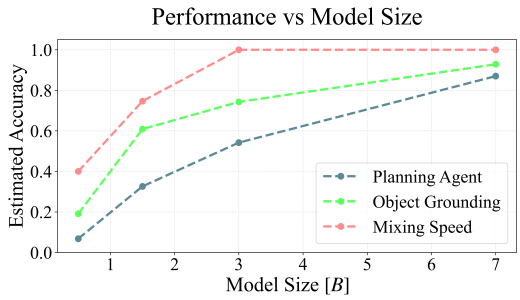

Figure 3: Achieved estimated accuracy across Qwen model sizes (0.5B, 1.5B, 3B, and 7B parameters) for the Planning Agent, Object Grounding, and Mixing Speed Extraction modules.

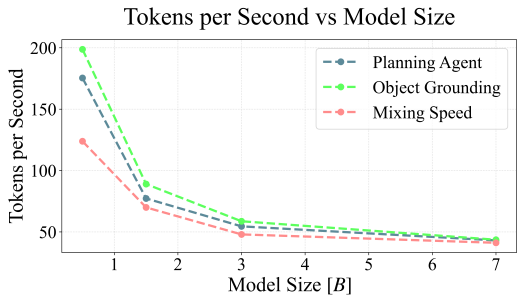

Figure 4: Tokens-per-second comparison across Qwen model sizes (0.5B, 1.5B, 3B, and 7B parameters) for the Planning Agent, Object Grounding, and Mixing Speed Extraction modules.

4.2. Planning Agent Internal Performance Analysis

The internal performance of the Planning Agent was evaluated using the 7B Qwen 2.5 configuration across the same 500 planning instructions mentioned in Section 4.1. The first stage (explicitization) achieved a raw accuracy of 74.6% based solely on the first LLM proposal call, corresponding to a 95% Wilson CI of [0.706, 0.782], which increased to 87.2% after the complete proposal–validation–correction cycle, with a 95% Wilson CI of [0.840, 0.899]. This corresponds to a 16.89% relative improvement, confirming the effectiveness of coordinated refinement. Most failures were due to redundant actions, incorrect object descriptions, and missing actions, as illustrated in Figure 5. The second stage (primitive normalization) was evaluated only on plans validated as correct in the first stage, achieving

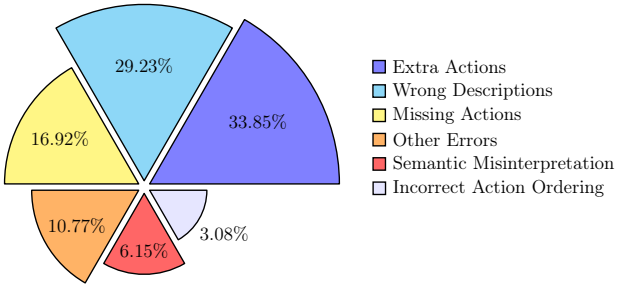

Figure 5: First-stage planning failure distribution by error category: extra actions, incorrect object descriptions, missing actions, semantic misinterpretation, incorrect action ordering.

96.1% accuracy (95% Wilson CI: [0.936, 0.976]) and confirming the structural robustness of the normalization process once a coherent explicit plan is available. The main second-stage failure modes are illustrated in Figure 6.

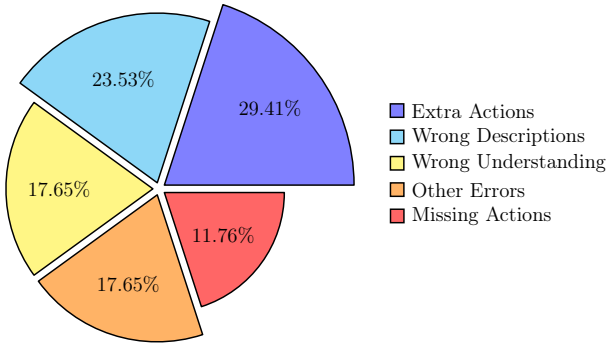

Figure 6: Second-stage planning failure distribution by error category: extra actions, incorrect object descriptions, semantic misinterpretation, missing actions, other errors.

4.3. End-to-End System Evaluation

A total of 150 complete experiments were conducted on the physical robotic platform, covering the entire pipeline from instruction interpretation to final execution. The system successfully completed 121 tasks, corresponding to an observed success rate of 80.67% (95% Wilson CI: [0.739, 0.857]). Failure analysis showed that most errors originated in the first planning stage (15 cases), followed by action execution issues (7 cases) and second-stage planning errors (4 cases). Object grounding and mixing speed extraction errors were instead comparatively rare (2 and 1 cases, respectively). The full end-to-end failure classification is reported in Figure 7.

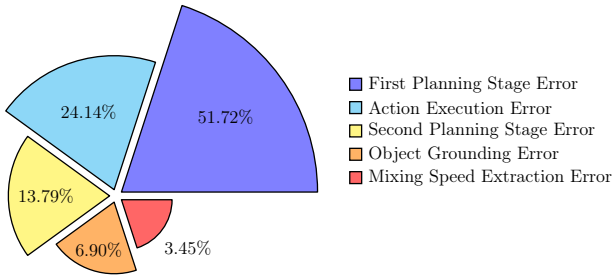

Figure 7: End-to-end failure distribution across error sources: first planning stage, action execution, second planning stage, object grounding, and mixing speed extraction.

5. Conclusions and Future Developments

This thesis presented a configuration-driven multi-agent framework for language-guided robotic manipulation, separating reasoning, scene modeling, and deterministic execution to ensure stable and consistent physical behavior.
The experimental evaluation shows that overall reliability is mainly determined by the first planning stage, while subsequent components, including second-stage normalization and parameter extraction, exhibit greater robustness. This reflects the intrinsic complexity of semantic explicitization and underscores the effectiveness of the proposal–validation–correction strategy in mitigating planning errors. Explicit scene modeling combined with structured grounding reduces ambiguity propagation from language to action, preserving alignment between symbolic reasoning and the physical environment. Finally, the modular architecture enhances interpretability, debuggability, and extensibility, enabling independent performance evaluation of components and systematic refinement.
Future developments include the integration of feedback-driven replanning for real-time adaptation, the introduction of interactive dialogue for instruction clarification, and the incorporation of augmented reality tools for visual plan preview. Extending the framework to heterogeneous platforms and multi-robot scenarios allows a better assessment of its scalability and generality. Overall, the work demonstrates that constrained language reasoning combined with explicit world modeling and modular design constitutes a reliable foundation for intelligent robotic manipulation systems.

References

[1] Directorate-General for Research and Innovation (European Commission), Breque, Maija, De Nul, Lars, and Petridis, Athanasios. *Industry 5.0: towards a sustainable, human centric and resilient European industry.* Publications Office of the European Union, 2021.

[2] Michael A. Goodrich and Alan C. Schultz. Human–Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, January 2008.

[3] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, June 1990.

[4] Andrzej Pronobis and Patric Jensfelt. Semantic mapping with mobile robots. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4604–4611. IEEE, September 2011.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. *Scaling Laws for Neural Language Models.* January 2020.

[7] Jiaqi Wang, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Bao Ge, and Shu Zhang. Large language models for robotics: Opportunities, challenges, and perspectives. *Journal of Automation and Intelligence*, 4(1):52–64, March 2025.

[8] Edmund H. Durfee. Distributed Problem Solving and Planning. In Michael Luck, Vladimír Mařík, Olga Štěpánková, and Robert Trappl, editors, *Multi-Agent Systems and Applications: 9th ECCAI Advanced Course, ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001 Prague, Czech Republic, July 2–13, 2001 Selected Tutorial Papers*, pages 118–149. Springer, Berlin, Heidelberg, 2001.