UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Dottorato di ricerca in Ingegneria dell'Innovazione Industriale

Ciclo XXXVIII

Towards Human-Centered Robot Programming: From Admittance-Based Hand Guidance to AI-Driven Interaction

Candidato: Matteo Nini

Relatore (Tutor): Prof. Cesare Fantuzzi

Coordinatore del Corso di Dottorato: Prof. Franco Zambonelli

"Προσποιοῦ ἕως ἂν κατορθώσῃς."

- Aristotle

Abstract

This PhD research explores methodologies to simplify robot programming by progressively integrating physical and cognitive human–robot interaction. The work began with the identification of the dynamic parameters of a 6-DoF industrial manipulator, providing a reliable dynamic model for advanced control design and validation.

An adaptive admittance control framework was then developed to enable safe hand-guiding. The control law adjusted inertia and damping parameters in real time according to the relative kinetic energy between the human and the robot, estimated through a vision-based system. This allowed operators to physically demonstrate trajectories that were later autonomously reproduced, effectively simplifying robot teaching.

The approach was further extended to collaborative robotics, where both algorithmic and optimization-based adaptive controllers were implemented. These controllers incorporated an energy-tank formulation and Power-and-Force-Limiting (PFL) to bound the relative kinetic energy and ensure safe, stable interaction during close proximity.

Finally, artificial intelligence methods were applied to achieve cognitive-level programming, using large language models to translate natural-language instructions into executable robot actions. Overall, the research outlines a coherent pathway from model-based identification to adaptive and AI-driven programming, contributing to safe and human-centered automation within the Industry 5.0 framework.

Contents

Summary **17**

Summary of Chapter 3: Dynamic Modeling and Parameter Identification . 20

Summary of Chapter 4: Distance-Aware Lead-Through Programming on Industrial Robots 22

Summary of Chapter 5: Energy-Based Variable Admittance Control for Collaborative Robotics 24

Summary of Chapter 6: Artificial Intelligence for Intuitive Robot Programming 26

Summary of Chapter 7: Results, Discussion, and Synthesis 27

Summary of Chapter 8: Conclusions and Future Outlook 29

1 Introduction **33**

1.1 Research Context and Motivation 34

1.2 Problem Statement 36

1.3 Research Objectives and Questions 37

1.4 Research Methodology and Approach 40

1.5 Thesis Outline 41

2 Literature Review **45**

2.1 From Industrial Automation to Collaborative Programming 46

2.2 Dynamic Modeling and Parameter Identification 48

2.3 Admittance for Physical Human–Robot Interaction 49

2.4 Safety Paradigms in Physical Human–Robot Interaction: SSM and PFL 51

2.5 Energy- and Passivity-Based Control Frameworks 52

2.6 Language-Based Robot Programming as Cognitive Interface 53

2.7 Synthesis and Research Gaps 55

3 Dynamic Modeling and Parameter Identification of an Industrial Robot 57

3.1 Scope and Role of the Dynamic Model 58

3.2 Kinematic and Dynamic Modeling of the Industrial Manipulator . . . 59

3.2.1 Generalized Coordinates and Kinematics 59

3.2.2 Rigid-Body Dynamic Model 59

3.2.3 Linear Parameterization for Identification 60

3.3 Dynamic Parameter Identification Procedure 61

3.3.1 Linear Regression Formulation 61

3.3.2 Separation of Static and Dynamic Identification 62

3.3.3 Regression Matrix Reduction and Identifiability 62

3.3.4 Least-Squares Estimation 63

3.3.5 Validation and Role within the Dissertation 64

3.4 Experimental Setup and Data Acquisition 64

3.4.1 Industrial Robot Platform 64

3.4.2 Measured Signals and Sampling 65

3.4.3 Static Identification Data Collection 66

3.4.4 Dynamic Identification Trajectories 66

3.4.5 Data Preprocessing and Consistency Checks 67

3.5 Validation Results and Discussion 67

3.5.1 Torque Reconstruction Accuracy 68

3.5.2 Interpretation of Residual Errors 69

3.5.3 Implications for Safety-Oriented Analysis 70

3.5.4 Limitations and Scope of Validity 71

3.5.5 Summary and Transition to Interaction Control 71

4 Admittance-Based Lead-Through Programming under Distance-Based Safety Supervision 73

4.1 Physical Lead-Through Programming as an Interaction Problem . . . 75

4.2 Admittance Control Formulation for Industrial Lead-Through Programming 77

4.3 Passivity Preservation Under Time-Varying Admittance via Energy Tanks 78

4.3.1 Energy Tank: State, Storage Function, and Power Balance . . 79

4.3.2 Passivity Constraint and Admissible Inertia Variation 79

4.4 Vision-Based Human State Estimation and Geometric Interface . . . 81

4.4.1 Human and Robot Geometric Abstraction 81

4.4.2 Intentional Contact and Selective Human Modeling 82

4.4.3 Minimum-Distance and Closest-Point Computation 82

4.4.4 Perception-to-Supervision Interface 84

4.4.5 Uncertainty and Conservativeness 84

4.5 Safety Metrics and Constraint-Based Admittance Parameter Adaptation 85

4.5.1 Relative Kinetic-Energy Constraint with Distance Modulation 85

4.5.2 Worst-Case Extraction, Scaling, and Parameter Update 87

4.6 Experimental Validation of Constraint-Based Lead-Through Programming 88

4.6.1 Experimental Setup and System Configuration 88

4.6.2 Validation Protocol 90

4.6.3 Behavior of Distance Modulation and Energy Constraints . . . 90

4.6.4 Admittance Parameter Adaptation and Interaction Feel 92

4.6.5 Discussion of Experimental Results 93
4.7 Discussion, Scope, and Limitations 93
4.7.1 Interpretation of the Safety Paradigm 93
4.7.2 Distance Modulation versus Distance Domination 94
4.7.3 Architectural Separation and Industrial Compatibility 94
4.7.4 Limitations and Outlook 95

5 Energy-Based Variable Admittance Control for Collaborative Robotics 97

5.1 Power and Force Limiting as an Energy Constraint 98
5.1.1 Transferred-energy bound in ISO/TS 15066 98
5.1.2 Projected motion quantities and directionality 99
5.1.3 Continuous safety residual and feasibility condition 100
5.2 Apparent Mass and Safety Coupling in Admittance Control 100
5.2.1 Apparent mass induced by admittance inertia 101
5.2.2 Velocity evolution under admittance damping 101
5.2.3 Implications for safety-oriented parameter adaptation 102
5.3 Passivity Preservation Under PFL-Driven Admittance Adaptation . . 102
5.4 Algorithmic Safety Adaptation: Nominal Interaction vs. Safe Correction 104
5.4.1 One-step safety gating and energy-loss scaling 105
5.4.2 Nominal interaction mode: ratio-preserving inertia–damping co-adaptation 105
5.4.3 Safe correction mode: feasibility restoration by damping injection 106
5.5 Optimization-Based Safety Adaptation Under PFL Constraints . . . 109
5.5.1 Decision variables and discrete-time parameter updates 110
5.5.2 One-step prediction of projected robot motion and feasibility . 111
5.5.3 Constraints: bounds, rates, and passivity-supervised inertia variation 112

5.5.4 Objective function and problem formulations 113
5.5.5 Implementation note and optimization-loop structure 115
5.6 Experimental Evaluation and Comparative Discussion 116
5.6.1 Experimental setup and protocol 116
5.6.2 Results with algorithmic safety adaptation 117
5.6.3 Results with optimization-based safety adaptation 119
5.6.4 Qualitative comparison and discussion 121

6 Artificial Intelligence for Intuitive Robot Programming 125

6.1 Large Language Models as Programming Interfaces, Not Control Systems 127
6.2 AI–ROS2 Interface Design: A Deterministic Behavioral Core Between Language and Action 129
6.2.1 Module Decomposition and ROS2 Communication Contract . 129
6.2.2 Internal Structure of the Behavior Core 130
6.2.3 Discrete Perception–Execution Cycles and Deterministic Flow 132
6.2.4 Limitations due to perception uncertainty 132
6.3 Semantic Understanding and Context Awareness via Structured Task Plans 133
6.3.1 TaskPlan JSON Schema: Fields, Roles, and Execution Semantics 133
6.3.2 Representative Plan Instance (Palletization Loop) 134
6.3.3 Validation Contract and Failure Containment 136
6.4 Experimental Demonstrations of Language-Guided Robot Programming 136
6.4.1 Experimental Setup and Execution Pipeline 137
6.4.2 Representative Language-Guided Programming Scenarios . . . 137
6.4.3 Qualitative Evaluation and Observations 138
6.4.4 Discussion of Scope and Limitations of the Experimental Validation 139
6.5 Limitations, Risks, and Explicit Non-Claims 140

7 Results, Discussion, and Synthesis **143**

7.1 Summary of Experimental Outcomes 143

7.1.1 Dynamic identification outcomes: accuracy as an epistemic instrument, not a control dependency 144

7.1.2 Industrial lead-through outcomes: distance-supervised compliance with energy-consistent adaptation 144

7.1.3 Collaborative interaction outcomes: PFL safety reasoning with energy-based variable admittance 145

7.1.4 LLM-based programming outcomes: bounded action space and deterministic execution, not certified safety 145

7.2 Comparative Analysis 146

7.2.1 Shared axes of comparison 146

7.2.2 What "safe and easy programming" means in this dissertation 147

7.3 Theoretical and Practical Contributions 148

7.3.1 Structured parameter identification via static–dynamic separation 148

7.3.2 Energy-consistent adaptation: variable admittance as a safety-relevant degree of freedom 149

7.3.3 Bounded autonomy for language programming: deterministic execution as the safety-relevant boundary 149

7.4 Limitations and Lessons Learned 149

7.4.1 Limits of distance-based supervision in industrial lead-through 149

7.4.2 Limits of PFL reasoning under modeling and contact assumptions 150

7.4.3 Limits of LLM "safe-ish" programming 150

8 Conclusions and Future Outlook **151**

8.1 Industrial Implications 152

8.2 Future Research Directions 153

Nomenclature **155**

Publications **159**

Acknowledgements **161**

List of Figures

1.1 Conceptual roadmap of the thesis, illustrating the progression from model-based foundations to adaptive, safety-aware interaction control and cognitive programming interfaces. 42

2.1 Conceptual positioning of the three complementary robot programming modalities investigated in this thesis. For each modality, the figure distinguishes the programming channel, the interaction-control layer, and the primary safety grounding (SSM supervision for separation-based physical programming; PFL constraints for contact-capable physical interaction; structured deterministic execution and runtime checks for language-based programming). 47

3.1 Overview of the dynamic parameter identification workflow adopted in this thesis. The procedure is structured into a static identification phase, targeting gravity-related parameters under quasi-static conditions, and a dynamic identification phase, focusing on inertial and friction-related effects using excitation trajectories. Recorded trajectories provide the measured joint signals used to evaluate and stack the regression matrix over N samples. The resulting regression problem is solved through least-squares estimation and refined via constrained optimization to enforce physical consistency, followed by offline validation against measured joint torques. 63

3.2 Mechanical structure of the six-degree-of-freedom industrial manipulator, highlighting joint couplings and load-compensation springs that affect joint-space dynamics. 66

3.3 Dynamic torque validation on a representative excitation trajectory. For each joint, the measured joint torque $\tau_i[k]$ (blue) is compared against the reconstructed torque $\hat{\tau}_i[k]$ (red) computed from the identified model. The horizontal axis reports the sample index k (uniform sampling), and torques are expressed in Nm. 70

4.1 Layered architecture for admittance-based lead-through programming under distance-modulated energy-based safety supervision. Safety supervision modulates interaction dynamics through parameter adaptation, while motion execution is delegated to the robot low-level controller. 75

4.2 Example of geometric abstraction for distance-based supervision. (a) Example of the simulation; (b) The operator is represented by a set of capsule primitives derived from the tracked skeleton, while the robot is approximated by link primitives consistent with its kinematic structure. The minimum distance d is evaluated as the smallest separation between any primitive pair $(h, r) \in \mathcal{H} \times \mathcal{R}$. 83

4.3 Control architecture for admittance-based lead-through programming under constraint-based safety supervision. The perception system provides geometric abstractions to the safety layer, which adapts admittance parameters without directly commanding robot motion. . 89

4.4 Representative time histories during lead-through programming, illustrating the causal chain from safety supervision to interaction behavior: adapted admittance parameters (inertia and damping), resulting scaling coefficient α, distance-modulated relative kinetic energy $k(d_{hr})\, C_{hr}$ for the critical human–robot pair, and minimum human–robot distance d (global diagnostic). 91

5.1 Illustration of the experimental setup for physical human-robot interaction. The system comprises a Universal Robot UR10e robotic manipulator and an OptiTrack motion capture system. The human operator holds the robot's end-effector, while an OptiTrack camera tracks a reflective marker cluster attached to the operator's chest to monitor their position during the collaborative task. 118

5.2 Representative time histories for the algorithmic PFL-based admittance adaptation. From top to bottom: evolution of the first diagonal admittance inertia and damping components (m_1, d_1), dissipated collision energy ΔK_e with respect to the PFL admissible bound $E_{\max}^{PFL}$, minimum human–robot distance d_{hr} (diagnostic), and energy tank level T. The vertical dotted line marks the onset of a fast approach motion leading to violation of the anticipatory feasibility condition ($h > 0$) and the switch to the Safe correction mode (5.4.3). 120

5.3 Representative time histories for the optimization-based PFL-driven admittance adaptation (zoomed view). From top to bottom: evolution of the first diagonal admittance inertia and damping components (m_1, d_1), PFL feasibility residual h, human–robot distance d_{hr} and energy tank level T. The highlighted interval corresponds to a fast approach motion bringing the system close to the feasibility boundary, without triggering discrete corrective actions. 122

6.1 Overall ROS2 architecture for language-based robot programming, highlighting the separation between LLM plan generation, deterministic validation/execution, and primitive actuation. 131

6.2 Behavior Core and its interfaces, including submodules responsible for primitive dispatch, state buffering, and trigger evaluation. 132

6.3 The system receives a natural-language command to palletize the boxes on the table. As additional boxes appear during execution, the YOLO-World perception module updates the world state at the next perception–execution cycle and the Behavior Core extends the loop accordingly, resulting in all four boxes being successfully palletized. . 138

6.4 The system receives a natural-language command to sort the boxes on the table into two corresponding areas. 139

List of Tables

3.1 Static validation results of the industrial robot, reported as per-joint RMSE values (Nm), and per-joint SDE (Standard Deviation of Error), computed over quasi-static configurations. 69

4.1 Main parameters used in experimental validation. 89

4.2 Qualitative interaction behavior observed during experimental validation. 92

5.1 Experimental setup and control parameters used for the comparative evaluation of algorithmic and optimization-based admittance adaptation strategies (from nini2025_algo). 117

7.1 Empirical outcomes and validation evidence across programming modalities. 146

7.2 Comparative matrix across modalities using the shared axes defined in Section 7.2.1. 147

Summary

Summary of Chapter 1: Introduction

Industrial robotics is undergoing a transition from paradigms of isolated automation toward collaborative scenarios in which humans and robots share workspaces, timing, and, in some cases, physical contact [1]. This transition challenges traditional assumptions underlying industrial robot design, where tasks are specified as fixed references and human presence is treated as an exception to be avoided. In Human–Robot Collaboration (HRC), the human becomes an intentional source of guidance and variability, requiring robot behavior to adapt continuously rather than merely reject disturbances. As a result, modeling, control, safety supervision, and programming can no longer be treated as loosely coupled layers, but must be conceived as parts of an integrated interaction architecture.

International safety standards such as ISO 10218 and ISO/TS 15066 formalize admissible collaborative modes and define quantitative limits on speed, force, and energy during interaction [2, 3]. While these standards elevate safety to a primary design constraint, they deliberately avoid prescribing specific control solutions. This shifts responsibility to system designers, who must reconcile safety compliance with interaction quality and usability. The core difficulty is therefore architectural: ensuring certifiable and predictable behavior without undermining the responsiveness and intuitiveness that motivate collaborative robotics in the first place.

Physical Human–Robot Interaction (pHRI) exemplifies this tension. Interaction modes such as hand-guiding and lead-through programming promise intuitive robot programming through direct physical guidance, but they also expose the limitations of classical position- or velocity-controlled architectures, which are not designed to interpret contact as a command channel [4]. Among compliant interaction strategies, admittance control has emerged as a practical and widely adopted solution, allowing robot motion to be generated in response to external forces through a virtual mass–damper behavior [5]. However, fixed-parameter admittance control embeds a static compromise between responsiveness and safety: parameters that ensure conservative behavior degrade transparency, while parameters that favor intuitive interaction may violate safety margins under adverse conditions [1]. This limitation motivates the introduction of time-varying interaction dynamics.

The need for adaptation becomes particularly evident under Power and Force Limiting (PFL), which regulates interaction by bounding the mechanical energy that can be transferred during contact [3]. Energy-based formulations reveal a tight coupling between robot velocity and apparent mass, the latter being directly influenced by control parameters such as the virtual inertia in admittance control [6]. Consequently, safety constraints cannot be treated as external supervisory conditions but must be integrated into the control design itself. Adaptive interaction strategies therefore require explicit consideration of stability and passivity, as time-varying parameters may inject energy into the human–robot system.

A further foundational aspect highlighted in the introduction is the role of dynamic modeling. Although interaction control is often formulated in task-space abstractions, industrial manipulators are characterized by significant inertial, gravitational, and frictional effects. Accurate dynamic parameter identification is essential not only for improving control performance, but also for ensuring the physical meaningfulness of energy-based safety metrics. Inaccurate models risk undermining both interaction quality and safety guarantees, especially in contact-rich collaborative scenarios.

Beyond physical interaction and safety, the introduction identifies usability as a persistent barrier to the adoption of collaborative robotics. Programming and configuring industrial robots remains an expert-driven process, limiting accessibility for non-specialist operators [7]. While higher-level programming interfaces, including natural language, offer promising avenues for reducing this burden, they raise fundamental challenges related to grounding abstract task descriptions into deterministic, verifiable, and safety-compliant robot execution [8, 9].

Existing approaches typically address dynamic modeling, interaction control, safety, and programming interfaces in isolation, or attempt to fuse them into monolithic architectures that blur execution authority and weaken interpretability. In contrast, this dissertation adopts a deliberately non-reductive perspective. Each contribution is grounded in a specific interaction modality and safety paradigm, with explicitly bounded assumptions and guarantees. In doing so, the thesis advances a single coherent argument not through unification, but through architectural separation: safe and easy robot programming is achieved by lowering the abstraction barrier at the human interface while preserving clear boundaries between intent specification, execution authority, and safety enforcement. This perspective underpins all chapters of the dissertation and frames human–robot collaboration as an architectural problem rather than a search for a universal control framework.

Summary of Chapter 2: Literature Review

The literature review reconstructs the conceptual landscape in which the problem of *safe and intuitive robot programming* has been addressed, adopting a problem-driven rather than discipline-driven perspective. Instead of cataloguing individual

techniques, the chapter analyzes how modeling, control, safety, and programming have historically evolved along partially independent trajectories, and how this fragmentation underlies many of the limitations observed in industrial Human–Robot Collaboration (HRC).

The review begins by examining the transition from classical industrial automation to collaborative robotics. Traditional industrial paradigms emphasize repeatability, precision, and strict human–robot separation, with programming performed offline or through low-level interfaces [4]. As production systems demand greater flexibility, programming effort has emerged as a central bottleneck, directly affecting deployment cost and accessibility [1, 7]. Collaborative robotics reframes this issue by allowing humans and robots to share workspaces and roles, but in doing so fundamentally alters the notion of programming: in scenarios such as hand-guiding or lead-through teaching, programming and execution become intertwined, and human intent is conveyed through physical interaction rather than symbolic code. The literature, however, often continues to treat programming interfaces, interaction control, and safety supervision as loosely coupled elements, obscuring their intrinsic interdependence in physical interaction-based programming.

Dynamic modeling and parameter identification are then reviewed as analytical foundations for interaction and safety reasoning. While rigid-body dynamics and parameter identification are well-established in robotics, their role in physical HRI is sometimes downplayed because many compliant control strategies do not rely directly on accurate models for stability [10]. The review clarifies that dynamic models remain essential not as universal control prerequisites, but as reference frames for interpreting interaction phenomena and computing safety-relevant quantities such as apparent inertia, power, and kinetic energy [11, 12]. Inaccurate models can distort these quantities, leading to overly conservative behavior or underestimated risk, particularly in energy-based safety formulations. From a programming standpoint, modeling supports the interpretability and repeatability of physically taught motions, even when compliance is achieved through model-agnostic control.

The chapter then analyzes admittance control as a dominant framework for physical HRI in industrial settings. Admittance control enables compliant behavior without requiring torque-level actuation, making it compatible with position- and velocity-controlled industrial robots [13]. Fixed-parameter admittance, however, embeds an unavoidable trade-off between responsiveness and stability, which becomes critical under variable human behavior and safety constraints. This limitation has motivated a large body of work on adaptive and variable admittance strategies [14, 15]. The review highlights a key unresolved issue: admittance parameters directly influence apparent mass and velocity response, and therefore the forces and energy exchanged during contact. As a result, adaptation strategies designed purely for interaction quality may inadvertently compromise safety if not explicitly coordinated with safety reasoning.

Safety paradigms are examined next, with particular focus on Speed and Separation Monitoring (SSM) and Power and Force Limiting (PFL). SSM enforces safety

through distance-based velocity regulation and is well suited to collision avoidance, but becomes increasingly conservative in close-proximity or intentional-contact scenarios [16]. PFL, by contrast, bounds the mechanical energy and forces exchanged during contact, enabling closer and more natural collaboration [3]. The review emphasizes that PFL introduces a structural coupling between control design and safety limits, since admissible velocities depend on apparent inertia, which is influenced by interaction control parameters [6]. This coupling challenges layered architectures that treat safety as an external supervisory function and motivates control strategies that explicitly regulate energy exchange.

Energy- and passivity-based frameworks are then discussed as tools for managing stability under time-varying interaction dynamics. Passivity provides robustness to unpredictable human behavior by constraining energy injection into the coupled system [17]. Energy tank mechanisms, in particular, enable stable adaptation of interaction parameters by mediating energy flow [18]. While these frameworks guarantee stability, the review clarifies that passivity alone does not ensure compliance with safety thresholds, highlighting the need for explicit integration between safety constraints and adaptation mechanisms.

Finally, the chapter reviews language-based robot programming as an emerging cognitive interaction modality. Natural language interfaces promise to lower the programming barrier by allowing users to specify tasks at a high level of abstraction [8, 19]. Recent advances in large language models have amplified this potential, but the literature consistently stresses their limitations: probabilistic outputs, lack of determinism, and absence of intrinsic safety awareness [19, 20]. Consequently, language-based programming is best understood not as a replacement for physical interaction or control, but as a complementary modality that requires structured execution, verification, and constraint enforcement to ensure safe deployment.

The synthesis of the literature identifies several persistent research gaps: the lack of a coherent role for dynamic modeling in interaction-oriented systems; the decoupling of adaptive admittance control from formal safety constraints; the insufficient integration between energy-based safety paradigms and control design; and the absence of unified architectures that ground cognitive programming interfaces in physically safe execution frameworks. These gaps motivate the thesis approach, which treats modeling, control, safety, and interaction as tightly coupled components of a single framework for safe and intuitive robot programming.

Summary of Chapter 3: Dynamic Modeling and Parameter Identification

Chapter 3 establishes the analytical foundation on which the subsequent safety-oriented interaction studies are interpreted, by developing and validating a dynamic model of the industrial manipulator used throughout the thesis. The chapter is

motivated by a tension that is often under-emphasized in interaction-oriented robotics: while physical programming and compliant control deliberately abstract the robot as an interaction port with shaped mass–damper behavior, the *physical consequences* of that abstraction remain mediated by the robot's true inertial, gravitational, frictional, and transmission properties. In heavy-duty industrial manipulators (characterized by substantial link masses, high payload capacity, reducer losses, and limited access to low-level torque loops) these physical effects are not incidental; they condition torque generation, energy exchange, and the interpretation of safety-relevant quantities during interaction.

A core conceptual move of the chapter is therefore to delimit the role of modeling in the dissertation architecture. Section 3.1 explicitly rejects a common misreading: the identified model is *not* a functional prerequisite for the admittance-based interaction controllers developed later. Those controllers are designed to remain robust and implementable under industrial constraints by relying on interaction-level abstractions and measurable signals, with stability treated through energy-consistent reasoning rather than through exact model cancellation. The identified dynamic model serves instead two sharply defined purposes: (i) it can support low-level torque feedforward compensation where the industrial control stack permits such a channel, reducing the burden on decentralized feedback loops without rearchitecting the controller; and (ii) it provides an offline, physically interpretable reference for computing and discussing safety- and interaction-related quantities (e.g., joint-space kinetic energy and configuration-dependent apparent inertia) without collapsing those analyses into purely qualitative claims.

The chapter then formulates the manipulator dynamics using the standard rigid-body joint-space model $\boldsymbol{\tau} = \mathbf{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_f(\dot{\boldsymbol{q}})$, with friction represented via viscous and Coulomb components. For identification, the dynamics are expressed in a linear regression form $\boldsymbol{\tau} = \mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{\pi}$, constructed via a Newton–Euler formulation to preserve computational efficiency and to accommodate industrial manipulator features [4]. The chapter emphasizes the methodological significance of linear parameterization: it enables principled estimation, identifiability analysis, and the construction of a *minimum* parameter set in the presence of structural dependencies that make the full parameter vector rank-deficient.

Building on this formulation, the identification procedure is presented as a structured two-stage workflow (static then dynamic) aligned with established practice and closely grounded in [21]. In the static phase, quasi-static configurations isolate gravity-dominated torques to estimate mass and center-of-mass related parameters with reduced coupling. In the dynamic phase, excitation trajectories spanning different velocity and acceleration regimes target inertial, motor, and friction-related parameters. The stacked regression is reduced to a minimal identifiable form and solved via least squares, followed by a refinement step through constrained optimization to enforce physical consistency (e.g., positivity of masses and inertia tensor validity). This sequencing is not merely procedural: it operationalizes the chapter's broader goal of producing a model that is *interpretable* and *usable as an analytical reference*, rather than a numerically convenient but physically questionable fit.

A distinctive contribution of the chapter lies in its treatment of the experimental platform as an *industrial mechanism*, not an ideal serial chain. The experimental setup is described with attention to features that materially affect torque balance and identifiability: load-compensation springs whose elongation depends on joint motion, mechanical couplings between joint pairs, reducer efficiencies and no-load losses that differ across joint groups, and base misalignment effects that can introduce gravitational components on joints that would otherwise appear unloaded. By explicitly acknowledging and incorporating these elements into the identification framework, the chapter guards against a common failure mode in industrial identification work: attributing systematic transmission or elastic effects to inertial parameters, thereby producing a model that may fit data locally but misrepresents the robot's physical structure.

Validation is performed by comparing measured joint torques against torques reconstructed from the identified model on trajectories not used for estimation, reporting both quantitative error metrics (e.g., per-joint RMSE) and time-domain comparisons that reveal regime-dependent residual structure. The chapter interprets residual errors as consistent with the unavoidable simplifications of rigid-body and friction modeling in the presence of couplings, elastic compensation, and reducer losses. Crucially, this interpretation is used to reinforce the chapter's architectural stance: the goal is not perfect torque prediction under all operating conditions, but a physically grounded approximation that captures dominant contributions and supports meaningful reasoning about safety margins and interaction energetics.

The chapter closes by translating modeling outcomes into thesis-level implications. The validated model enables computation of energy- and inertia-related quantities used later to interpret experiments and to discuss PFL-oriented constraints, while the *enforcement* of safety in subsequent chapters remains deliberately model-light to preserve robustness under modeling uncertainty and industrial variability. In this way, Chapter 3 performs a precise intellectual job within the dissertation: it provides the physical reference frame that makes later claims about "energy," "apparent mass," and "safety margins" analytically accountable, without turning the entire interaction-control architecture into a model-dependent edifice.

Summary of Chapter 4: Distance-Aware Lead-Through Programming on Industrial Robots

Chapter 4 addresses the problem of enabling *safe and intuitive lead-through programming* on a standard industrial manipulator operating outside the assumptions of collaborative robotics. The chapter is explicitly situated in a non-collaborative industrial context, where sustained physical contact cannot be assumed to be intrinsically safe and where interaction control must coexist with proprietary low-level controllers and safety-certified subsystems. Within this setting, the central challenge is not the feasibility of physical guidance itself, but the design of an interaction architecture

that preserves usability while discouraging unsafe proximity through continuous, non-disruptive safety supervision.

Conceptually, the chapter builds directly on the modeling foundations established in Chapter 3, while preserving a strict architectural separation between modeling, interaction control, and safety supervision. The identified dynamic model is deliberately not embedded in the admittance control law. Instead, it is employed in a narrowly defined and defensible role: supporting the computation and interpretation of physically meaningful safety-related quantities (such as equivalent mass and relative kinetic energy) and, optionally, improving motion tracking quality through feedforward compensation within the robot-side control stack. This choice reflects a core thesis-level stance: interaction control for industrial lead-through programming should remain robust to modeling uncertainty and compatible with closed industrial control architectures, while still benefiting from physically grounded analysis.

The chapter reframes lead-through programming as an *interaction-driven trajectory generation problem*. Human-applied wrenches are interpreted in real time as motion intent through a Cartesian admittance controller, which generates smooth motion references tracked by the robot's internal controller. Admittance control is adopted not only for its suitability to industrial position-controlled robots, but also for the conceptual separation it enables between interaction interpretation and motion execution. This separation is crucial in industrial environments, where direct torque control is unavailable and safety-certified control layers cannot be modified.

The core contribution of the chapter lies in the integration of *distance-aware safety supervision* into the interaction dynamics themselves. Rather than enforcing safety through discrete actions such as velocity clamping or emergency stops, the proposed architecture embeds supervision into the interaction layer by continuously adapting the admittance parameters. As unintended human–robot proximity increases, the apparent inertia and damping of the admittance model are progressively reshaped, reducing the robot's responsiveness to human-applied forces while preserving continuity of motion and the semantics of physical guidance. This design choice directly addresses a key usability concern: abrupt safety actions near proximity limits can degrade interaction quality and elicit compensatory operator behavior that is counterproductive from a safety standpoint.

Safety supervision is grounded in a single, physically interpretable constraint based on relative kinetic energy, whose influence is modulated by geometric distance. Distance is not treated as an independent or competing constraint, but as a contextual variable that regulates the severity of motion-related risk as proximity decreases. The kinetic-energy metric is evaluated along the closest-point direction between the robot and non-interacting parts of the human body, using a selective geometric abstraction that explicitly excludes intentional contact at the hand-guiding interface. This distinction between intended and unintended contact is essential to reconcile distance-aware supervision with the realities of lead-through programming.

To ensure stability under time-varying interaction dynamics, the chapter in-

troduces an energy-tank mechanism that preserves passivity of the admittance mapping despite online parameter adaptation. The energy tank acts as a passivity supervisor that mediates between safety-driven parameter requests and the actual applied parameters, ensuring energetic consistency without introducing collision or injury guarantees. Importantly, the energy-tank framework is positioned as a stability-preserving mechanism rather than as a safety standard in itself, reinforcing the chapter's careful separation between interaction stability and formal safety compliance.

The perception system is treated as an independent provider of geometric information rather than as a decision-making component. Vision-based human state estimation supplies conservative geometric abstractions of the human and robot, pairwise distances, and closest-point directions through a clearly defined interface. All safety logic, constraint evaluation, and parameter adaptation are implemented deterministically within the supervision layer, preserving inspectability and modularity.

Experimental validation on a real industrial platform demonstrates that the proposed framework yields smooth, bounded adaptation of admittance parameters in response to proximity and motion intensity. The results show that distance modulation activates energy-based supervision progressively, avoiding abrupt behavior changes while effectively attenuating robot responsiveness in critical conditions. From the operator's perspective, this manifests as a gradual increase in resistance rather than as disruptive motion inhibition, confirming the viability of the approach as a programming modality.

The chapter concludes by explicitly delineating the scope and limitations of the proposed paradigm. The framework does not claim ISO-certified Speed and Separation Monitoring compliance, nor does it implement Power and Force Limiting or injury-based safety guarantees. Its safety claim is architectural and conditional: given conservative geometric modeling and reliable perception, interaction dynamics are shaped to discourage unsafe proximity without relying on contact-based safety assumptions. This positioning is deliberate and sets the stage for the next chapter, which moves beyond distance-aware supervision toward energy-based safety paradigms suitable for collaborative and contact-rich human–robot interaction.

Summary of Chapter 5: Energy-Based Variable Admittance Control for Collaborative Robotics

This chapter addresses safety in physical human–robot interaction under a *Power and Force Limiting (PFL)* paradigm, marking a deliberate conceptual shift from the distance-modulated supervision adopted in Chapter 4. While the previous chapter treated safety as an external constraint acting on motion in non-collaborative industrial settings, here safety is reformulated as an *intrinsic energetic feasibility*

condition that must be satisfied during intentional physical contact, in accordance with ISO/TS 15066.

The chapter establishes an energy-based interpretation of PFL by expressing safety as a bound on the transferred kinetic energy during potential human–robot contact. This formulation reveals that feasibility depends jointly on the relative closing velocity and on the robot's apparent mass along the interaction direction. Under Cartesian admittance control, both quantities are directly shaped by the admittance parameters, making inertia and damping safety-critical design variables rather than mere interaction-tuning parameters.

Building on this observation, the chapter explicitly exposes the coupling between admittance inertia, apparent mass, damping, and velocity evolution. This coupling motivates the use of adaptive admittance strategies that regulate safety by reshaping interaction dynamics, rather than by imposing external velocity limits or stopping actions. Passivity preservation under time-varying admittance is guaranteed through the energy-tank mechanism introduced in Chapter 4, which is retained unchanged and acts as an architectural supervisor separating energetic consistency from safety enforcement.

Two distinct PFL-oriented admittance adaptation strategies are developed and analyzed. The first is an *algorithmic* approach based on a two-mode logic: a nominal mode that smoothly co-adapts inertia and damping while preserving a fixed ratio to maintain interaction consistency, and a safe-correction mode that injects damping in a targeted and anticipatory manner when the PFL feasibility residual becomes positive. This strategy emphasizes determinism, interpretability, and rapid recovery at the cost of short-lived interaction discontinuities.

The second strategy replaces rule-based switching with an *optimization-based* formulation. Here, inertia and damping rates are treated as coupled decision variables selected at each control cycle by solving a constrained optimization problem. Safety feasibility, parameter bounds, rate limits, and passivity-aware inertia variation are enforced as constraints, while interaction comfort and smoothness are encoded in the objective function. A soft-constraint variant allows graceful degradation when strict feasibility cannot be achieved instantaneously, avoiding discrete corrective actions.

Experimental validation on a collaborative manipulator demonstrates that both strategies successfully regulate the PFL feasibility residual while preserving passivity. The algorithmic approach yields fast and predictable feasibility restoration but introduces perceptible impedance discontinuities during corrective phases. The optimization-based approach produces smoother and more homogeneous interaction behavior by continuously trading off apparent mass reduction and dissipation, at the expense of increased computational complexity and tuning effort.

Overall, the chapter shows that PFL-compliant safety can be embedded directly into admittance-controlled interaction through energy-based reasoning, without relying on external motion scaling or stopping mechanisms. The comparative analysis

highlights a fundamental design trade-off between reactivity and continuity, positioning the two strategies as complementary solutions for different collaborative robotics scenarios.

Summary of Chapter 6: Artificial Intelligence for Intuitive Robot Programming

This chapter addresses the *cognitive bottleneck* of robot programming by introducing large language models (LLMs) as high-level programming interfaces, explicitly decoupled from control, execution, and safety. While previous chapters focused on physical interaction and safety-aware control, the contribution here targets a complementary dimension: reducing the conceptual and cognitive effort required to specify robot tasks, especially in flexible and collaborative settings.

The chapter starts from the observation that the primary obstacle to widespread robot adoption is not control capability but intent misalignment: humans naturally describe tasks at a semantic level, whereas robots require precise, low-level specifications. Traditional programming paradigms impose a steep learning curve and remain inaccessible to non-experts, even when physical interaction or demonstration is available. In this context, LLMs are positioned not as decision-makers or controllers, but as *semantic translators* that convert natural language into structured task descriptions.

A central design principle of the chapter is the strict scoping of the LLM's authority. Language models are confined to producing symbolic task plans expressed in a predefined JSON schema. They have no access to robot state, sensors, control parameters, or safety logic, and they cannot synthesize new skills or executable code. All execution authority is retained by a deterministic *Behavior Core*, implemented within a modular ROS2 architecture. This separation preserves determinism, traceability, and verifiability while mitigating the risks associated with probabilistic language generation and hallucination.

The chapter formalizes this architecture by introducing a clear module decomposition: perception, language interface, Behavior Core, and robot interface. Open-vocabulary perception enables flexible object references, while execution remains constrained to a closed set of primitive skills. The Behavior Core validates language-generated plans, manages symbolic state, evaluates triggers, and dispatches primitives in discrete perception–execution cycles. As a result, every executed motion can be traced to a specific plan element and deterministic decision, independent of the internal workings of the language model.

Semantic grounding is achieved through a structured task-plan schema that supports initialization actions, cyclic behaviors, trigger conditions, and bounded symbolic data flow. This representation enables expressive behaviors such as palletization and

sorting while forbidding arbitrary computation or direct control synthesis. Validation contracts ensure primitive closure, reference integrity, and restricted trigger logic, containing failures at the interface level rather than at execution.

Experimental demonstrations show that users can specify tasks such as palletizing or sorting objects using unconstrained natural language, with the system translating these instructions into executable behaviors as long as they lie within the available primitive set. The experiments are intentionally qualitative, emphasizing cognitive accessibility, execution transparency, and graceful failure over performance metrics or safety guarantees.

The chapter concludes with a critical discussion of limitations and explicit non-claims. The proposed framework does not address physical safety, does not provide autonomy or learning capabilities, and does not claim robustness to adversarial inputs or suitability for safety-critical environments. Its contribution lies in showing that LLMs can be integrated into robotic systems in a conservative and architecturally disciplined manner, serving as programming interfaces that lower cognitive barriers while preserving deterministic execution and clear responsibility boundaries.

Overall, this chapter positions language-based programming as an additional interaction modality (orthogonal to physical interaction and safety-aware control) aimed at improving usability and expressiveness without overextending the role of AI in robotic systems.

Summary of Chapter 7: Results, Discussion, and Synthesis

Chapter 7 provides the integrative synthesis that the earlier chapters intentionally avoided. Rather than presenting the dissertation as a linear progression from classical robotics to "AI robotics," the chapter argues that the thesis is unified by a consistent discipline of *claims and guarantees*: each programming modality makes robot programming easier, but each does so by placing a different kind of boundary on what is allowed and what can be asserted as safe.

Core unifying claim. The dissertation is not unified by a single reusable algorithm, but by a repeated architectural stance: keep the human-facing interface expressive (physical guidance or language), while keeping the execution authority bounded and auditable. Across modalities, the chapter defends the idea that "safe and easy programming" is not one predicate; it is a family of modality-specific guarantees that must be stated without rhetorical inflation.

Summary of outcomes without false aggregation. The chapter first consolidates experimental outcomes, emphasizing that each modality is evaluated by metrics intrinsic to its own safety logic. Dynamic identification is validated through torque reconstruction; industrial lead-through through distance-informed supervision and constraint activations; collaborative PFL through satisfaction of an energy-residual inequality; and LLM-mediated programming through plan admissibility and deterministic executability. The chapter explicitly rejects the temptation to compress these into a single scalar score, and instead summarizes outcomes as *validated claims* tied to observable evidence, captured in a modality-agnostic table of "observed property vs. validation artifact."

Comparative analysis along shared axes. To avoid "apples vs. oranges" objections, the chapter compares modalities along four shared axes: (i) where intent enters the system (physical guidance vs. language), (ii) what is bounded (distance-modulated behavior vs. transferred energy vs. action-space closure), (iii) what the system assumes (sensing and projections vs. PFL tables and residuals vs. schemas and trigger languages), and (iv) dominant failure modes (sensing uncertainty and unmodeled effects vs. real-time feasibility and residual conservatism vs. semantic miscompilation and perception mismatch). A comparative matrix makes explicit that each modality has a different core guarantee and, equally important, different *explicit non-claims* (e.g., "SSM-inspired but not ISO-certified," "PFL reasoning under modeling assumptions," "deterministic execution but not physical safety certification").

What "safe and easy programming" means here. A key conceptual contribution of the chapter is to clarify that "safe" is polysemous across the thesis. In physical-interaction modalities, safety is tied to continuous-time physically grounded quantities (separation margins, transferred energy). In the language modality, "safe-ish" refers to *command admissibility and execution structure*: the LLM is prevented from expanding the robot's action repertoire, and execution is mediated by a deterministic executive. The chapter argues that the thesis remains coherent only if this multiplicity is made explicit rather than collapsed into a single safety narrative.

Contributions as a design methodology. The chapter then reframes the dissertation's contributions as methodological rather than as isolated algorithmic novelties. Three contributions are stated.

First, the identification work contributes a structured static–dynamic separation that improves interpretability and validation, positioning the model as an *epistemic instrument* for explanation and diagnostics rather than as a hidden control dependency.

Second, the variable-admittance work contributes an energy-consistent view of adaptation: time-varying admittance is treated as a safety-relevant degree of freedom

because inertia shapes apparent mass and therefore impacts admissible interaction energy; algorithmic and optimization-based adaptations operationalize this coupling while remaining constrained by passivity-preserving supervision.

Third, the LLM programming work contributes a bounded-autonomy architecture in which the LLM is demoted to a constrained plan generator and a deterministic Behavior Core enforces checkable correctness properties (schema validity, primitive closure, typed parameters, reference integrity, restricted triggers). The contribution is not "LLMs control robots," but "LLMs can enable intuitive programming *if* execution boundaries are deterministic and enforceable."

Limitations and lessons learned. Finally, the chapter states limitations as boundary conditions on generalization. Distance-based supervision is only as credible as sensing fidelity and geometry abstractions, risking either unsafe optimism or unusable conservatism. PFL reasoning is powerful but assumption-dependent (equivalent human mass, stiffness, contact parameters, projection direction), so it should be treated as structured conservatism rather than universal truth. LLM "safe-ish" programming constrains *what* can be commanded, but does not certify *how safely* actions unfold, and schema-valid plans can still be semantically wrong under perception uncertainty or underspecified intent. The overarching lesson is that credible progress comes from strengthening explicit boundaries and failure handling, not from collapsing modalities into a single, overclaimed autonomy narrative.

Summary of Chapter 8: Conclusions and Future Outlook

Chapter 8 concludes the dissertation by consolidating its intellectual stance and situating its contributions within both industrial practice and future research trajectories. Rather than claiming a unified control framework or a single notion of safety, the chapter reiterates and defends the dissertation's central design doctrine: *robot programming can be made substantially easier for humans only if safety guarantees are made explicit, bounded, and modality-specific.* The coherence of the thesis lies not in algorithmic uniformity, but in architectural discipline and epistemic restraint.

A non-reductive synthesis of programming modalities. The chapter emphasizes that the three programming modalities developed throughout the dissertation—industrial lead-through programming, collaborative physical interaction under PFL, and language-based programming via LLMs—are deliberately non-interchangeable. Each modality addresses the same overarching problem (ease of programming under safety constraints), but does so by bounding a different quantity that is appropriate to its interaction channel. This pluralism is not a weakness but a necessary response to the heterogeneous nature of human–robot interaction contexts.

Ease through abstraction, safety through explicit bounds. Across all modalities, ease of programming is achieved by lowering the abstraction level at which humans specify intent: physical guidance replaces explicit trajectory specification; compliant interaction replaces rigid motion commands; natural language replaces formal robot programming constructs. Safety, however, is never delegated to the interface itself. Instead, it is preserved by bounding a clearly defined quantity (distance-modulated energetic behavior in industrial lead-through, transferred energy in collaborative PFL interaction, and structural admissibility in language-based programming) and by enforcing these bounds through architectures that separate intent specification from execution authority.

Industrial lead-through: conservative supervision, not collaborative safety. In the industrial case, the chapter reiterates that intuitive hand-guiding of non-collaborative robots becomes viable when variable admittance control is coupled with distance-aware supervision and passivity-inspired energy consistency. Crucially, the safety claim is framed negatively as well as positively: the system is SSM-inspired but not SSM-certified, and it does not rely on force or power limitation. The dynamic model identified earlier in the thesis is reaffirmed as an analytical and interpretive tool, not as a hidden control dependency.

Collaborative interaction: safety indexed to energy, not modeling precision. For collaborative robots operating under PFL assumptions, the conclusion stresses that safety is formulated and validated in energetic terms. Variable admittance becomes a safety-relevant degree of freedom because it shapes apparent mass and admissible velocities under fixed energy limits. Energy tanks and residual-based constraints provide a principled way to regulate adaptation without equating safety with exact dynamic inversion. The chapter highlights that this framing deliberately prioritizes bounded energy exchange over model accuracy.

Language-based programming: structural safety without physical claims. The language-based modality is explicitly positioned outside the physical safety paradigms of ISO standards. Its contribution lies in demonstrating that natural-language programming can coexist with deterministic, auditable execution when LLMs are confined to semantic translation roles. The system is described as "safe-ish" only in the structural sense: bounded action spaces, schema validation, admissible trigger logic, and a deterministic behavior core prevent the LLM from issuing arbitrary or unsafe commands. The chapter explicitly rejects attributing physical safety properties to language models themselves.

Industrial implications: architecture over interfaces. From an industrial standpoint, the chapter argues that human-centered automation is fundamentally an architectural challenge. The practical value of the dissertation lies in showing

that intuitive programming interfaces can be integrated into existing robotic systems without undermining safety, provided that responsibility boundaries are explicit and enforceable. Transparency about non-claims—what the system does not guarantee—is presented as a prerequisite for trust, certification, and responsible deployment.

Future research framed by restraint. Finally, the chapter outlines future research directions that extend the presented work without collapsing its carefully maintained separations. Predictive safety reasoning, richer runtime monitoring for language-based execution, and the use of digital twins as validation instruments are identified as promising avenues. Importantly, these directions are framed as extensions of the existing paradigms, not as attempts to merge them into a single, ambiguous notion of safety.

Closing stance. The concluding message of Chapter 8 is methodological rather than technological. Safe and easy robot programming, the chapter argues, is achieved not by increasing autonomy or intelligence indiscriminately, but by bounding interaction channels, separating concerns, and making safety claims commensurate with what the system can actually guarantee. In this sense, the dissertation offers both concrete technical contributions and a disciplined design philosophy that can guide future research and industrial practice.

Chapter 1

Introduction

Industrial robotics is undergoing a profound transformation, driven by the increasing demand for flexibility, adaptability, and closer integration between human operators and automated systems [1]. Traditional automation paradigms, which rely on rigid task execution and strict spatial separation between humans and robots, are progressively giving way to collaborative models in which robots are expected to operate safely and effectively in shared workspaces. This shift is not merely technological, but conceptual: it requires rethinking how robots are modeled, controlled, programmed, and certified for interaction with humans.

The emergence of Human–Robot Collaboration (HRC) as a recognized industrial paradigm has been accompanied by the formalization of international safety standards, most notably ISO 10218 and ISO/TS 15066, which define admissible modes of collaboration and quantify acceptable limits for speed, force, and energy during interaction [2, 3]. These standards establish safety as a first-class design constraint, but they do not prescribe how control architectures should be structured to satisfy such constraints while preserving performance and usability. As a result, the responsibility for reconciling safety, efficiency, and interaction quality is largely delegated to control and system-level design.

Within this context, physical Human–Robot Interaction (pHRI) represents both a critical opportunity and a significant challenge. Tasks such as hand-guiding, lead-through programming, and cooperative manipulation promise to make industrial robots more accessible and intuitive, but they also expose the limitations of conventional control strategies when confronted with direct human contact. Survey works in robotics and HRC emphasize that achieving safe and effective interaction requires a tight integration of dynamic modelling, compliant control, perception, and real-time adaptation [1, 4, 7].

This thesis is positioned within this evolving landscape and addresses the problem of enabling safe, adaptive, and intuitive human–robot collaboration in industrial robotics. Rather than treating modeling, control, safety, and interaction as independent layers, the thesis adopts a progressive and integrated perspective, in

which each stage builds upon the previous one. The chapter introduces the research context, articulates the limitations of existing approaches, and defines the objectives and methodological trajectory that guide the dissertation.

The remainder of this chapter is structured as follows. Section 1.1 introduces the broader research context and motivation, situating the work within current developments in industrial robotics and HRC. Section 1.2 formulates the problem statement by identifying the conceptual and technical gaps that motivate the proposed research. Section 1.3 presents the research objectives and guiding questions, while Section 1.4 outlines the methodological approach adopted throughout the thesis. Finally, Section 1.5 provides an overview of the thesis structure and the logical progression of its contributions.

1.1 Research Context and Motivation

Building on the transition from rigid industrial automation to collaborative paradigms outlined at the beginning of this chapter, this section focuses on the control and interaction implications of Human–Robot Collaboration. In particular, it highlights how these paradigms challenge conventional modelling and control assumptions in industrial robotics.

Human–Robot Collaboration (HRC) has emerged as a central paradigm aimed at overcoming these limitations by enabling humans and robots to work together within shared workspaces. HRC seeks to combine complementary capabilities: human adaptability, perception, and decision-making on the one hand, and robotic strength, repeatability, and endurance on the other. Comprehensive surveys highlight that achieving effective collaboration requires not only advances in mechanical design and perception, but also fundamental changes in control architectures and interaction strategies [1, 7].

The transition from isolated automation to collaborative operation is accompanied by the formalization of international safety standards. These standards introduce admissible collaborative modes, including Speed and Separation Monitoring (SSM) and Power and Force Limiting (PFL), and specify quantitative limits on speed, force, pressure, and energy during interaction [2, 3]. While these standards establish essential safety requirements, they deliberately avoid prescribing specific control solutions. As a consequence, ensuring both safety compliance and acceptable task performance becomes a system-level responsibility that must be addressed through appropriate modeling, control, and supervision strategies.

Physical Human–Robot Interaction (pHRI) represents one of the most challenging and impactful manifestations of HRC. In pHRI scenarios, such as hand-guiding, lead-through programming, and cooperative manipulation, humans and robots exchange forces directly, and the robot motion is shaped by physical interaction rather than by predefined references. These interaction modes promise to significantly reduce

the programming burden and increase accessibility for non-expert users. However, they also expose the intrinsic limitations of classical position- or velocity-controlled robots when confronted with direct human contact, uncertainty, and variability in operator behavior [4].

Admittance control has become a widely adopted framework for pHRI, as it allows the robot to generate motion in response to externally applied forces by emulating a virtual mass–damper system [5]. By appropriately tuning inertia and damping parameters, admittance control enables compliant and intuitive interaction while maintaining stability. Nevertheless, extensive literature points out that fixed-parameter admittance control is insufficient in realistic collaborative scenarios, where interaction conditions, human intent, and safety margins vary continuously over time [1]. In particular, static parameter choices force a trade-off between responsiveness and safety that cannot be resolved satisfactorily across all operating conditions.

A critical aspect often underlined in the literature is that the effectiveness of compliant interaction strategies is influenced by the accuracy of the underlying robot dynamic model. Precise knowledge of inertial properties, friction, and gravity effects plays a key role in certain control architectures and in the estimation of physically meaningful quantities such as apparent mass and kinetic energy, which play a central role in safety assessment. Inaccurate dynamic modeling can therefore undermine both interaction quality and safety guarantees, especially in energy-based formulations of collaborative control. This dependency motivates treating dynamic modeling as a foundational element rather than a secondary implementation detail.

As collaborative scenarios become more contact-rich, distance-based safety supervision alone becomes increasingly conservative or insufficient. In this context, Power and Force Limiting (PFL) emerges as a particularly relevant safety paradigm, as it regulates interaction by bounding the mechanical energy that can be transferred during contact [3]. Energy-based safety formulations naturally couple robot velocity and apparent mass, revealing a direct link between control parameters and admissible interaction dynamics [6]. This coupling motivates control strategies that explicitly adapt interaction dynamics in response to safety constraints, rather than enforcing fixed limits or binary mode switches.

Beyond physical interaction and safety, a further barrier to the widespread adoption of collaborative robotics lies in the cognitive accessibility of robot programming. Even when compliant and safety-aware control strategies are available, programming and configuring industrial robotic systems remains a complex task that requires expert knowledge and detailed understanding of robot behavior. Surveys in robotics and intelligent systems highlight that this usability gap represents a significant obstacle to the deployment of collaborative robots, particularly in small and medium-sized enterprises [7]. At the same time, attempts to introduce higher-level and more intuitive programming interfaces raise fundamental challenges related to the grounding of abstract task descriptions in physical robot execution. Ensuring that high-level commands remain interpretable, deterministic, and compatible with safety constraints imposed by physical interaction and industrial standards remains

an open problem in human–robot collaboration [8, 9].

Taken together, these considerations motivate a research direction centered on safe and intuitive robot programming, in which physical interaction and cognitive interfaces represent complementary programming modalities with different safety guarantees. This thesis is situated within this perspective and aims to contribute to the development of industrial robots that are not only safe and efficient, but also adaptive, transparent, and accessible to human operators.

1.2 Problem Statement

Despite significant advances in collaborative robotics, enabling safe, efficient, and intuitive human–robot interaction in industrial environments remains an open and multifaceted problem. Existing approaches address individual aspects of collaboration (such as safety certification, compliant control, or intuitive programming) but often fail to provide an integrated solution capable of reconciling these requirements simultaneously.

A first fundamental limitation concerns the rigidity of classical industrial control architectures. Traditional robot controllers are designed around position or velocity tracking with limited adaptability, implicitly assuming a predictable environment and minimal human interference. While such architectures can be augmented with external safety systems, they remain ill-suited for scenarios involving direct physical interaction, where robot behavior must continuously adapt to human intent, motion, and proximity [4]. As a result, safety is frequently enforced through conservative overrides (such as emergency stops or aggressive speed reductions) that interrupt task execution and degrade interaction quality.

Despite their widespread adoption in physical Human–Robot Interaction, admittance-based control strategies in industrial settings remain predominantly static or heuristically tuned. This static configuration forces an inherent trade-off: parameters chosen to ensure safety and stability tend to reduce responsiveness and transparency, whereas parameters optimized for ease of interaction may violate safety margins under adverse conditions [1]. The literature lacks a principled framework for resolving this trade-off dynamically and continuously during interaction.

A second critical issue arises from the relationship between control design and safety assessment. In practice, safety supervision is often implemented through distance-based monitoring or binary mode switching, particularly in Speed and Separation Monitoring (SSM). While effective for collision avoidance, these approaches become increasingly conservative as human–robot distance decreases and are poorly suited for intentional contact or hand-guiding tasks.

Energy-based safety paradigms, such as Power and Force Limiting (PFL), offer a more natural framework for close collaboration by bounding the mechanical

energy exchanged during interaction. However, PFL formulations reveal a tight coupling between robot velocity and apparent mass, which is directly influenced by control parameters such as the desired inertia in admittance control. This coupling exposes a structural limitation of existing methods: safety constraints and interaction dynamics are often treated as independent layers, even though they are physically interdependent [3]. Without explicit coordination, safety enforcement may lead to overly restrictive behavior or unstable interaction.

The problem is further compounded by uncertainties in robot dynamic modeling. Accurate estimation of inertial parameters, friction, and gravity effects is essential for reliable control and for computing physically meaningful safety metrics. Yet, in many industrial applications, dynamic models are either unavailable or insufficiently accurate, leading to discrepancies between predicted and actual interaction behavior. These discrepancies can undermine the effectiveness of certain compliant control strategies and the reliability of energy-based safety reasoning, particularly in scenarios where safety margins are tight.

Beyond physical interaction and safety, a further unresolved challenge concerns the usability of industrial robotic systems. Even when compliant and safety-aware control strategies are available, programming and configuring robots remains a complex and expert-driven process. This usability gap is widely recognized as a major barrier to the adoption of collaborative robotics, particularly in industrial contexts that require frequent reconfiguration or involve non-expert operators [7]. Efforts to address this gap through higher-level programming abstractions introduce additional challenges rather than resolving them outright. In particular, translating abstract task descriptions into executable robot behaviors requires mechanisms that guarantee determinism, verifiability, and compatibility with physical safety constraints. In such contexts, safety is often ensured through structured execution and constraint enforcement rather than through formal compliance with industrial safety standards.

Taken together, these limitations reveal a core unresolved problem: current industrial robotic systems lack a unified framework that integrates accurate dynamic modeling, adaptive and safety-aware interaction control, and intuitive human-centered programming. Existing solutions tend to address these aspects in isolation, resulting in fragmented architectures that either sacrifice performance for safety, usability for rigor, or adaptability for predictability. Addressing this problem requires a systematic approach that treats modeling, control, safety, and interaction not as independent modules, but as tightly coupled components of a coherent human–robot collaboration paradigm.

1.3 Research Objectives and Questions

The overarching objective of this thesis is to develop a coherent framework for safe, adaptive, and intuitive human–robot collaboration in industrial robotics. In response

to the limitations identified in Section 1.2, the thesis does not address modeling, control, safety, and interaction in isolation, but rather investigates how these elements can be systematically integrated into a unified architecture that supports physical human–robot interaction while preserving industrial reliability and safety compliance.

To achieve this overarching goal, the research is structured around a set of specific objectives, each addressing a distinct yet interconnected aspect of the problem.

Objective 1: Dynamic modeling and parameter identification to support low-level compensation, physical interpretation of interaction phenomena, and validation of safety-oriented metrics

The first objective is to obtain accurate dynamic models of an industrial six-degree-of-freedom manipulator through systematic parameter identification. This objective addresses the need for reliable estimation of inertial, frictional, and gravity-related effects. The thesis seeks to reduce the uncertainty that undermines compliant interaction and safety assessment in industrial settings.

Objective 2: Adaptive Admittance Control for Safe Physical Interaction

The second objective is to design and validate an adaptive admittance control strategy that enables safe and intuitive hand-guiding and lead-through programming. This objective responds to the limitations of fixed-parameter admittance control by introducing real-time adaptation of interaction dynamics in response to human-applied forces and safety-related metrics. The aim is to achieve a continuous compromise between responsiveness, stability, and safety, rather than relying on conservative static tuning or discrete mode switching.

Objective 3: Energy-Based Safety Integration in Collaborative Scenarios

The third objective is to extend adaptive admittance control to collaborative scenarios characterized by close proximity and intentional contact, where distance-based safety supervision is insufficient. This objective focuses on integrating energy-based safety constraints derived from Power and Force Limiting principles into the control architecture. By explicitly regulating interaction dynamics to bound kinetic energy and power transfer, the thesis aims to reconcile safety compliance with effective task execution in contact-rich human–robot collaboration.

Objective 4: Optimization-Based Adaptation under Safety Constraints

The fourth objective is to formalize the adaptation of interaction parameters as a constrained optimization problem. This objective addresses the limitations of purely heuristic or algorithmic adaptation strategies by providing a principled mechanism to balance performance, stability, and safety requirements. By embedding safety constraints directly into the control optimization process, the thesis seeks to ensure predictable and verifiable behavior under varying interaction conditions.

Objective 5: Intuitive Robot Programming through Cognitive Interfaces

The fifth objective is to investigate the integration of cognitive interaction mechanisms for intuitive robot programming. Specifically, this objective explores how high-level task descriptions expressed in natural language can be translated into executable robot behaviors through a structured and safety-aware interface. The goal is to reduce the programming burden for human operators while preserving determinism and consistency with the underlying control and safety architecture.

Building upon these objectives, the thesis is guided by the following research questions:

- **RQ1:** How accurately can the dynamic parameters of an industrial manipulator be identified in a way that is suitable for both interaction control and energy-based safety assessment?
- **RQ2:** How can admittance control parameters be adapted online to balance transparency, stability, and safety during physical human–robot interaction?
- **RQ3:** How can energy-based safety constraints derived from Power and Force Limiting principles be integrated into adaptive interaction control without excessively degrading task performance?
- **RQ4:** How can safety and interaction requirements be formulated as explicit constraints within an optimization-based control framework for collaborative robotics?
- **RQ5:** How can natural language interaction be grounded in low-level robot control in a manner that remains safe, deterministic, and compatible with industrial requirements?

Collectively, these objectives and research questions define a structured research trajectory that progresses from model-based foundations to adaptive and safety-aware

control, and ultimately to intuitive human-centered interaction. Each subsequent chapter of the thesis addresses one or more of these questions, ensuring that the dissertation forms a coherent and logically necessary whole rather than a collection of independent contributions.

1.4 Research Methodology and Approach

The research methodology adopted in this thesis follows a progressive and integrative approach, reflecting the progressive nature of human–robot collaboration in industrial settings. Rather than relying on a single methodological paradigm, the work combines model-based analysis, control-theoretic design, experimental validation, and system-level integration. This combination is necessary to address the intertwined challenges of physical interaction, safety assurance, and usability identified in the preceding sections.

At its foundation, the methodology adopts a model-based perspective. Accurate dynamic modeling is treated as a prerequisite for both control performance and safety reasoning. The identification of robot dynamic parameters is approached through systematic excitation, measurement, and estimation procedures grounded in classical robot dynamics and inverse dynamics formulations [4]. Model validation is performed by comparing predicted and measured joint torques under representative motion profiles, ensuring that the identified parameters are suitable for subsequent interaction control and energy-based analysis.

Building upon this foundation, the thesis employs control-theoretic methods to design interaction strategies for physical Human–Robot Interaction. Admittance control is selected as the primary interaction paradigm due to its suitability for force-guided motion and its widespread adoption in both academic and industrial contexts. The control design explicitly considers the trade-offs between transparency, stability, and robustness discussed in the HRC literature [1]. To overcome the limitations of fixed-parameter control, adaptive mechanisms are introduced, enabling real-time modification of interaction dynamics in response to measured forces and safety-related variables.

Safety is treated not as an external supervisory layer, but as an intrinsic component of the control architecture. In this respect, the methodology integrates safety constraints derived from international standards directly into the interaction control loop. Distance-based supervision and energy-based Power and Force Limiting principles are interpreted as quantitative constraints on admissible robot motion and interaction dynamics [3]. This perspective allows safety requirements to be expressed in physically meaningful terms and to be enforced continuously rather than through discrete mode switching.

To ensure stability under time-varying interaction dynamics, the methodology leverages passivity-based analysis and energy-based reasoning. Passivity theory

provides a rigorous framework for analyzing the stability of coupled human–robot systems, particularly in the presence of variable control parameters [4]. Energy storage mechanisms, such as virtual energy tanks, are employed to regulate energy exchange during interaction and to preserve passivity despite online adaptation of control parameters.

As interaction scenarios become more complex, the methodology transitions from purely algorithmic adaptation rules to optimization-based formulations. Interaction parameter adaptation and safety enforcement are cast as constrained optimization problems, where performance objectives are balanced against explicit safety and stability constraints. This formulation enables systematic handling of competing requirements and provides a clear mathematical structure for analyzing feasibility, robustness, and predictability of the resulting behavior.

Experimental validation constitutes a central component of the research methodology. All proposed models and control strategies are implemented and tested on a real industrial robotic platform, ensuring relevance beyond simulation. A modular software architecture based on ROS 2 [22] is adopted to integrate sensing, control, safety supervision, and higher-level reasoning. This choice supports real-time execution, reproducibility, and extensibility, and reflects current best practices in experimental robotics research [23].

Finally, to address the usability challenges of collaborative robotics, the methodology extends beyond physical interaction control to include cognitive interaction mechanisms. High-level task specification through natural language is investigated as a complementary layer built upon the validated control and safety framework. Rather than replacing classical control, language-based interfaces are treated as supervisory tools that generate structured task representations, which are then executed through deterministic and safety-aware control pipelines. This layered approach aligns with emerging perspectives on Vision–Language–Action systems, which emphasize the importance of grounding symbolic reasoning in physical execution [7].

In summary, the methodology adopted in this thesis is characterized by a deliberate progression from modeling to control, from safety-aware adaptation to optimization, and from physical interaction to cognitive interfaces. This progression ensures that each contribution is grounded in validated foundations and that higher-level interaction capabilities do not compromise the physical safety and reliability required in industrial robotics. Figure 1.1 provides a high-level overview of the conceptual progression underlying the thesis.

1.5 Thesis Outline

This thesis is structured to reflect a coherent progression from foundational modeling to adaptive and safety-aware interaction control, and ultimately to intuitive human-centered robot programming. Each chapter builds upon the previous ones,

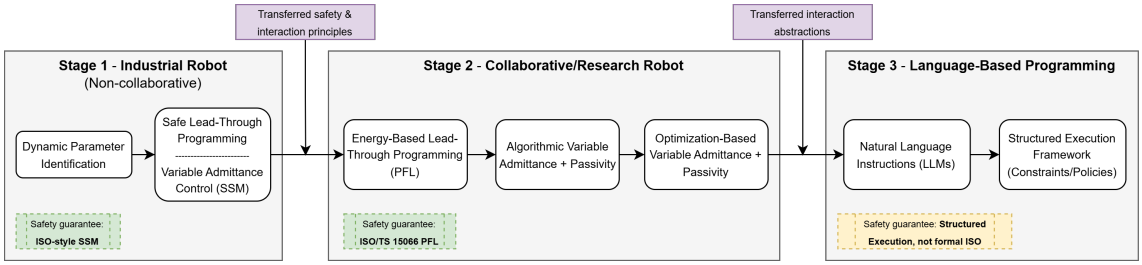


Figure 1.1: Conceptual roadmap of the thesis, illustrating the progression from model-based foundations to adaptive, safety-aware interaction control and cognitive programming interfaces.

contributing a necessary component to the overall research objective of enabling safe, adaptive, and intuitive human–robot collaboration in industrial environments.

Chapter 2 presents a comprehensive review of the relevant literature. It situates the thesis within the broader context of industrial and collaborative robotics, covering dynamic modeling and parameter identification, physical Human–Robot Interaction, admittance control, passivity and energy-based safety frameworks, and recent advances in artificial intelligence for robot programming. The chapter concludes by synthesizing the reviewed works into a conceptual framework that highlights the research gaps addressed by the thesis.

Chapter 3 focuses on the dynamic parameter identification of an industrial six-degree-of-freedom manipulator. It introduces the kinematic and dynamic modeling of the robot and details the experimental procedures used for parameter estimation and validation. This chapter provides a validated dynamic characterization of the platform, supporting model-based analysis and informing later discussions on interaction and safety metrics.

Chapter 4 addresses adaptive admittance control for safe hand-guiding and lead-through programming. After introducing the fundamentals of admittance control, the chapter presents a real-time adaptive strategy that modifies interaction dynamics in response to human-applied forces and safety-related variables. Experimental results demonstrate how adaptive admittance enables intuitive interaction while preserving stability and safety in industrial programming tasks.

Chapter 5 extends the interaction framework to collaborative scenarios. The chapter introduces an energy-based control formulation grounded in Power and Force Limiting principles and integrates passivity-based mechanisms to ensure stability under time-varying interaction dynamics. Both algorithmic and optimization-based adaptation strategies are presented, along with experimental and simulation results that illustrate the trade-offs between performance, responsiveness, and safety.

Chapter 6 explores the integration of artificial intelligence for intuitive robot programming. It investigates how high-level task descriptions expressed in natural language can be translated into executable robot behaviors through a structured and

safety-aware interface. The chapter presents a structured execution architecture that enables language-based robot programming while enforcing task-level constraints and safety-oriented execution policies.

Chapter 7 provides a comprehensive discussion and synthesis of the results obtained throughout the thesis. It compares the proposed approaches with baseline methods, highlights theoretical and practical contributions, and analyzes the limitations and implications of the work. The chapter emphasizes cross-cutting insights that emerge from the integration of modeling, control, safety, and cognitive interaction.

Finally, Chapter 8 concludes the thesis by summarizing its main contributions and situating them within the broader vision of human-centered and resilient industrial automation. The chapter discusses industrial implications and outlines future research directions, including predictive safety control, embodied intelligence, digital twin integration, and scalable frameworks for ethical and explainable human–robot collaboration.

Chapter 2

Literature Review

The objective of this chapter is not to provide an exhaustive survey of human–robot interaction, collaborative robotics, or robot programming, but to reconstruct the conceptual landscape within which the problem of *safe and intuitive robot programming* has been historically addressed. Rather than organizing the literature along disciplinary boundaries, the chapter adopts a problem-driven perspective, focusing on how modeling, control, safety, and interaction have been treated as partially independent design dimensions in existing research.

A central argument underpinning this review is that many of the limitations encountered in industrial human–robot collaboration do not stem from the absence of suitable control or safety techniques per se, but from their fragmented integration. Dynamic modeling, compliant control, safety supervision, and programming interfaces are often developed within distinct research traditions, each optimizing different objectives and operating at different levels of abstraction [4, 1, 7]. As a consequence, solutions that are effective within a narrow scope frequently fail to scale to realistic industrial scenarios involving physical interaction, safety certification, and usability constraints.

In line with the methodological progression outlined in Chapter 1, this literature review is structured to progressively move from low-level physical foundations to higher-level interaction and programming paradigms. The chapter first examines the evolution of industrial robotics toward collaborative and interaction-oriented applications, emphasizing programming effort as a central but often implicit bottleneck. It then reviews dynamic modeling and parameter identification as analytical foundations for interaction and safety reasoning, followed by a critical analysis of admittance-based control strategies for physical human–robot interaction. Subsequently, safety paradigms based on distance monitoring and energy limitation are discussed, together with passivity-based frameworks that enable stable interaction under time-varying dynamics. Finally, the chapter briefly introduces cognitive and language-based programming interfaces as emerging, complementary modalities for robot programming.

Throughout the chapter, emphasis is placed on identifying conceptual gaps and unresolved tensions in the literature, rather than on cataloguing individual methods. This perspective prepares the ground for the subsequent chapters, which address these gaps through a coherent progression from model-based analysis of robot dynamics and safety-relevant quantities to adaptive, safety-aware interaction control and structured cognitive programming interfaces.

2.1 From Industrial Automation to Collaborative Programming

Industrial robotics has historically been shaped by an automation paradigm centered on task repeatability, high precision, and strict separation between human operators and robotic systems. In this paradigm, robots are programmed offline or through teach pendants, and execute predefined trajectories within fenced or otherwise isolated workspaces [4]. Safety is primarily achieved through physical separation, emergency stop mechanisms, and conservative operational envelopes, rather than through continuous interaction-aware control.

As manufacturing systems have evolved toward higher product variability and shorter production cycles, the limitations of this paradigm have become increasingly apparent. The effort required to program and reprogram industrial robots has been repeatedly identified as a major barrier to flexibility, particularly in small- and medium-scale production environments [1, 7]. In this context, programming is no longer a purely technical activity, but a systemic constraint that directly affects deployment cost, downtime, and accessibility.

Collaborative robotics has emerged as a response to these limitations, proposing closer physical and functional integration between humans and robots within shared workspaces. Rather than replacing human operators, collaborative robots are intended to support them by combining robotic strength and precision with human adaptability and decision-making [1]. This shift has profound implications not only for mechanical design and safety certification, but also for how robots are programmed and instructed.

A key observation in the literature is that collaboration inherently redefines the notion of robot programming. In collaborative scenarios, especially those involving hand-guiding, lead-through programming, or cooperative manipulation, robot motion is often generated or shaped online through physical interaction rather than through predefined trajectories [13, 10]. As a result, programming and execution become temporally intertwined, and the traditional separation between programming phase and operational phase begins to break down.

From this perspective, physical interaction can be interpreted as a form of embodied programming, in which human intent is conveyed through forces, motion, and proximity rather than through symbolic code or geometric waypoints [10, 24].

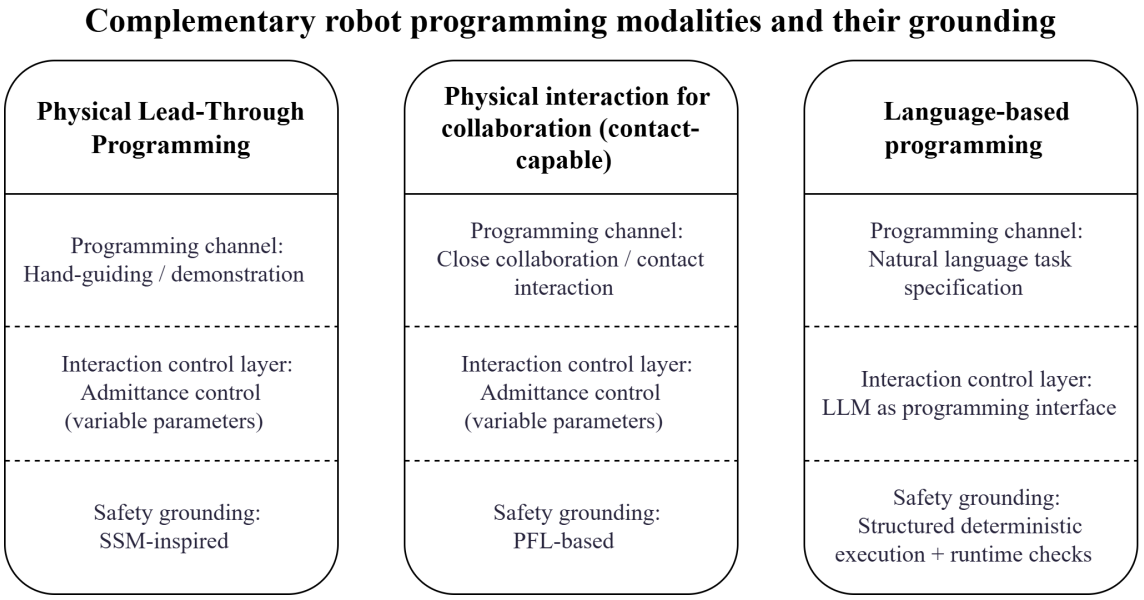


Figure 2.1: Conceptual positioning of the three complementary robot programming modalities investigated in this thesis. For each modality, the figure distinguishes the programming channel, the interaction-control layer, and the primary safety grounding (SSM supervision for separation-based physical programming; PFL constraints for contact-capable physical interaction; structured deterministic execution and runtime checks for language-based programming).

This interpretation highlights that programming effort is redistributed rather than eliminated: while the burden of explicit coding is reduced, new requirements arise in terms of compliant control, stability guarantees, and safety supervision during interaction.

Despite this conceptual shift, much of the industrial literature continues to treat programming interfaces, interaction control, and safety supervision as loosely coupled components. Lead-through programming is often presented as a usability feature, while safety mechanisms are introduced as external constraints enforced through monitoring or override strategies [25, 16]. This separation obscures the fact that, in physical human–robot interaction, programming, control, and safety are inherently interdependent.

Physical interaction-based programming, adaptive compliant control, and higher-level cognitive interfaces can be understood as complementary modalities that differ in abstraction level, expressiveness, and safety guarantees, rather than as competing alternatives. This interpretation motivates a systematic examination of how modeling, control, and safety frameworks support (or limit) different forms of intuitive robot programming, which constitutes the focus of the remainder of this chapter.

As summarized in Fig. 2.1, the programming modalities addressed in this thesis are complementary and differ primarily in their level of abstraction and in the mechanisms by which safety is grounded and enforced.

2.2 Dynamic Modeling and Parameter Identification

Dynamic modeling has long constituted a foundational component of robotic system analysis, providing a mathematical description of the relationship between joint motion, applied torques, and resulting forces at the end-effector [4, 26]. In its classical formulation, the rigid-body dynamics of serial manipulators express joint torques as a function of inertial, Coriolis and centrifugal, gravitational, and frictional effects. This structure has been extensively studied and forms the basis for simulation, feedforward compensation, and performance analysis in industrial robotics.

Within this framework, parameter identification aims to estimate the inertial parameter set of the manipulator (i.e., link masses, centers of mass, and inertia tensors) together with friction-related terms, in a form suitable for inverse-dynamics prediction and physically grounded analysis [4, 26, 27, 28]. Numerous studies have shown that nominal models provided by manufacturers are often insufficiently accurate for quantitative analysis, particularly when interaction forces, energy exchange, or torque-level behavior are of interest [28, 29]. Identification procedures based on linear parameterization of the dynamics and least-squares or optimization-based estimation have therefore become standard practice in both academic and industrial research [27, 28].

It is important to distinguish, however, between the different roles that dynamic models can play within a robotic system. In classical trajectory tracking and inverse dynamics control, accurate models are directly embedded in the control loop to cancel nonlinear dynamics and enforce desired closed-loop behavior [30, 31]. In contrast, many interaction-oriented control strategies (particularly those based on admittance or impedance formulations) do not require explicit dynamic models for stability or basic functionality [10]. As a result, the relevance of parameter identification for physical human–robot interaction is sometimes questioned or treated as secondary.

Nevertheless, the literature indicates that dynamic modeling and identification remain valuable in interaction scenarios, not because compliant control necessarily depends on a perfect model, but because models provide an analytical reference for computing and interpreting physically meaningful quantities used in assessment and monitoring. First, dynamic models enable the computation of physically meaningful quantities such as apparent inertia, kinetic energy, and power, which are central to interaction analysis and safety assessment [11, 12]. Second, model accuracy directly affects the interpretation of measured joint torques and interaction forces, particularly in sensorless or partially instrumented systems [13]. Third, uncertainties in inertial and friction parameters can propagate into safety-related estimations, potentially leading to either overly conservative behavior or underestimated risk [16, 32].

From a programming perspective, dynamic modeling also plays an indirect but significant role. Lead-through programming and hand-guiding rely on the assumption that the robot responds predictably to human-applied forces and that the resulting

motion can be reproduced or generalized reliably. While admittance control can provide compliant behavior without explicit model inversion, the consistency of physically taught trajectories depends on the overall predictability of the robot's response to interaction, which is shaped by sensing quality, controller bandwidth, and the chosen compliance behavior; dynamic models are mainly relevant here as a reference for analysis and for the estimation of safety-relevant quantities, rather than as a prerequisite for compliant motion generation [10, 13]. In this sense, dynamic modeling supports the *interpretability* of physical interaction as a programming modality.

These considerations motivate treating dynamic modeling and parameter identification as analytical foundations rather than as universal prerequisites for interaction control. Accurate models are not required for all compliant control strategies, but they provide a necessary reference frame for evaluating interaction dynamics, quantifying safety margins, and understanding the physical implications of interaction design choices and the safety-related quantities used to supervise them. This perspective aligns with the broader view that safe and intuitive robot programming emerges from the coherent integration of modeling, control, and safety reasoning, rather than from any single component in isolation.

It is worth noting that, while accurate dynamic models are central to many model-based control paradigms, a substantial body of interaction-oriented control research deliberately decouples stability and safety guarantees from model accuracy, favoring control formulations whose robustness relies on passivity, energy regulation, or interaction-level abstractions rather than on precise knowledge of the robot's inertial parameters.

2.3 Admittance for Physical Human–Robot Interaction

Physical human–robot interaction requires control strategies that explicitly account for external forces applied by the human operator and that shape the robot motion in a compliant and intuitive manner. Impedance- and admittance-based control frameworks have emerged as the dominant paradigms for this purpose, as they allow the dynamic relationship between force and motion to be specified explicitly [4, 10].

In impedance control, the robot is commanded to behave as a virtual mechanical impedance by directly regulating the relationship between motion errors and interaction forces. This approach is particularly effective in torque-controlled robots, where force feedback can be tightly integrated into the control loop [4]. However, many industrial robots operate under position or velocity control architectures that limit direct access to joint torques. In such cases, admittance control is often preferred, as it maps measured external forces into motion commands through a virtual mass–damper system without requiring torque-level actuation [10, 13]. This point is

not merely practical: hand-guiding a robot whose native control stack is not designed for physical interaction makes compliance an *added capability* that must coexist with industrial motion interfaces and safety supervision, rather than a built-in property of the actuation system [1, 10, 25].

Admittance control has therefore become a standard solution for hand-guiding and lead-through programming in industrial contexts. By tuning the virtual inertia and damping parameters, designers can trade off responsiveness against stability and robustness to noise [10]. Low apparent inertia improves transparency and reduces operator effort, while higher damping enhances stability and suppresses oscillations. This trade-off is intrinsic and cannot be eliminated through static tuning alone.

A substantial body of literature has demonstrated that fixed-parameter admittance control is insufficient for realistic interaction scenarios. Human behavior is inherently variable, and interaction conditions may change rapidly depending on task phase, operator intent, and proximity constraints [1, 25]. As a result, parameters that yield intuitive behavior in one situation may lead to excessive compliance or instability in another. This observation has motivated numerous adaptive and variable admittance strategies, in which inertia and damping parameters are modified online based on force magnitude, motion velocity, or estimated human intent [14, 33, 15].

While these approaches improve interaction quality, they also expose a critical limitation. Admittance parameters directly influence the robot's apparent mass and velocity response, which in turn affect the forces and energy exchanged during contact [31]. Consequently, adaptation mechanisms that are designed purely from an interaction or performance perspective may inadvertently compromise safety, particularly in close-proximity or contact-rich scenarios. Many existing methods address this issue through heuristic bounds or conservative parameter limits, rather than through explicit safety reasoning [25, 16].

From the perspective of robot programming, admittance control plays a dual role. On the one hand, it enables intuitive physical teaching by allowing operators to guide the robot directly, effectively embedding programming within interaction [13]. On the other hand, it defines the physical vocabulary through which intent is expressed: the same human-applied force can result in markedly different motion depending on the chosen admittance parameters. This dependency underscores that programming through physical interaction is inseparable from control design choices.

These considerations highlight the need for admittance control strategies that explicitly account for safety constraints and physical interaction limits, rather than treating them as external supervisory concerns. The literature reviewed in this section establishes admittance control as a powerful enabler of intuitive robot programming, while simultaneously revealing its limitations when safety, adaptability, and predictability must be guaranteed under varying interaction conditions. Addressing these limitations requires a closer examination of safety paradigms and energy-based interaction frameworks, which is the focus of the following sections.

2.4 Safety Paradigms in Physical Human–Robot Interaction: SSM and PFL

Safety in physical human–robot interaction is not a monolithic concept, but is articulated through distinct paradigms that impose fundamentally different constraints on robot behavior. Among these, Speed and Separation Monitoring (SSM) and Power and Force Limiting (PFL) represent two complementary yet conceptually divergent approaches to ensuring human safety during collaboration [3, 1].

SSM is based on the continuous monitoring of the relative distance between the human and the robot and enforces safety by regulating robot speed as a function of this distance. The underlying logic is preventive: physical contact is treated as an event to be avoided, and safety is ensured by guaranteeing sufficient separation or by reducing kinetic energy through velocity limitation before contact occurs [34, 35]. Within this paradigm, safety constraints are primarily geometric and kinematic, relying on perception systems and conservative assumptions about human motion and reaction time [36].

A key strength of SSM lies in its conceptual simplicity and compatibility with existing industrial safety infrastructures. By acting on velocity limits and separation thresholds, SSM can be integrated into a wide range of control architectures without requiring detailed modeling of contact dynamics [25]. However, this same abstraction introduces structural limitations. As human–robot distance decreases, SSM tends to enforce increasingly conservative behavior, often resulting in abrupt slowdowns or task interruptions that degrade interaction fluency and programming usability [16]. Moreover, SSM does not naturally accommodate scenarios in which intentional contact is required, such as hand-guiding or cooperative manipulation.

In contrast, PFL addresses safety by bounding the mechanical energy, force, and pressure exchanged during physical contact. Rather than avoiding contact altogether, PFL assumes that contact may occur and seeks to limit its consequences to within acceptable thresholds derived from biomechanical considerations [3, 12]. This shift has profound implications for control design, as safety constraints are no longer purely geometric but depend on the dynamic state of the robot, including velocity, apparent mass, and interaction dynamics [11].

Energy-based formulations of PFL reveal an intrinsic coupling between robot motion and safety limits. In particular, admissible velocities depend on the effective inertia experienced at the point of contact, which in turn is influenced by control parameters and configuration [6, 31]. As a result, safety cannot be enforced independently of interaction control: the same control law that shapes compliance also shapes the safety envelope. This interdependence distinguishes PFL from SSM and challenges traditional layered architectures in which safety is treated as an external supervisory function.

The literature highlights that, while PFL enables closer and more natural col-

laboration, it also demands more explicit coordination between safety reasoning and control design [16, 32]. Without such coordination, conservative parameter choices may negate the benefits of contact-based interaction, while aggressive tuning may violate safety margins. Unlike SSM, where safety constraints can often be enforced through velocity clamping, PFL requires continuous modulation of interaction dynamics to remain effective across operating conditions [34].

From a programming standpoint, the distinction between SSM and PFL is particularly significant. SSM constrains when and how physical programming can occur by restricting proximity and motion speed, whereas PFL constrains the admissible dynamics of interaction itself. This difference implies that programming through physical interaction is shaped not only by control responsiveness but also by the underlying safety paradigm. Understanding this distinction is essential for interpreting the limitations of fixed interaction parameters and motivates the exploration of control frameworks that explicitly regulate energy exchange, as discussed in the following section.

2.5 Energy- and Passivity-Based Control Frameworks

The challenges posed by time-varying interaction dynamics and energy-based safety constraints have motivated the adoption of passivity-based control frameworks in physical human–robot interaction. Passivity provides a system-theoretic notion of stability grounded in energy exchange, offering guarantees that remain valid even in the presence of uncertain or time-varying environments [4, 17].

In the context of human–robot interaction, passivity is particularly attractive because the human operator can be modeled as an energy-generating or energy-dissipating system whose behavior is neither fully predictable nor easily constrained [18]. By ensuring that the robot controller does not inject unbounded energy into the coupled system, passivity-based approaches can guarantee stable interaction regardless of operator behavior, provided that the overall energy balance is respected.

Energy tank architectures represent a prominent instantiation of this principle. In such frameworks, the robot controller is endowed with a virtual energy storage element that regulates how much energy can be injected into or dissipated by the system [18, 15]. When interaction parameters are modified online (such as during variable admittance control), the energy tank acts as a mediator, ensuring that adaptations do not violate passivity conditions. This mechanism enables stable time-varying behavior that would otherwise be difficult to guarantee.

The relevance of energy-based control extends beyond stability considerations. Energy tanks and related formulations provide a natural interface between interaction control and safety reasoning, particularly in the context of PFL. By explicitly tracking

and limiting kinetic and exchanged energy, these frameworks allow safety constraints to be expressed in physically meaningful terms that align with standard-based limits [6]. Importantly, this alignment does not imply automatic compliance with safety standards, but it facilitates systematic reasoning about how control parameters influence safety-relevant quantities.

Despite these advantages, passivity-based control should not be conflated with safety enforcement. Passivity guarantees stability of interaction, but it does not ensure that force, pressure, or energy thresholds remain within acceptable limits under all conditions [12]. Consequently, additional mechanisms are required to translate safety constraints into admissible energy budgets or control parameter bounds. The literature reflects ongoing efforts to bridge this gap through combinations of passivity, constraint enforcement, and supervisory control [32, 36].

From a programming perspective, energy- and passivity-based frameworks play a subtle but critical role. By enabling stable interaction under adaptive dynamics, they support forms of physical programming that would otherwise be too unpredictable or unsafe. At the same time, they expose the inherent trade-offs between responsiveness, stability, and safety, making explicit that intuitive interaction cannot be decoupled from rigorous energy management. These observations set the stage for examining how higher-level programming interfaces can be grounded in physically safe execution frameworks, which is addressed in the next section.

2.6 Language-Based Robot Programming as Cognitive Interface

While physical interaction provides a powerful means for conveying motion intent through embodied cues, it does not exhaust the space of intuitive robot programming modalities. A parallel research trajectory has investigated how higher-level cognitive interfaces can enable users to specify tasks, goals, and constraints in ways that are decoupled from continuous physical interaction. In this context, natural language has emerged as a particularly attractive medium due to its expressiveness, familiarity, and ability to convey abstract intent [8, 9].

Historically, language-based and symbolic robot programming has been studied within the broader field of task planning and human-aware decision-making. Early architectures focused on translating structured commands into symbolic plans that could be executed by robotic systems under explicit logical constraints [8, 37]. These approaches emphasized deliberation, transparency, and verifiability, but were often limited by the rigidity of predefined vocabularies and the difficulty of grounding abstract symbols in physical execution.

Recent advances in machine learning, and in particular in large language models, have renewed interest in language-based robot programming. Large language models

exhibit strong capabilities in parsing unstructured natural language, generating procedural descriptions, and composing multi-step plans from high-level instructions [38, 19]. As a result, they have been proposed as interfaces for generating robot code, task sequences, or action descriptions directly from user prompts [39, 40]. Surveys highlight their potential to reduce the cognitive and technical burden associated with traditional programming interfaces [41].

Despite this potential, the literature also underscores significant limitations that prevent language-based programming from being treated as a replacement for physical interaction or low-level control. Large language models are inherently probabilistic, may produce inconsistent outputs, and lack intrinsic guarantees of determinism or correctness [19, 20]. Moreover, they do not possess direct awareness of physical constraints, safety limits, or real-time system states unless explicitly coupled with external verification and execution layers [42]. As such, language models cannot be considered controllers, nor can their outputs be assumed to be compliant with industrial safety standards.

From the perspective of human–robot interaction, it is therefore more appropriate to interpret language-based programming as a distinct interaction modality, orthogonal to physical programming. Whereas physical interaction conveys intent through forces, motion, and proximity, language conveys intent through symbolic abstraction, task decomposition, and semantic constraints. These modalities differ not only in expressiveness, but also in their relationship to safety: physical interaction is governed by continuous dynamic constraints, while language-based interaction relies on structured execution and constraint enforcement to ensure predictable behavior [43, 44].

The emerging consensus in the literature is that effective language-based robot programming requires a layered architecture in which high-level, potentially ambiguous instructions are translated into structured and verifiable representations before execution [45, 46]. Safety in such systems is not achieved through formal certification of the language model itself, but through deterministic execution pipelines, explicit constraints, and supervision mechanisms that mediate between symbolic intent and physical action [42]. This perspective aligns with broader trends in embodied artificial intelligence, which emphasize grounding and constraint-based execution as prerequisites for reliable deployment [47].

In this sense, language-based programming complements rather than replaces physical interaction. It enables forms of task specification and reconfiguration that are difficult to express through direct manipulation alone, while remaining dependent on the underlying control and safety infrastructure for execution. Recognizing this orthogonality is essential for integrating cognitive interfaces into industrial robotic systems without conflating programming convenience with physical safety guarantees.

2.7 Synthesis and Research Gaps

The literature reviewed in this chapter reveals a rich and mature body of work addressing individual aspects of human–robot collaboration, including dynamic modeling, compliant interaction control, safety supervision, and intuitive programming interfaces. At the same time, it exposes persistent conceptual and architectural gaps that limit the deployment of safe and intuitive robot programming in realistic industrial scenarios.

A first gap concerns the role of dynamic modeling in interaction-oriented systems. While accurate dynamic models are widely recognized as essential for simulation, analysis, and safety-related estimation, they are often treated as either strictly necessary for control or largely irrelevant to interaction design. The literature lacks a coherent perspective that situates dynamic modeling as an analytical foundation that informs interaction quality and safety reasoning without imposing unnecessary constraints on control architecture selection.

A second gap emerges in the design of admittance-based interaction control. Variable and adaptive admittance strategies have been extensively studied to improve transparency and usability, yet their adaptation mechanisms are frequently decoupled from formal safety considerations. As a result, interaction parameters are often tuned heuristically, and safety constraints are enforced externally rather than being embedded in the interaction dynamics themselves.

A third gap lies in the relationship between safety paradigms and control design. Speed and Separation Monitoring and Power and Force Limiting impose fundamentally different constraints on robot behavior, yet many architectures treat them as interchangeable supervisory layers. The literature indicates that energy-based safety paradigms, in particular, require tighter integration between control parameters and safety limits, which is not adequately addressed by conventional layered approaches.

A fourth gap concerns the formalization of interaction adaptation under safety constraints. While passivity- and energy-based frameworks provide stability guarantees for time-varying interaction dynamics, they do not by themselves ensure compliance with safety thresholds. The systematic integration of safety constraints into adaptation mechanisms (particularly through explicit optimization or constraint-based formulations) remains an open research direction.

Finally, a gap exists in the integration of cognitive programming interfaces with physically grounded execution. Language-based robot programming has demonstrated promising capabilities for task specification and user interaction, but its safe deployment depends critically on structured execution frameworks that mediate between symbolic intent and physical action. The literature often treats these interfaces as standalone innovations, rather than as components that must be coherently aligned with interaction control and safety architectures.

Taken together, these gaps point to a central unresolved challenge: enabling robot programming modalities that are both intuitive for human operators and rigorously grounded in physical safety and system predictability. Addressing this challenge requires a progressive approach that does not seek a single universal solution, but rather integrates complementary paradigms across modeling, control, safety, and interaction. The subsequent chapters of this thesis adopt this perspective, advancing from model-based analysis of robot dynamics and safety-relevant quantities toward adaptive, safety-aware interaction control and structured cognitive programming interfaces within a unified conceptual framework.

Chapter 3

Dynamic Modeling and Parameter Identification of an Industrial Robot

Industrial robot programming through physical interaction inevitably exposes a tension between intuitive human guidance and the need for physically grounded interpretation of robot motion and effort. While many interaction control strategies deliberately abstract away the internal dynamics of the manipulator in favor of compliant, task-space formulations, the underlying physical system remains governed by inertial, gravitational, and frictional effects that directly influence torque generation, energy exchange, and mechanical loading. In industrial environments, where manipulators are typically characterized by high payload capacity, non-negligible link masses, and proprietary low-level control architectures, these effects cannot be dismissed as secondary.

Within this context, dynamic modeling and parameter identification retain a role that is conceptually distinct from interaction control design. Rather than serving as a prerequisite for compliant behavior, a validated dynamic model provides a physically interpretable description of how joint torques, motion, and energy arise from the robot's mechanical structure. This description is particularly relevant when interaction-based programming is deployed on non-collaborative industrial manipulators, where safety supervision, low-level control integration, and performance assessment must coexist with established industrial control loops.

This chapter addresses the dynamic modeling and parameter identification of the six-degree-of-freedom industrial robot used as the experimental platform for the subsequent studies. The objective is to establish a reliable analytical reference for validation and physical interpretation; the boundaries of its use with respect to interaction control are stated explicitly in Section 3.1. In doing so, the chapter contributes a foundational analytical layer that complements (rather than constrains) the interaction control and programming approaches introduced in the following

chapters.

3.1 Scope and Role of the Dynamic Model

Before introducing the mathematical formulation of the robot dynamics and the associated identification procedures, it is necessary to clarify the scope and role of the dynamic model within the overall architecture of this dissertation. This clarification is essential to avoid a common misinterpretation, namely that the identified model constitutes a functional requirement for the interaction control strategies developed in the subsequent chapters.

In classical robot control, dynamic models are often embedded directly within the control loop, for instance in inverse-dynamics or computed-torque schemes, where accurate knowledge of inertial and friction parameters is required to cancel nonlinear effects and impose desired closed-loop behavior. In contrast, the interaction-oriented control approaches adopted in this thesis (particularly admittance-based formulations for physical human-robot interaction) do not rely on explicit dynamic models for stability, passivity, or basic functionality. The compliant behavior of the robot during interaction is instead governed by deliberately simplified mass–damper relationships defined at the interaction port, combined with safety supervision mechanisms grounded in either SSM or PFL principles.

The role of dynamic modeling and parameter identification in this dissertation is instead twofold. First, the identified model is suitable for supporting low-level torque feedforward compensation within the industrial control architecture (where such a channel is available), thereby reducing the burden on decentralized feedback loops without altering their structure. Second, the model is used as an offline reference to compute and interpret mechanically grounded quantities (e.g., joint-space kinetic energy and direction-dependent apparent inertia) when discussing experimental safety margins; it is not required for online safety enforcement.

By explicitly separating the analytical and interpretative function of the dynamic model from the design of interaction controllers, this chapter establishes a clear conceptual boundary between modeling, control, and safety paradigms. This separation is maintained throughout the remainder of the thesis and reflects a deliberate design choice aimed at preserving both robustness and interpretability in safety-oriented human–robot interaction.

3.2 Kinematic and Dynamic Modeling of the Industrial Manipulator

This section introduces the kinematic and dynamic model of the six-degree-of-freedom industrial manipulator considered in this thesis. The formulation follows standard rigid-body modeling assumptions and serves as a physically consistent analytical reference for parameter identification, low-level torque compensation, and the interpretation of interaction-related quantities.

3.2.1 Generalized Coordinates and Kinematics

The configuration of the manipulator is described by the joint position vector

$$\boldsymbol{q} \in \mathbb{R}^6, \tag{3.1}$$

where each component represents the angular position of a revolute joint. The corresponding joint velocity and acceleration vectors are denoted as

$$\dot{\boldsymbol{q}} \in \mathbb{R}^6, \qquad \ddot{\boldsymbol{q}} \in \mathbb{R}^6. \tag{3.2}$$

The kinematic structure of the robot is defined through a serial chain of rigid links connected by actuated joints. Coordinate transformations between successive link frames are described using standard Denavit–Hartenberg conventions, which allow the computation of link positions, orientations, and Jacobians as functions of the joint configuration. These kinematic quantities constitute the foundation for the dynamic formulation introduced in the following subsection.

3.2.2 Rigid-Body Dynamic Model

Under the assumption of rigid links and ideal joint actuation, the joint-space dynamics of the manipulator can be expressed in the standard form

$$\boldsymbol{\tau} = \mathbf{B}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_f(\dot{\boldsymbol{q}}), \tag{3.3}$$

where:

- $\boldsymbol{\tau} \in \mathbb{R}^6$ is the vector of joint torques,
- $\mathbf{B}(\boldsymbol{q}) \in \mathbb{R}^{6\times 6}$ is the symmetric positive-definite joint-space inertia matrix,
- $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{6\times 6}$ collects Coriolis and centrifugal effects,
- $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^6$ represents gravitational torques,

- $\boldsymbol{\tau}_f(\dot{\boldsymbol{q}}) \in \mathbb{R}^6$ models joint friction effects.

The inertia matrix $\mathbf{B}(\boldsymbol{q})$ captures the distribution of mass and inertia across the robot structure and depends on the link masses, centers of mass, and inertia tensors. The matrix $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is defined such that the matrix $\dot{\mathbf{B}}(\boldsymbol{q}) - 2\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is skew-symmetric, ensuring energy consistency of the model. The gravity vector $\boldsymbol{g}(\boldsymbol{q})$ accounts for the effect of gravitational forces acting on each link.

Joint friction is modeled as the superposition of viscous and Coulomb components, yielding

$$\boldsymbol{\tau}_f(\dot{\boldsymbol{q}}) = \mathbf{F}_v \, \dot{\boldsymbol{q}} + \mathbf{F}_s \, \mathrm{sgn}(\dot{\boldsymbol{q}}), \tag{3.4}$$

where $\mathbf{F}_v \in \mathbb{R}^{6\times6}$ and $\mathbf{F}_s \in \mathbb{R}^{6\times6}$ are diagonal matrices containing viscous and Coulomb friction coefficients, respectively.

3.2.3 Linear Parameterization for Identification

For the purpose of parameter identification, the dynamic model in (3.3) can be rewritten in a form that is linear with respect to a vector of dynamic parameters. Specifically, the joint torques can be expressed as

$$\boldsymbol{\tau} = \mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \, \boldsymbol{\pi}, \tag{3.5}$$

where:

- $\mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^{6\times p}$ is the regression matrix,
- $\boldsymbol{\pi} \in \mathbb{R}^p$ is the vector of dynamic parameters.

$p \in \mathbb{N}$ denotes the number of (candidate) dynamic parameters included in the chosen parameterization of the rigid-body model (e.g., masses, centers of mass, link inertial parameters, motor inertias, and friction coefficients). The parameter vector $\boldsymbol{\pi}$ collects inertial properties of the links (masses, centers of mass, inertia tensor components), motor inertias, and friction coefficients. While the explicit structure of $\mathbf{Y}$ depends on the chosen dynamic formulation, the Newton–Euler approach is adopted in this work due to its recursive structure and computational efficiency, which are well suited to industrial manipulators with complex mechanical features such as joint couplings and load-assisting springs [4].

The linear parameterization in (3.5) constitutes the basis for the static and dynamic identification procedures presented in Section 3.3. As clarified in Section 3.1, this linear parameterization is introduced for identification and analysis, and it is not a functional dependency of the interaction controllers developed in later chapters.

3.3 Dynamic Parameter Identification Procedure

This section describes the procedure adopted to identify the dynamic parameters of the industrial manipulator introduced in Section 3.2. The identification methodology is grounded in the linear parameterization of the robot dynamics presented in (3.5) and follows a structured two-stage approach consisting of static and dynamic identification phases. The formulation and implementation closely follow the methodology presented in [21], which is here reformulated and contextualized to serve the analytical role of this dissertation.

The objective of the identification process is to obtain a physically consistent set of dynamic parameters that accurately reproduces the observed joint torques under known motion conditions.

3.3.1 Linear Regression Formulation

Starting from the linear parameterization of the joint-space dynamics,

$$\boldsymbol{\tau} = \mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\, \boldsymbol{\pi}, \tag{3.6}$$

the identification problem can be cast as the estimation of the parameter vector

$$\boldsymbol{\pi} \in \mathbb{R}^p, \tag{3.7}$$

given measurements of joint positions, velocities, accelerations, and torques.

The regression matrix

$$\mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^{6\times p} \tag{3.8}$$

is constructed using a recursive Newton–Euler formulation, which enables an efficient computation of the dynamic contributions associated with each link. This choice is particularly suited to industrial manipulators characterized by non-negligible link masses, joint couplings, and auxiliary mechanical elements such as load-assisting springs [21].

To improve numerical conditioning and ensure sufficient excitation of all identifiable parameters, the regression matrix is evaluated over a set of N samples collected along suitably designed trajectories. Stacking the resulting equations yields the global regression problem

$$\boldsymbol{\tau}_{\text{tot}} = \mathbf{Y}_{\text{tot}}\, \boldsymbol{\pi}, \tag{3.9}$$

where

$$\boldsymbol{\tau}_{\text{tot}} \in \mathbb{R}^{6N}, \qquad \mathbf{Y}_{\text{tot}} \in \mathbb{R}^{6N\times p}. \tag{3.10}$$

3.3.2 Separation of Static and Dynamic Identification

The dynamic parameter vector $\boldsymbol{\pi}$ generally contains parameters associated with both configuration-dependent and motion-dependent effects. In order to improve identifiability and reduce coupling between parameters, the identification procedure is decomposed into two sequential phases.

Static Identification. The static identification phase targets parameters that contribute to joint torques under quasi-static conditions, namely link masses and centers of mass. By considering a set of static configurations satisfying

$$\dot{\boldsymbol{q}} = \mathbf{0}, \qquad \ddot{\boldsymbol{q}} = \mathbf{0}, \tag{3.11}$$

the dynamic model reduces to a gravity-dominated form, and the regression problem simplifies accordingly. This allows the estimation of mass-related parameters with reduced sensitivity to noise and modeling uncertainties.

Dynamic Identification. Once mass and center-of-mass parameters have been estimated, the dynamic identification phase focuses on motion-dependent quantities, including inertia tensor components, motor inertias, and friction coefficients. In this phase, excitation trajectories involving nonzero velocities and accelerations are employed, and the previously identified static parameters are treated as known quantities.

This staged approach reduces parameter coupling and improves numerical robustness, particularly in manipulators with mechanical couplings and auxiliary elastic elements [21].

3.3.3 Regression Matrix Reduction and Identifiability

The global regression matrix $\mathbf{Y}_{\text{tot}}$ is generally rank-deficient due to structural dependencies among dynamic parameters. As a result, not all elements of $\boldsymbol{\pi}$ are independently identifiable. Following standard practice, the regression matrix is reduced to a minimal form by eliminating non-identifiable parameters and grouping linearly dependent terms.

Let

$$\boldsymbol{\pi}_{\text{min}} \in \mathbb{R}^{p_{\text{min}}} \tag{3.12}$$

denote the minimum identifiable parameter vector, and

$$\mathbf{Y}_{\text{min}} \in \mathbb{R}^{6N \times p_{\text{min}}} \tag{3.13}$$

the corresponding reduced regression matrix. The identification problem then becomes

$$\boldsymbol{\tau}_{\text{tot}} = \mathbf{Y}_{\text{min}}\, \boldsymbol{\pi}_{\text{min}}. \tag{3.14}$$

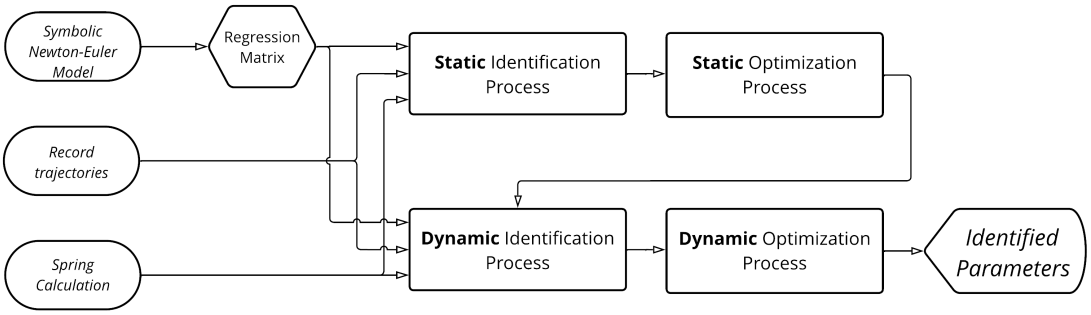


Figure 3.1: Overview of the dynamic parameter identification workflow adopted in this thesis. The procedure is structured into a static identification phase, targeting gravity-related parameters under quasi-static conditions, and a dynamic identification phase, focusing on inertial and friction-related effects using excitation trajectories. Recorded trajectories provide the measured joint signals used to evaluate and stack the regression matrix over N samples. The resulting regression problem is solved through least-squares estimation and refined via constrained optimization to enforce physical consistency, followed by offline validation against measured joint torques.

This reduction step ensures that the estimated parameters are uniquely determined by the available data and avoids ill-conditioned least-squares solutions.

3.3.4 Least-Squares Estimation

The minimum parameter vector $\boldsymbol{\pi}_{\min}$ is estimated by solving the least-squares problem

$$\hat{\boldsymbol{\pi}}_{\min} = \arg\min_{\boldsymbol{\pi}_{\min}} \left\| \mathbf{Y}_{\min}\, \boldsymbol{\pi}_{\min} - \boldsymbol{\tau}_{\text{tot}} \right\|^2, \tag{3.15}$$

which admits the closed-form solution

$$\hat{\boldsymbol{\pi}}_{\min} = \left(\mathbf{Y}_{\min}^{\top} \mathbf{Y}_{\min} \right)^{-1} \mathbf{Y}_{\min}^{\top} \boldsymbol{\tau}_{\text{tot}}, \tag{3.16}$$

provided that $\mathbf{Y}_{\min}^{\top} \mathbf{Y}_{\min}$ is full rank.

The resulting parameter estimates are subsequently refined through constrained optimization to enforce physical consistency, such as positivity of link masses and positive definiteness of inertia tensors, following the approach detailed in [21]. This refinement step improves interpretability and ensures compatibility with simulation and low-level torque computation frameworks.

The overall identification workflow, including the separation between static and dynamic phases and the subsequent optimization and validation steps, is summarized in Fig. 3.1.

3.3.5 Validation and Role within the Dissertation

The identified parameters are validated by comparing measured joint torques with model-predicted torques over trajectories not used during identification. The quality of the identification is assessed using standard error metrics and qualitative inspection of torque profiles.

The relationship between the identified model and the interaction control architectures is governed by the boundary stated in Section 3.1; the present section focuses exclusively on the estimation procedure.

3.4 Experimental Setup and Data Acquisition

This section describes the experimental setup and data acquisition procedure adopted for the identification of the dynamic parameters discussed in Section 3.3. The experiments are conducted on a six-degree-of-freedom industrial serial manipulator specifically selected for its relevance to heavy-duty industrial programming scenarios. The robot is introduced and characterized in detail in this section, as its mechanical structure directly influences the identification strategy and the interpretation of the resulting model.

3.4.1 Industrial Robot Platform

The experimental platform is a six-degree-of-freedom serial industrial manipulator designed for high-payload applications and characterized by a rigid mechanical structure and non-negligible link masses. Unlike lightweight collaborative robots, the system incorporates several mechanical features that significantly affect its joint-space dynamics and therefore require explicit consideration during parameter identification.

First, the manipulator integrates two load-compensation springs positioned between joints 1 and 3; their elongation is driven by the motion of joint 2. These springs introduce configuration-dependent torque contributions that partially counterbalance gravitational effects on the upper links. As a result, the measured joint torques do not purely reflect rigid-body dynamics but include elastic contributions that must be isolated or compensated during the identification process.

Second, the robot exhibits mechanical coupling between specific joint pairs. In particular, the motion and torque transmission of joints 2 and 3, as well as joints 4 and 5, are not independent but linked through the internal transmission architecture. This coupling implies that the torque applied by a motor may contribute to multiple joint torques, leading to a non-diagonal transmission relationship between motor torques and joint torques. Consequently, joint torques used for identification must be reconstructed by explicitly accounting for the coupling structure rather than

assuming one-to-one actuation.

Third, the actuation system relies on industrial gear reducers with non-unit efficiency and non-negligible no-load torque losses. These effects introduce systematic torque offsets and scaling factors that depend on the reducer characteristics and differ across joint groups. In the considered robot, different reducer families are used for proximal and distal joints, resulting in distinct efficiency and loss parameters that must be modeled separately to avoid bias in the identified inertial and friction parameters.

Finally, small misalignments of the robot base with respect to the gravity vector, typically negligible in idealized models, can generate measurable gravitational torque components on joints that would otherwise not be gravity-loaded. To account for this effect, the identification framework allows for the inclusion of base orientation parameters, ensuring that gravitational contributions are consistently propagated through the Newton–Euler formulation.

As illustrated in Fig. 3.2, the manipulator features mechanical couplings between joint pairs and load-compensation springs acting on the proximal joints, which directly influence the observed joint torque distribution.

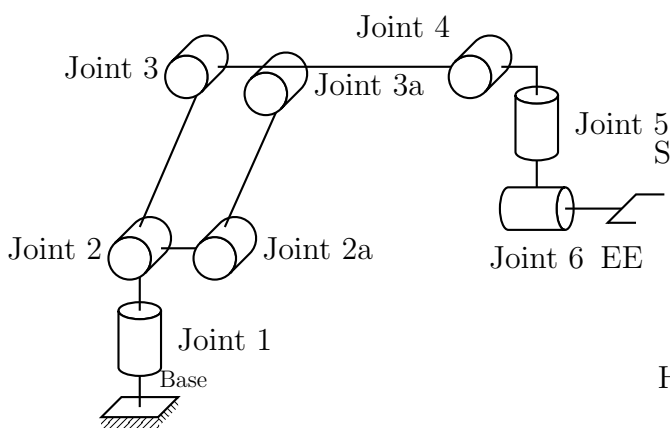
All these elements contribute to shaping the effective torque balance observed at the joints and therefore directly influence the structure of the regression model used for identification. These mechanical features are explicitly incorporated in the identification formulation adopted in this chapter, consistent with the experimental methodology reported in [21].

3.4.2 Measured Signals and Sampling

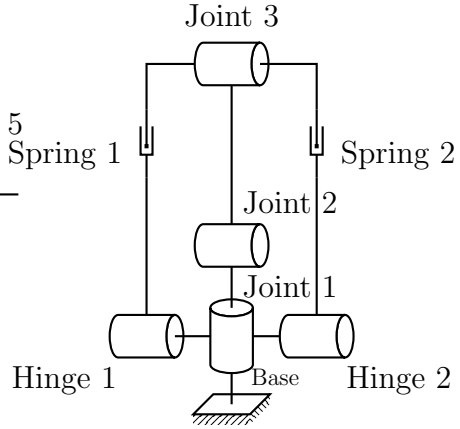
During the identification experiments, the following joint-level signals are acquired:

- joint positions $\boldsymbol{q}(t) \in \mathbb{R}^6$,
- joint velocities $\dot{\boldsymbol{q}}(t) \in \mathbb{R}^6$,
- joint accelerations $\ddot{\boldsymbol{q}}(t) \in \mathbb{R}^6$,
- motor or joint torques $\boldsymbol{\tau}(t) \in \mathbb{R}^6$.

All signals are sampled at a fixed sampling frequency sufficiently high to capture the relevant dynamic behavior of the system. Joint accelerations are obtained either through numerical differentiation of velocity measurements or directly from the robot controller when available. To mitigate noise amplification due to differentiation, appropriate filtering and smoothing techniques are applied during preprocessing, ensuring a suitable trade-off between signal fidelity and noise suppression.



(a) The industrial robot diagram illustrates the positioning of joints and highlights the coupling between joints 2 and 3 through a closed-kinematic chain (Joint 2a and 3a). The coupling between Joints 4 and 5 (a bevel pair) and Springs are not depicted in this diagram.



(b) The industrial robot spring layout details the placement of springs within the robot's structure, specifically between joints 1 and 3. These springs extend as joint 2 moves, producing a configuration-dependent torque about Joint 2.

Figure 3.2: Mechanical structure of the six-degree-of-freedom industrial manipulator, highlighting joint couplings and load-compensation springs that affect joint-space dynamics.

3.4.3 Static Identification Data Collection

For the static identification phase, a set of quasi-static robot configurations is selected to excite gravity-related torque components while minimizing dynamic effects. Each configuration satisfies

$$\dot{\boldsymbol{q}} \approx \mathbf{0}, \qquad \ddot{\boldsymbol{q}} \approx \mathbf{0}, \tag{3.17}$$

so that joint torques are dominated by gravitational contributions.

The selected configurations are distributed throughout the robot workspace to ensure sufficient excitation of all identifiable mass-related parameters. At each configuration, joint torques are recorded once steady-state conditions are reached. The resulting dataset forms the basis for the static identification stage described in Section 3.3.

3.4.4 Dynamic Identification Trajectories

Dynamic identification requires excitation of inertial and friction-related effects. To this end, a set of joint-space trajectories is designed to span a broad range of velocities and accelerations while remaining within the robot's operational limits. The trajectories are executed at different speed levels to improve identifiability of

viscous and Coulomb friction components.

Let

$$\boldsymbol{q}(t), \quad \dot{\boldsymbol{q}}(t), \quad \ddot{\boldsymbol{q}}(t), \qquad t \in [0, T], \tag{3.18}$$

denote a generic excitation trajectory of duration T. Multiple trajectories are executed, and the corresponding measurements are concatenated to form the global dataset used to construct the stacked regression problem introduced in Section 3.3.

Special care is taken to avoid trajectories that induce excessive vibrations or saturate the actuators, as such conditions may introduce unmodeled effects that degrade identification quality.

3.4.5 Data Preprocessing and Consistency Checks

Prior to parameter estimation, the acquired data are subjected to preprocessing steps aimed at improving numerical robustness and physical consistency. These steps include:

- synchronization of position, velocity, acceleration, and torque signals,
- filtering of high-frequency noise components,
- removal of transient segments associated with trajectory start-up and termination,
- consistency checks to discard samples affected by saturation or measurement anomalies.

The resulting dataset constitutes the input to the regression and optimization procedures described in Section 3.3. By enforcing a clear separation between data acquisition and parameter estimation, the experimental methodology ensures repeatability and transparency, which are essential for the analytical role played by the identified model within this dissertation.

3.5 Validation Results and Discussion

This section presents the validation of the identified dynamic parameters and discusses their implications within the scope defined in Chapter 3. The objective of the validation is twofold: first, to assess the capability of the identified model to reproduce measured joint torques under representative operating conditions; second, to clarify the extent to which the resulting model supports physical interpretation and analytical reasoning.

3.5.1 Torque Reconstruction Accuracy

Validation is performed by comparing the joint torques predicted by the identified dynamic model with the torques measured on the robot during trajectories not used for parameter estimation. Given the measured joint states

$$\boldsymbol{q}(t), \quad \dot{\boldsymbol{q}}(t), \quad \ddot{\boldsymbol{q}}(t), \tag{3.19}$$

the predicted torque vector is computed as

$$\hat{\boldsymbol{\tau}}(t) = \mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\, \hat{\boldsymbol{\pi}}, \tag{3.20}$$

where $\hat{\boldsymbol{\pi}}$ denotes the estimated parameter vector obtained through the procedure described in Section 3.3.

The comparison between measured torques $\boldsymbol{\tau}(t)$ and reconstructed torques $\hat{\boldsymbol{\tau}}(t)$ is evaluated using standard error metrics computed over the validation dataset. In particular, the root mean square error (RMSE) for each joint is defined as

$$\mathrm{RMSE}_i = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \left(\tau_i[k] - \hat{\tau}_i[k]\right)^2}, \tag{3.21}$$

where k indexes the sampled data points and $N \in \mathbb{N}$ is the number of samples in the considered validation dataset.

The validation results are reported separately for static and dynamic conditions. Static validation is summarized in Table 3.1, whereas Fig. 3.3 reports a representative time-domain comparison for dynamic excitation trajectories. For joints primarily influenced by gravitational effects, the reconstructed torque profiles accurately capture the amplitude and sign of the measured torques across the workspace, indicating that the identified mass and center-of-mass parameters provide a consistent representation of the robot's weight distribution. In addition to RMSE, the standard deviation of the torque reconstruction error (SDE) is reported to quantify the dispersion of residuals around the mean error.

During dynamic validation trajectories, the identified model reproduces the dominant inertial and friction-related contributions, yielding torque profiles that closely follow the measured signals in terms of phase and overall magnitude. For proximal joints, which experience higher inertial loading, the reconstructed torques capture both acceleration-dependent peaks and steady-state friction effects. Distal joints exhibit smaller absolute torque levels and correspondingly lower reconstruction errors.

Quantitatively, the per-joint RMSE values in Table 3.1 support the intended analytical use of the model, with the largest residuals concentrated on proximal joints where coupling and elastic compensation effects most strongly shape the torque balance. Complementarily, Fig. 3.3 illustrates the time-domain structure

Static Identification		
	RMSE (Nm)	SDE
Joint 1	13.03	12.54
Joint 2	32.73	25.59
Joint 3	82.59	74.14
Joint 4	4.11	3.07
Joint 5	3.75	3.73
Joint 6	0.75	0.52

Table 3.1: Static validation results of the industrial robot, reported as per-joint RMSE values (Nm), and per-joint SDE (Standard Deviation of Error), computed over quasi-static configurations.

of the residuals under dynamic excitation. These discrepancies remain consistent across repeated trajectories and do not indicate instability or systematic bias in the identified parameters, but rather reflect the simplifying assumptions adopted in the rigid-body and friction modeling.

For dynamic validation, the thesis reports representative time-domain comparisons to emphasize residual structure and regime-dependent deviations, rather than compressing performance into a single scalar metric.

3.5.2 Interpretation of Residual Errors

Despite the overall accuracy of the reconstructed torques, residual discrepancies are observed for specific joints and motion regimes. These discrepancies are not unexpected in industrial manipulators characterized by complex mechanical structures and proprietary low-level control loops.

Several contributing factors can be identified. First, joint couplings introduce torque redistribution effects that are difficult to model perfectly using lumped-parameter formulations. Second, load-assisting springs generate configuration dependent torques whose effective behavior may deviate from idealized models, particularly in the presence of wear or unmodeled compliance. Third, friction phenomena exhibit nonlinearities and hysteresis effects that are only approximated by the static and viscous friction model adopted in (3.4).

In particular, the observed residual discrepancies are consistent with the combined effects of joint torque redistribution due to mechanical couplings, configuration-dependent elastic contributions from load-compensation springs, and reducer-related losses, whose simplified modeling represents a deliberate trade-off between physical fidelity and analytical tractability within the scope of this chapter.

Importantly, these residual errors do not undermine the analytical role of the identified model. The purpose of the model is not to achieve exact torque prediction

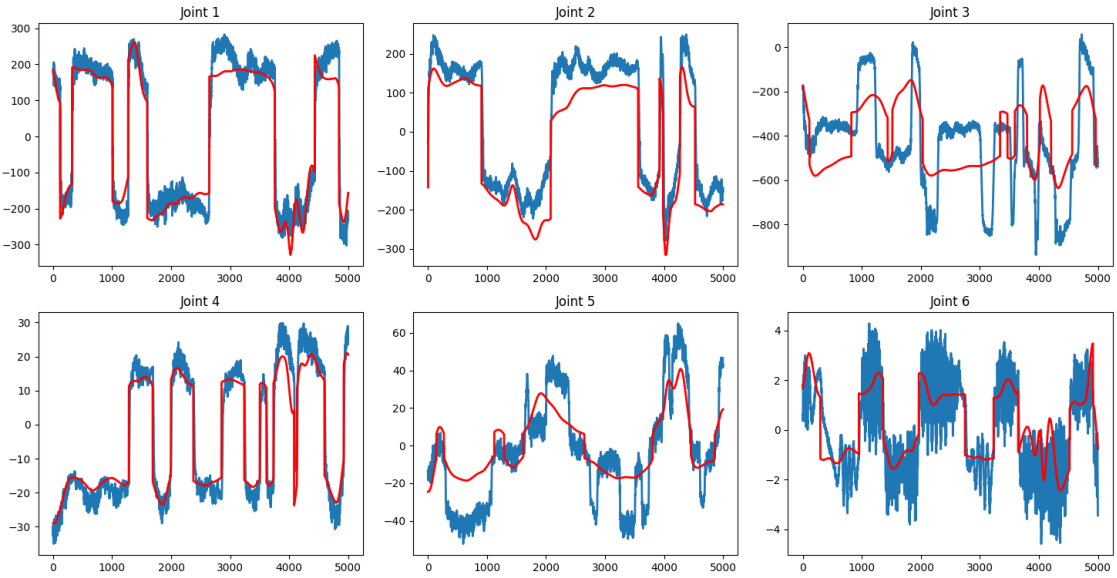


Figure 3.3: Dynamic torque validation on a representative excitation trajectory. For each joint, the measured joint torque $\tau_i[k]$ (blue) is compared against the reconstructed torque $\hat{\tau}_i[k]$ (red) computed from the identified model. The horizontal axis reports the sample index k (uniform sampling), and torques are expressed in Nm.

under all operating conditions, but to provide a physically interpretable approximation that captures the dominant dynamic contributions and enables meaningful validation of interaction-related quantities.

3.5.3 Implications for Safety-Oriented Analysis

One of the key motivations for dynamic modeling in this dissertation is to support the interpretation of physical quantities relevant to safety-oriented reasoning, such as apparent inertia, kinetic energy, and power exchange. The validated dynamic model provides a consistent mapping between joint-space variables and these quantities, allowing analytical assessments that would otherwise rely on ad hoc assumptions or overly conservative bounds.

Consistent with the architectural boundary established in Section 3.1, the safety mechanisms developed in subsequent chapters (whether based on speed and separation monitoring or on power and force limiting) do not require the identified model to function correctly. Safety enforcement is implemented using interaction-level abstractions and measurable signals, ensuring robustness with respect to modeling uncertainties. The identified model instead serves as a reference framework for understanding how mechanical properties influence interaction behavior, rather than as a real-time computational dependency.

3.5.4 Limitations and Scope of Validity

The identification and validation results presented in this chapter are subject to several limitations that must be acknowledged. The identified parameters are specific to the considered robot configuration and may vary with changes in payload, wear, or maintenance conditions. Furthermore, the identification procedure assumes rigid links and neglects high-frequency elastic effects, which may become relevant under aggressive motion profiles.

These limitations reinforce the architectural separation adopted in this dissertation. By avoiding reliance on accurate dynamic models for interaction control and safety enforcement, the proposed programming and control frameworks remain applicable even when model accuracy degrades. The dynamic model developed in this chapter should therefore be interpreted as an analytical support tool, rather than as a prerequisite for safe physical human–robot interaction.

3.5.5 Summary and Transition to Interaction Control

In summary, this chapter has presented the dynamic modeling, parameter identification, and validation of a six-degree-of-freedom industrial manipulator. The identified model demonstrates sufficient accuracy to support low-level torque compensation, analytical validation, and physically grounded interpretation of interaction phenomena. At the same time, its role is explicitly confined to analysis and validation, preserving the independence of the interaction control strategies introduced in the following chapters.

The next chapter builds upon this foundation by introducing admittance-based physical interaction control under speed and separation monitoring constraints. In contrast to the modeling-centric focus of the present chapter, the emphasis will shift toward interaction regulation and safety enforcement using model-light, energy-consistent control formulations.

Chapter 4

Admittance-Based Lead-Through Programming under Distance-Based Safety Supervision

Safety scope note. The supervision strategy in this chapter is described as *SSM-inspired*: human–robot geometric distance is used as the primary variable to modulate interaction dynamics, drawing conceptually from the Speed and Separation Monitoring (SSM) paradigm in ISO 10218 and ISO/TS 15066. This does *not* imply ISO certification, compliance with the specific safety functions mandated by those standards (e.g., safety-rated monitored stop, certified stopping-distance computation), or readiness for unprotected factory deployment. Any real-world industrial deployment would require additional independently certified safety layers. This boundary is maintained throughout the chapter and discussed explicitly in Section 4.7.

This chapter addresses the problem of *safe and intuitive lead-through programming* of an industrial manipulator through physical human–robot interaction. The considered scenario is deliberately *non-collaborative* in the sense of industrial safety practice: the robot is a standard high-payload manipulator operating under the safety requirements and integration constraints prescribed for industrial robots, where sustained physical contact cannot be treated as an admissible mode of operation and where the interaction layer must coexist with proprietary low-level control and safety-certified hardware/software components [48, 49]. Within this setting, the central technical question is not whether physical guidance is possible (it is) but rather how it can be made (i) sufficiently transparent to be usable as a programming modality, while (ii) remaining compatible with a *distance-aware* safety supervision that discourages unsafe proximity without resorting to disruptive discrete actions.

The chapter is positioned as a direct conceptual continuation of Chapter 3. There, the manipulator dynamic parameters were identified and validated to obtain

a physically interpretable model of the plant. Here, that model is *not* elevated to a functional dependency of the interaction controller. Instead, it plays an architectural role that is both narrower and, in an industrial context, more defensible: (i) it supports analysis and validation of interaction-related quantities (e.g., power/energy consistency, mechanical loading, and physically meaningful safety metrics), and (ii) it can be employed within the robot-side motion control stack as a feedforward component, in combination with feedback regulation, to improve tracking of the motion references generated by the interaction layer. In other words, dynamic modeling is exploited to *interpret* and *support* the closed-loop behavior of the overall pipeline, not to compute the admittance law itself.

From a safety standpoint, the chapter adopts a supervision logic grounded in *geometric separation*. The relative spatial distance between the operator and the robot is continuously estimated by an external perception system, and safety-relevant constraints are derived as functions of such separation. This is aligned with the *philosophy* of Speed and Separation Monitoring (SSM), where safety is achieved by regulating the robot behavior based on human–robot distance and the ability to maintain separation [49]. However, the implementation discussed here is *not* a canonical SSM realization in the strict standards sense: rather than enforcing safety through certified stopping-distance computation, safety-rated monitored stop, or direct velocity scaling at the safety layer, the supervision acts by continuously modulating the *interaction dynamics*. This choice is intentional. In lead-through programming, abrupt velocity clamps or discrete stops can degrade usability, induce discontinuities in the human-perceived dynamics, and paradoxically elicit compensatory operator behaviors (e.g., increased pushing) that are undesirable precisely near proximity limits.

Concretely, safety supervision is integrated into the interaction layer by adapting the parameters of a Cartesian admittance filter as a function of distance-derived metrics. As the operator approaches the robot, the virtual inertia and damping of the admittance model are reshaped so that the robot progressively becomes less responsive to externally applied wrenches, limiting its kinetic interaction capability while preserving continuity of motion and preserving the semantics of "guidance" rather than "commanded motion inhibition." This chapter focuses on this *parameter adaptation* mechanism: how proximity information is converted into bounded, continuous modifications of the admittance parameters, and how these modifications propagate through the full industrial control chain.

A critical consequence of this design is that safety must be framed with conceptual discipline. The adopted supervision is *distance-aware* and *SSM-inspired*, but it does not claim ISO-certified SSM compliance because it does not implement the standard-prescribed safety functions as such [49]. Likewise, it must not be conflated with Power and Force Limiting (PFL): the robot is not treated as collaborative, the safety logic is not grounded in contact tolerance, and the controller does not attempt to guarantee injury-related bounds under collision [50]. The safety claim of this chapter is therefore architectural and conditional: *given* a continuously available and sufficiently conservative distance estimate, the interaction behavior is shaped

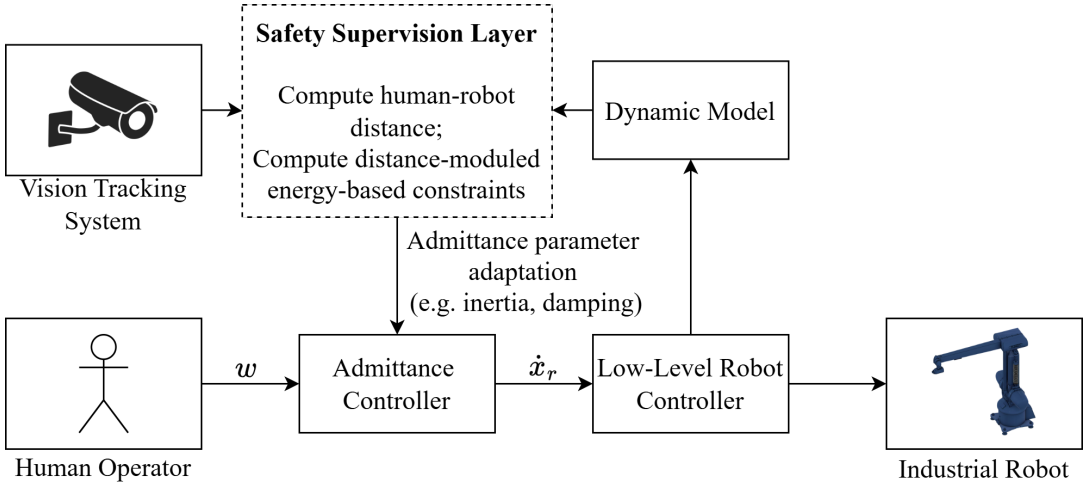


Figure 4.1: Layered architecture for admittance-based lead-through programming under distance-modulated energy-based safety supervision. Safety supervision modulates interaction dynamics through parameter adaptation, while motion execution is delegated to the robot low-level controller.

so as to discourage unsafe proximity and reduce the robot's capacity to generate rapid motion under human-applied forces, without relying on contact-based safety assumptions.

Finally, the chapter makes explicit the separation of concerns that underpins the whole dissertation: (i) the perception system estimates geometric quantities (human pose abstraction and minimum human–robot distance), potentially relying on efficient distance computations between geometric primitives [51]; (ii) the safety supervisor maps these quantities into admissible interaction dynamics via bounded parameter adaptation; (iii) the admittance controller maps human-applied wrenches into smooth motion references; and (iv) the robot-side low-level controller executes these references, optionally aided by model-based feedforward terms for tracking quality [52].

4.1 Physical Lead-Through Programming as an Interaction Problem

Physical lead-through programming aims at enabling an operator to specify robot motion by directly guiding the manipulator through physical interaction. Unlike conventional offline or teach-pendant-based programming, the desired trajectory is not precomputed or explicitly parameterized, but instead emerges from a continuous human–robot interaction process. From a control-theoretic standpoint, lead-through programming is therefore not a tracking problem, but an *interaction-driven trajectory generation* problem, in which externally applied forces must be interpreted as motion intent in real time.

This distinction has important consequences in an industrial context. The robot is not free-floating, nor passively gravity-compensated, but remains fully actuated and governed by an internal low-level controller at all times. As a result, the robot does not simply "follow" the operator's hands, but reacts to interaction forces through a closed-loop control stack whose internal dynamics are only partially accessible. Lead-through programming must therefore be realized as a supervisory interaction layer that coexists with proprietary motion control, safety-certified subsystems, and strict interface constraints.

Under these conditions, physical guidance can be formalized as the problem of mapping human-applied interaction wrenches into admissible robot motion commands, while simultaneously satisfying stability, usability, and safety requirements. Let $\boldsymbol{w} \in \mathbb{R}^6$ denote the wrench applied by the operator at the robot end-effector, measured by a force/torque sensor and expressed in a Cartesian frame. The role of the interaction controller is to transform $\boldsymbol{w}$ into a motion reference that reflects the operator's intent without amplifying measurement noise, inducing oscillations, or producing abrupt or unintuitive robot responses.

Among classical interaction control strategies, admittance control is particularly suited to this formulation. In an admittance-based approach, the robot is endowed with a *virtual mechanical behavior* whose apparent inertia and damping define how applied forces are converted into motion. The interaction controller thus specifies the dynamic relationship between $\boldsymbol{w}$ and the resulting motion, while the execution of this motion is delegated to the underlying robot controller. This separation is especially advantageous for industrial manipulators, as it avoids any reliance on direct torque control and can be implemented through standard motion command interfaces.

Crucially, in the context of lead-through programming, admittance control provides a natural conceptual separation between *interaction interpretation* and *motion execution*. The admittance filter defines how compliant or resistive the robot should feel to the operator, whereas the low-level controller ensures accurate tracking of the generated motion references. Provided that the tracking performance is sufficiently high, the interaction behavior can be shaped independently of the robot's internal dynamic model, control structure, or actuator technology.

However, admittance control alone does not resolve the safety problem inherent to physical human–robot interaction. In particular, a fixed interaction behavior that is comfortable and responsive during nominal operation may become unsafe when the operator moves into close proximity with the robot structure. This is especially critical for non-collaborative industrial manipulators, where physical contact cannot be assumed to be intrinsically safe and must be actively discouraged.

For this reason, the interaction problem addressed in this chapter is intrinsically coupled with a distance-modulated energy-based safety supervision mechanism. Rather than enforcing safety through discrete actions such as motion inhibition or velocity clamping, the proposed architecture integrates safety into the interaction layer by modulating the admittance parameters as a function of proximity-related

metrics. As illustrated in Fig. 4.1, safety supervision operates as an external layer that observes the interaction context and continuously reshapes the admissible interaction dynamics, while leaving the structure of the interaction controller unchanged.

This layered interpretation is fundamental for the developments that follow. Lead-through programming is treated here as a *dynamic interaction process* whose properties depend not only on the applied forces, but also on the spatial relationship between human and robot and on the ability of the control system to regulate responsiveness accordingly. The next section formalizes the admittance control law adopted in this work, establishing a precise interface through which distance-modulated energy-based safety supervision can act on the interaction dynamics.

4.2 Admittance Control Formulation for Industrial Lead-Through Programming

The interaction problem formulated in Section 4.1 requires a control law capable of converting human-applied wrenches into smooth, predictable robot motion while remaining compatible with the constraints of industrial control architectures. In this chapter, this role is fulfilled by a Cartesian admittance controller implemented as a supervisory interaction layer, which generates motion references to be tracked by the robot low-level controller.

Let $\boldsymbol{x} \in \mathbb{R}^6$ denote the Cartesian pose of the robot end-effector, expressed as the concatenation of position and orientation coordinates, and let $\dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}} \in \mathbb{R}^6$ denote the corresponding Cartesian velocity and acceleration. The interaction between the human operator and the robot is modeled through the following admittance relation:

$$\mathbf{M}_a \, \ddot{\boldsymbol{x}} + \mathbf{D}_a \, \dot{\boldsymbol{x}} = \boldsymbol{w}, \tag{4.1}$$

where $\boldsymbol{w} \in \mathbb{R}^6$ is the wrench applied by the operator at the end-effector, and $\mathbf{M}_a, \mathbf{D}_a \in \mathbb{R}^{6\times6}$ are virtual inertia and damping matrices defining the desired interaction dynamics.

Equation (4.1) describes a *virtual mechanical system* that does not correspond to the physical dynamics of the manipulator. Instead, it specifies how externally applied forces should be interpreted as motion intent at the human–robot interface. The matrices $\mathbf{M}_a$ and $\mathbf{D}_a$ are design parameters that regulate the apparent mass and dissipation perceived by the operator during lead-through programming. Lower values promote transparency and ease of guidance, whereas higher values yield a more resistive and conservative interaction behavior.

The output of the admittance model is a Cartesian motion signal, typically expressed as a velocity reference $\dot{\boldsymbol{x}}_r \in \mathbb{R}^6$, obtained by numerical integration of (4.1). This reference is subsequently mapped into joint-space motion commands $(\boldsymbol{q}_r, \dot{\boldsymbol{q}}_r, \ddot{\boldsymbol{q}}_r)$ through differential inverse kinematics and transmitted to the robot controller for

execution. At no stage does the admittance controller compute joint torques or require explicit knowledge of the robot inertial, Coriolis, or gravitational parameters.

This architectural choice is deliberate and fundamental. Although a dynamic model of the robot has been identified and validated in Chapter 3, the admittance controller formulated here is *model-independent*: its structure and operation do not depend on the availability or accuracy of such a model. The role of the admittance layer is strictly confined to shaping the interaction behavior, while the physical realization of the commanded motion is delegated to the proprietary low-level controller of the industrial manipulator.

At the same time, the identified dynamic model is not discarded. Within the robot-side control stack, it can be employed to augment motion tracking performance through feedforward compensation. In particular, joint-space feedforward torques of the form

$$\boldsymbol{\tau}_{\mathrm{ff}} = \mathbf{B}(\boldsymbol{q})\,\ddot{\boldsymbol{q}}_r + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}_r + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_f(\dot{\boldsymbol{q}}), \tag{4.2}$$

where $\boldsymbol{q}_r$ denotes the joint-space reference associated with $\dot{\boldsymbol{x}}_r$, may be combined with feedback regulation to improve tracking accuracy and transparency. Crucially, this feedforward action is confined to the low-level controller and does not alter the admittance law in (4.1). The dynamic model thus supports execution quality and physical consistency without becoming a functional dependency of the interaction controller.

In the context of this chapter, the admittance parameters $\mathbf{M}_a$ and $\mathbf{D}_a$ are not treated as fixed quantities. Instead, they are regarded as time-varying design variables whose values can be adapted online in response to safety-related constraints derived from the interaction context. The explicit formulation in (4.1) provides a well-defined and physically interpretable interface through which distance-modulated energy-based safety supervision can act on the interaction dynamics. The definition of such constraints and the corresponding parameter adaptation mechanisms are introduced in the next section.

4.3 Passivity Preservation Under Time-Varying Admittance via Energy Tanks

The safety supervision strategy introduced in this chapter relies on the online adaptation of the Cartesian admittance parameters. While constant-parameter admittance is passive under standard definiteness assumptions, time variations of the (virtual) inertia matrix may inject energy into the interaction port, compromising passivity and potentially inducing unstable or unintuitive behavior. This section therefore introduces an energy-tank mechanism that enforces passivity of the variable-admittance mapping by explicitly accounting for (i) energy dissipated by damping and (ii) energy that can be injected by inertia variations.

We denote the time-varying admittance parameters as $\mathbf{M}_a(t)$ and $\mathbf{D}_a(t)$. The admittance relation is

$$\mathbf{M}_a(t)\,\ddot{\boldsymbol{x}} + \mathbf{D}_a(t)\,\dot{\boldsymbol{x}} = \boldsymbol{w}, \tag{4.3}$$

where $(\boldsymbol{w}, \dot{\boldsymbol{x}})$ defines the interaction power port.

4.3.1 Energy Tank: State, Storage Function, and Power Balance

The energy tank is defined by a scalar state $x_t \in \mathbb{R}_{\geq 0}$ with storage function

$$T(x_t) = \frac{1}{2}x_t^2, \qquad x_t \geq 0, \tag{4.4}$$

whose time derivative is

$$\dot{T}(x_t) = \frac{\partial T}{\partial x_t}\dot{x}_t = x_t\,\dot{x}_t. \tag{4.5}$$

The tank dynamics are defined so that the tank is *replenished* by dissipated power and *depleted* by the power that may be injected by inertia variations:

$$\dot{x}_t = \frac{1}{x_t}\left(\varphi\, P_D - \gamma\, P_M\right). \tag{4.6}$$

The terms P_D and P_M are defined as

$$P_D = \dot{\boldsymbol{x}}^\top \mathbf{D}_a(t)\,\dot{\boldsymbol{x}}, \tag{4.7}$$

$$P_M = \frac{1}{2}\dot{\boldsymbol{x}}^\top \dot{\mathbf{M}}_a(t)\,\dot{\boldsymbol{x}}. \tag{4.8}$$

The scalar gates $\varphi, \gamma \in \{0, 1\}$ prevent unbounded tank growth and avoid tank depletion:

$$\varphi = \begin{cases} 1, & T(x_t) < T_{\max}, \\ 0, & T(x_t) \geq T_{\max}, \end{cases} \qquad \gamma = \begin{cases} 1, & T(x_t) > T_{\min}, \\ 0, & T(x_t) \leq T_{\min}, \end{cases} \tag{4.9}$$

with prescribed bounds $0 < T_{\min} < T_{\max}$.

4.3.2 Passivity Constraint and Admissible Inertia Variation

The interaction port is defined by the pair $(\boldsymbol{w}, \dot{\boldsymbol{x}})$ and the corresponding instantaneous power

$$P_{\text{int}} = \boldsymbol{w}^\top \dot{\boldsymbol{x}}. \tag{4.10}$$

For constant parameters, the standard storage function associated with the virtual inertia is $\frac{1}{2}\dot{\boldsymbol{x}}^\top \mathbf{M}_a(t)\,\dot{\boldsymbol{x}}$ and the damping term dissipates power. When $\mathbf{M}_a(t)$ varies with time, the term P_M in (4.8) captures the portion of power that can be injected by inertia variations. The energy-tank construction enforces passivity by requiring

that any potentially active contribution (through P_M) is "paid for" by the tank, while dissipated power P_D replenishes it (subject to $T_{\max}$).

Since $M_a(t)$ is diagonal, it is possible to obtain, as shown in [15], an upper bound for each diagonal element of the maximum admissible inertia variation that guarantees the preservation of the passivity condition, derived in discrete time assuming constant velocity over Δt:

$$\dot{\boldsymbol{m}}_i^{\text{pass}}(t) = \frac{2\omega_i(T - T_{\min})}{\dot{x}_{i,\max}^2 \Delta t} \quad \forall i = \{1, \dots, 6\}, \tag{4.11}$$

where $\dot{\boldsymbol{m}}_i^{\text{pass}}(t) \in \mathbb{R}$ represents the inertia matrix variation for the i-th diagonal element and $\dot{x}_{i,max} \in \mathbb{R}^+$ represents the velocity bound of the i-th Cartesian component, which can be estimated offline from conservative bounds on end-effector velocities under nominal lead-through conditions. Lastly, $\omega_i \in \mathbb{R}^+$ denotes the weights used to distribute the energy available in the tank across the six DOFs of robot motion, and they are designed such that:

$$\sum_{i=1}^{6} \omega_i = 1. \tag{4.12}$$

For simplicity, all the variations can be stored in a single diagonal matrix:

$$\dot{\boldsymbol{M}}_a^{\text{pass}}(t) = \text{diag}\left(\dot{\boldsymbol{m}}_1^{\text{pass}}(t), \dots, \dot{\boldsymbol{m}}_6^{\text{pass}}(t)\right). \tag{4.13}$$

Depending on the energy stored in the tank, each diagonal component of $\dot{\boldsymbol{m}}^{\text{pass}}(t)$ may reach large values. In practice, this is translated into a significant energy injection even if the system remains passive. To prevent such situations, the actual variation is defined as:

$$\dot{\mathbf{M}}_{a,\max}^{\text{pass}}(t) = \text{diag\_min}\left(\dot{\boldsymbol{M}}^{\text{pass}}(t), \dot{\boldsymbol{M}}^{U}(t)\right) \tag{4.14}$$

where $\text{diag\_min}(\cdot,\cdot)$ computes the diagonal element-wise minimum and $\dot{\boldsymbol{M}}^{U}(t) \in \mathbb{R}^{6\times 6}$ is a diagonal matrix containing, for each element, the chosen maximum threshold designed to limit excessive energy injection.

The equation

$$\dot{\mathbf{M}}_a(t) \leq \dot{\mathbf{M}}_{a,\max}^{\text{pass}}(t) \tag{4.15}$$

guarantees that the energy that could be injected by inertia variation remains compatible with the tank level (i.e., inertia can only be increased as long as the tank can supply the required energy while staying above $T_{\min}$). When $T(t) \to T_{\min}$, the admissible $\dot{\mathbf{M}}_a(t)$ tends to zero, effectively freezing inertia increases until the tank is replenished via P_D.

Interpretation in the present chapter. The energy tank does *not* provide a collision-injury guarantee and it is not a PFL mechanism; its role is strictly to preserve passivity of the interaction mapping under time-varying admittance

parameters. Architecturally, the tank acts as a passivity supervisor that sits between the safety-driven parameter adaptation law (which requests changes in $\mathbf{M}_a(t), \mathbf{D}_a(t)$) and the actual applied parameters, enforcing (4.9)–(4.15) to maintain energetic consistency during lead-through programming on an industrial robot.

4.4 Vision-Based Human State Estimation and Geometric Interface

The safety supervision mechanisms introduced in this chapter rely on the availability of geometric information describing the spatial relationship between the human operator and the robot. In the proposed architecture, this information is provided by an external vision-based perception system, which operates independently from the interaction controller and the robot low-level control stack. The role of perception is strictly delimited: it does not perform safety decisions, nor does it directly influence robot motion. Instead, it supplies a geometric abstraction of the human–robot configuration that constitutes the input to the constraint evaluation pipeline described in the next section.

This section formalizes the perception-to-supervision interface by specifying (i) how the human and robot are geometrically represented, (ii) how intentional contact inherent to hand-guiding is accounted for, and (iii) which geometric quantities are exposed to the safety supervision layer.

4.4.1 Human and Robot Geometric Abstraction

The perception system reconstructs the human operator configuration from visual data, typically in the form of a tracked skeletal model composed of keypoints and articulated segments. To enable real-time safety supervision, this representation is converted into a set of conservative geometric primitives. Let

$$\mathcal{H} = \{h_1, h_2, \ldots, h_{N_H}\}$$

denote the set of human-associated geometric primitives, where each primitive h_i is modeled as a capsule defined by a skeletal segment and an associated radius accounting for sensor noise, clothing, and modeling uncertainty.

Similarly, the robot is represented by a set of geometric primitives

$$\mathcal{R} = \{r_1, r_2, \ldots, r_{N_R}\},$$

each corresponding to a link-level volumetric approximation derived from the robot kinematic model or CAD data. The abstraction is constructed so as to conservatively bound the physical volume occupied by the robot throughout its motion, accounting for calibration tolerances and joint motion.

The use of primitive-based abstractions ensures computational efficiency and robustness, while preserving the ability to evaluate proximity relations in a geometrically meaningful way.

4.4.2 Intentional Contact and Selective Human Modeling

A defining characteristic of lead-through programming is that physical contact between the operator and the robot is intentional and continuous. The operator applies forces directly at the end-effector or at a dedicated guiding interface, and such contact is not only admissible but required for task execution. Consequently, the minimum geometric distance between the human and the robot cannot be interpreted globally as a collision-avoidance metric, since portions of the human body are expected to be in persistent contact with the robot during nominal operation.

To account for this condition, the human geometric abstraction is constructed selectively. Body segments that are directly involved in hand-guiding (such as hands and forearms) are explicitly excluded from the set $\mathcal{H}$ used for safety supervision. Including these segments would trivially lead to zero-distance conditions that are not indicative of hazardous proximity and would undermine the interpretability of distance-based metrics.

The resulting set $\mathcal{H}$ therefore represents only those human body parts whose proximity to the robot is unintended and potentially unsafe, such as the torso and head. Under this selective abstraction, geometric distance is interpreted as a measure of *unintended proximity*, rather than as an indicator of contact per se. This distinction is essential to reconcile distance-aware supervision with intentional physical interaction. A conceptual example of the human and robot geometric abstractions employed for distance computation is shown in Fig. 4.2.

4.4.3 Minimum-Distance and Closest-Point Computation

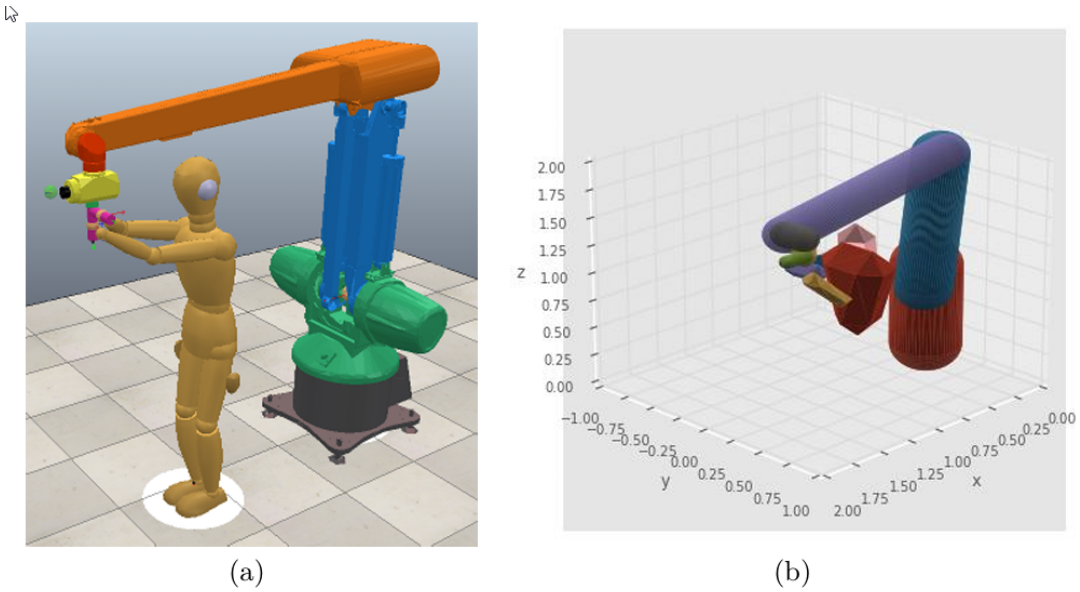
Given the sets of geometric primitives $\mathcal{H}$ and $\mathcal{R}$, the perception system evaluates the pairwise distances between all human–robot primitive pairs. For each pair $(h, r) \in \mathcal{H} \times \mathcal{R}$, a closest-point routine computes the Euclidean distance d_{hr} together with the corresponding closest points $\boldsymbol{p}_h \in \mathbb{R}^3$ and $\boldsymbol{p}_r \in \mathbb{R}^3$.

The minimum human–robot distance is then defined as

$$d = \min_{h \in \mathcal{H},\, r \in \mathcal{R}} d_{hr}, \tag{4.16}$$

and the full set of pairwise distances is collected as

$$\mathcal{D} = \{ d_{hr} \mid h \in \mathcal{H},\, r \in \mathcal{R} \}. \tag{4.17}$$



(a) (b)

Figure 4.2: Example of geometric abstraction for distance-based supervision. (a) Example of the simulation; (b) The operator is represented by a set of capsule primitives derived from the tracked skeleton, while the robot is approximated by link primitives consistent with its kinematic structure. The minimum distance d is evaluated as the smallest separation between any primitive pair $(h, r) \in \mathcal{H} \times \mathcal{R}$.

In the implemented system, distance and closest-point computations are performed using a convex distance algorithm based on the Gilbert–Johnson–Keerthi (GJK) method, which provides both the minimum distance and the associated closest-point pair in real time. The GJK routine is applied independently to each primitive pair and executed at the perception update rate.

Algorithm 1 summarizes the distance computation procedure executed at each perception update cycle.

Pairwise distances d_{hr} (and their minimum d as a diagnostic) are used to modulate the contribution of motion-related safety metrics.

In addition to scalar distances, the safety supervision layer requires the unit vector along the minimum-distance direction, defined as

$$n_{hr} = \begin{cases} \dfrac{\boldsymbol{p}_r - \boldsymbol{p}_h}{\|\boldsymbol{p}_r - \boldsymbol{p}_h\|}, & \boldsymbol{p}_r \neq \boldsymbol{p}_h, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{4.18}$$

This direction provides a physically interpretable axis along which relative motion and kinetic-energy metrics are later evaluated.

Algorithm 1 Minimum human–robot distance computation (pairwise GJK)

Inputs: $\mathcal{H}$, $\mathcal{R}$
Outputs: $\mathcal{D}$ and (optionally) closest points for each pair
$\mathcal{D} \leftarrow \emptyset$
for all $h \in \mathcal{H}$ **do**
    for all $r \in \mathcal{R}$ **do**
        $(d_{hr}, \boldsymbol{p}_h, \boldsymbol{p}_r) \leftarrow \text{GJKdistance}(h, r)$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{d_{hr}\}$
    end for
end for
$d \leftarrow \min(\mathcal{D})$

4.4.4 Perception-to-Supervision Interface

The perception system delivers to the safety supervision layer a limited and well-defined set of geometric quantities:

- the sets of geometric primitives $\mathcal{H}$ and $\mathcal{R}$,
- the pairwise distances d_{hr} and the minimum distance d,
- the closest-point pairs $(\boldsymbol{p}_h, \boldsymbol{p}_r)$ associated with each d_{hr},
- the unit vectors n_{hr} along the minimum-distance directions.

These quantities constitute the complete interface between perception and safety supervision. Importantly, the perception system does not perform any thresholding, scaling, or decision-making related to safety. All safety-relevant logic (such as constraint evaluation, distance modulation, and parameter adaptation) is implemented deterministically within the supervision layer described in the next section.

4.4.5 Uncertainty and Conservativeness

Vision-based human state estimation is inherently subject to uncertainty arising from sensor noise, occlusions, and limited field of view. Rather than attempting to eliminate these uncertainties through complex estimation techniques, the proposed architecture addresses them structurally through conservative modeling choices. Capsule radii are selected to upper-bound plausible human body occupancy, and robot primitives are inflated to account for calibration and kinematic uncertainties.

As a result, distance estimates produced by the perception system tend to underestimate, rather than overestimate, the true separation between human and robot. This conservative bias is appropriate for safety supervision, as it leads to earlier activation of constraint-based adaptations without relying on hard distance

thresholds. Crucially, perception uncertainty affects the interaction behavior only through explicit geometric quantities and does not introduce hidden couplings into the interaction controller.

This completes the definition of the geometric interface required by the safety supervision layer. Building on these inputs, the next section introduces the distance- and energy-based safety metrics and formalizes the constraint evaluation and parameter adaptation mechanisms used to regulate the interaction dynamics.

4.5 Safety Metrics and Constraint-Based Admittance Parameter Adaptation

This section formalizes the safety supervision layer by defining the safety metrics and constraints used to modulate the admittance parameters during lead-through programming. The adopted strategy relies on a single, physically grounded constraint based on relative kinetic energy, whose influence is continuously modulated by geometric proximity. Distance is not treated as an independent or competing safety constraint, but rather as a contextual variable that regulates the severity of motion-related risk as unintended human–robot proximity increases.

4.5.1 Relative Kinetic-Energy Constraint with Distance Modulation

Geometric separation alone is insufficient to characterize risk during lead-through programming, particularly in the presence of intentional physical contact. Conversely, motion-related quantities such as kinetic energy lack contextual awareness if evaluated independently of proximity. For this reason, the supervision strategy adopted in this chapter employs a single kinetic-energy–based constraint, whose influence is continuously modulated by distance.

The kinetic-energy metric is evaluated along the closest-point direction and employs the identified robot dynamics through the inertia matrix $\mathbf{B}(\boldsymbol{q})$. For each pair $(h, r) \in \mathcal{H} \times \mathcal{R}$, let $n_{hr} \in \mathbb{R}^3$ be the unit vector directed along the closest-point direction at the closest points. Let $\mathbf{J}_r^p(\boldsymbol{q}) \in \mathbb{R}^{3\times 6}$ denote the translational Jacobian associated with the robot primitive r.

An equivalent mass along n_{hr} is computed as

$$m_{hr} = \left(n_{hr}^\top \mathbf{J}_r^p(\boldsymbol{q})\, \mathbf{B}^{-1}(\boldsymbol{q})\, \mathbf{J}_r^p(\boldsymbol{q})^\top n_{hr}\right)^{-1}, \tag{4.19}$$

and the corresponding relative velocity component is

$$v_{hr} = n_{hr}^\top \mathbf{J}_r^p(\boldsymbol{q})\, \dot{\boldsymbol{q}}. \tag{4.20}$$

Algorithm 2 Kinetic-energy constraint evaluation (distance-modulated, model-based)

Inputs: $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\mathcal{H}$, $\mathcal{R}$, $\boldsymbol{p}_h$, $\boldsymbol{p}_r$
Parameters: k_1, k_2, $C_{\max}$
Outputs: $\mathcal{SC}_c$
$\mathcal{SC}_c \leftarrow \emptyset$
for all $h \in \mathcal{H}$ **do**
    for all $r \in \mathcal{R}$ **do**
        Compute d_{hr} and direction n_{hr}
        $m_{hr} \leftarrow \left(n_{hr}^{\top} \mathbf{J}_r^p \, \mathbf{B}^{-1} \, \mathbf{J}_r^{p\top} \, n_{hr}\right)^{-1}$
        $v_{hr} \leftarrow n_{hr}^{\top} \mathbf{J}_r^p \, \dot{\boldsymbol{q}}$
        $C_{hr} \leftarrow \frac{1}{2} m_{hr} v_{hr}^2$
        $k \leftarrow 1 - \tanh(k_1 d_{hr} + k_2)$
        $SC_{c,hr} \leftarrow C_{\max} - k \cdot C_{hr}$
        $\mathcal{SC}_c \leftarrow \mathcal{SC}_c \cup \{SC_{c,hr}\}$
    end for
end for

The relative kinetic energy is then defined as

$$C_{hr} = \frac{1}{2} m_{hr} v_{hr}^2. \tag{4.21}$$

Distance is used to regulate the influence of kinetic energy through a smooth, monotonic modulation function

$$k(d_{hr}) = 1 - \tanh(k_1 d_{hr} + k_2)\,, \tag{4.22}$$

where $k_1, k_2 \in \mathbb{R}$ are shaping parameters. The function $k(d_{hr})$ is designed such that its contribution is negligible at large distances and increases smoothly as d_{hr} decreases, ensuring that kinetic-energy–based supervision is progressively activated only in close-proximity conditions.

Let $C_{\max} \in \mathbb{R}_{>0}$ denote an admissible bound for the distance-modulated relative kinetic energy. The corresponding constraint is defined as

$$SC_{c,hr} = C_{\max} - k(d_{hr})\, C_{hr}, \tag{4.23}$$

and the kinetic-energy constraint set is

$$\mathcal{SC}_c = \{SC_{c,hr} \mid h \in \mathcal{H},\, r \in \mathcal{R}\}\,. \tag{4.24}$$

Algorithm 2 summarizes the constraint computation.

4.5.2 Worst-Case Extraction, Scaling, and Parameter Update

The kinetic-energy constraint set $\mathcal{SC}_c$ is reduced to a worst-case value by selecting its minimum element,

$$SC_c^{\min} = \min_{h\in\mathcal{H},\, r\in\mathcal{R}} SC_{c,hr}. \tag{4.25}$$

which represents the most critical human–robot configuration at the current time. The worst-case constraint $SC_c^{\min}$ corresponds to the critical pair

$$(h^\star, r^\star) = \arg\min_{h\in\mathcal{H},\, r\in\mathcal{R}} SC_{c,hr}. \tag{4.26}$$

This value is saturated within prescribed bounds $[SC_c^l, SC_c^U]$ and mapped into a normalized scaling coefficient

$$\alpha = \frac{SC_c^{\min} - SC_c^l}{SC_c^U - SC_c^l}, \tag{4.27}$$

ensuring $\alpha \in [0, 1]$.

The scaling coefficient α does not directly define a velocity limit or a stopping action. Instead, it parameterizes a continuous reshaping of the interaction dynamics by interpolating the admittance matrices between two empirically selected operating regimes. Specifically, $\mathbf{M}_a^{\min}, \mathbf{D}_a^{\min} \in \mathbb{R}^{6\times6}$ denote the *nominal* admittance parameters tuned to achieve high transparency and comfortable hand-guiding during normal lead-through operation, while $\mathbf{M}_a^{\max}, \mathbf{D}_a^{\max} \in \mathbb{R}^{6\times6}$ denote a *conservative* parameter set corresponding to a heavily damped, high apparent inertia behavior intended to reduce the robot responsiveness in critical conditions.

These bounds were selected empirically through iterative tuning on the target industrial platform. The tuning process aimed to satisfy three practical requirements: (i) the nominal set $(\mathbf{M}_a^{\min}, \mathbf{D}_a^{\min})$ should yield a predictable and low-effort guidance feel without oscillations under typical operator inputs; (ii) the conservative set $(\mathbf{M}_a^{\max}, \mathbf{D}_a^{\max})$ should significantly attenuate the motion response to the same inputs while preserving continuity of interaction (i.e., avoiding abrupt "freezing" or stick–slip effects perceived by the operator); and (iii) intermediate values produced by interpolation should remain stable and well-conditioned when executed through the industrial low-level motion controller. In the implementation, these matrices were chosen diagonal to decouple translational and rotational axes and to simplify empirical tuning, although the framework does not require this restriction.

$$\mathbf{M}_a = \mathbf{M}_a^{\min} + \alpha\left(\mathbf{M}_a^{\max} - \mathbf{M}_a^{\min}\right), \qquad \mathbf{D}_a = \mathbf{D}_a^{\min} + \alpha\left(\mathbf{D}_a^{\max} - \mathbf{D}_a^{\min}\right). \tag{4.28}$$

As α increases, corresponding to increasing distance-modulated kinetic energy, the admittance parameters are driven toward conservative values, reducing the

Algorithm 3 Constraint-based scaling and admittance parameter update

Inputs: $\mathcal{SC}_c$
Parameters: SC_c^U, SC_c^l, bounds on $\mathbf{M}_a, \mathbf{D}_a$
$SC_c^{\min} \leftarrow \min(\mathcal{SC}_c)$
Saturate $SC_c^{\min}$ in $[SC_c^l, SC_c^U]$
$\alpha \leftarrow \dfrac{SC_c^{\min} - SC_c^l}{SC_c^U - SC_c^l}$
$\mathbf{M}_a \leftarrow \mathbf{M}_a^{\min} + \alpha(\mathbf{M}_a^{\max} - \mathbf{M}_a^{\min})$
$\mathbf{D}_a \leftarrow \mathbf{D}_a^{\min} + \alpha(\mathbf{D}_a^{\max} - \mathbf{D}_a^{\min})$

robot's responsiveness to externally applied wrenches. Conversely, when the kinetic-energy constraint is inactive, α approaches zero and the interaction behavior returns smoothly to its nominal, highly transparent configuration.

Algorithm 3 summarizes the complete scaling and parameter update procedure.

This strategy yields a continuous, bounded, and physically interpretable adaptation mechanism. The dynamic model affects supervision only through the computation of the equivalent mass in (4.19), providing a principled measure of motion-related risk without introducing a functional dependency between the interaction control law and inverse dynamics.

4.6 Experimental Validation of Constraint-Based Lead-Through Programming

This section presents the experimental validation of the proposed lead-through programming framework, with the objective of assessing the practical impact of the constraint-based safety supervision strategy introduced in Sections 4.4 and 4.5. The validation does not aim to demonstrate certified safety compliance, nor to benchmark optimal performance; rather, it seeks to verify that the interaction dynamics, safety metrics, and parameter adaptation mechanisms behave coherently when deployed on a real industrial platform and subjected to representative human guidance actions.

In particular, the validation focuses on three aspects: (i) the evolution of energy-based safety constraints during physical interaction, (ii) the resulting adaptation of admittance parameters, and (iii) the qualitative interaction behavior perceived by the operator during lead-through programming.

4.6.1 Experimental Setup and System Configuration

The experimental setup consists of an industrial manipulator equipped with an end-effector force/torque sensor and operating within a robot cell instrumented with

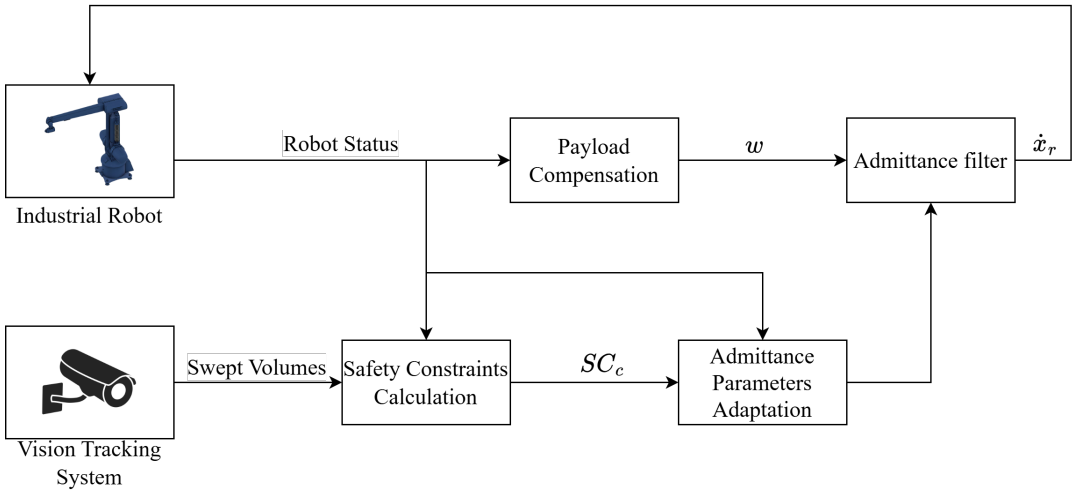


Figure 4.3: Control architecture for admittance-based lead-through programming under constraint-based safety supervision. The perception system provides geometric abstractions to the safety layer, which adapts admittance parameters without directly commanding robot motion.

Table 4.1: Main parameters used in experimental validation.

Category	Parameter	Value
Admittance	$\mathbf{M}_a^{\min}$	{15.0,15.0,15.0,1.5,1.5,1.5}
Admittance	$\mathbf{M}_a^{\max}$	{30.0,30.0,30.0,3.0,3.0,3.0}
Admittance	$\mathbf{D}_a^{\min}$	{60.0,60.0,60.0,6.0,6.0,6.0}
Admittance	$\mathbf{D}_a^{\max}$	{120.0,120.0,120.0,12.0,12.0,12.0}
Distance modulation	k_1, k_2	8.0, -3.5

an external vision system for human state estimation. The admittance controller runs as a supervisory module and generates Cartesian velocity references, which are tracked by the robot's internal low-level controller. The safety supervision layer operates in parallel, receiving geometric information from the perception module and dynamically adapting the admittance parameters according to the constraint evaluation pipeline described in Section 4.5.

A schematic overview of the control architecture is shown in Fig. 4.3. The figure highlights the separation between perception, safety supervision, interaction control, and low-level motion execution, which is a central design principle of the proposed framework.

Key parameters of the experimental configuration, including nominal and conservative admittance bounds and distance-modulation parameters, are summarized in Table 4.1. This table fixes the operating conditions under which the results presented in the following subsections were obtained.

4.6.2 Validation Protocol

The validation protocol is designed to elicit a wide spectrum of interaction conditions while remaining representative of realistic lead-through programming scenarios. During each trial, the operator physically guides the robot along predefined paths within the workspace, combining straight segments and curved motions, while naturally varying both proximity to the robot structure and interaction intensity. No explicit constraints are imposed on speed or distance, so as to preserve an intuitive and unconstrained interaction style and to allow safety-relevant conditions to emerge organically.

Each experimental run logs, in a time-synchronized manner, the quantities required to reconstruct the full perception–supervision–interaction pipeline. In particular, the following signals are recorded: (i) the adapted admittance parameters (scalar summaries of inertia and damping), (ii) the global scaling coefficient α resulting from constraint evaluation, (iii) the distance-modulated relative kinetic energy $k(d_{hr})\, C_{hr}$ associated with the critical human–robot primitive pair, and (iv) the minimum human–robot distance d, reported as a global proximity diagnostic.

Representative time histories of these quantities are reported in Fig. 4.4. The figure is deliberately organized to reflect the causal structure of the proposed framework rather than to provide a statistical characterization. From top to bottom, it illustrates how changes in the interaction context and proximity give rise to variations in the distance-modulated kinetic-energy metric, how these variations activate the safety supervision through the scaling coefficient α, and how the resulting supervisory action manifests as continuous adaptation of the admittance parameters. The minimum distance d is included to contextualize the interaction but does not directly drive the adaptation.

4.6.3 Behavior of Distance Modulation and Energy Constraints

The experimental results confirm that when the operator remains at moderate distances, the modulation function $k(d_{hr})$ significantly attenuates the contribution of kinetic energy, resulting in negligible constraint activation even in the presence of nonzero robot velocity. In these conditions, the admittance parameters remain close to their nominal values and the robot exhibits high transparency.

As the operator approaches the robot, the reduction in d_{hr} increases the gain $k(d_{hr})$, thereby amplifying the contribution of relative kinetic energy to the constraint evaluation. This mechanism causes the energy-based constraint $SC_c^{\min}$ to become active earlier than a simple distance-based threshold strategy would, effectively penalizing faster motions in close-proximity conditions without enforcing abrupt motion inhibition. This behavior is consistent across repeated trials and different operator interaction styles.

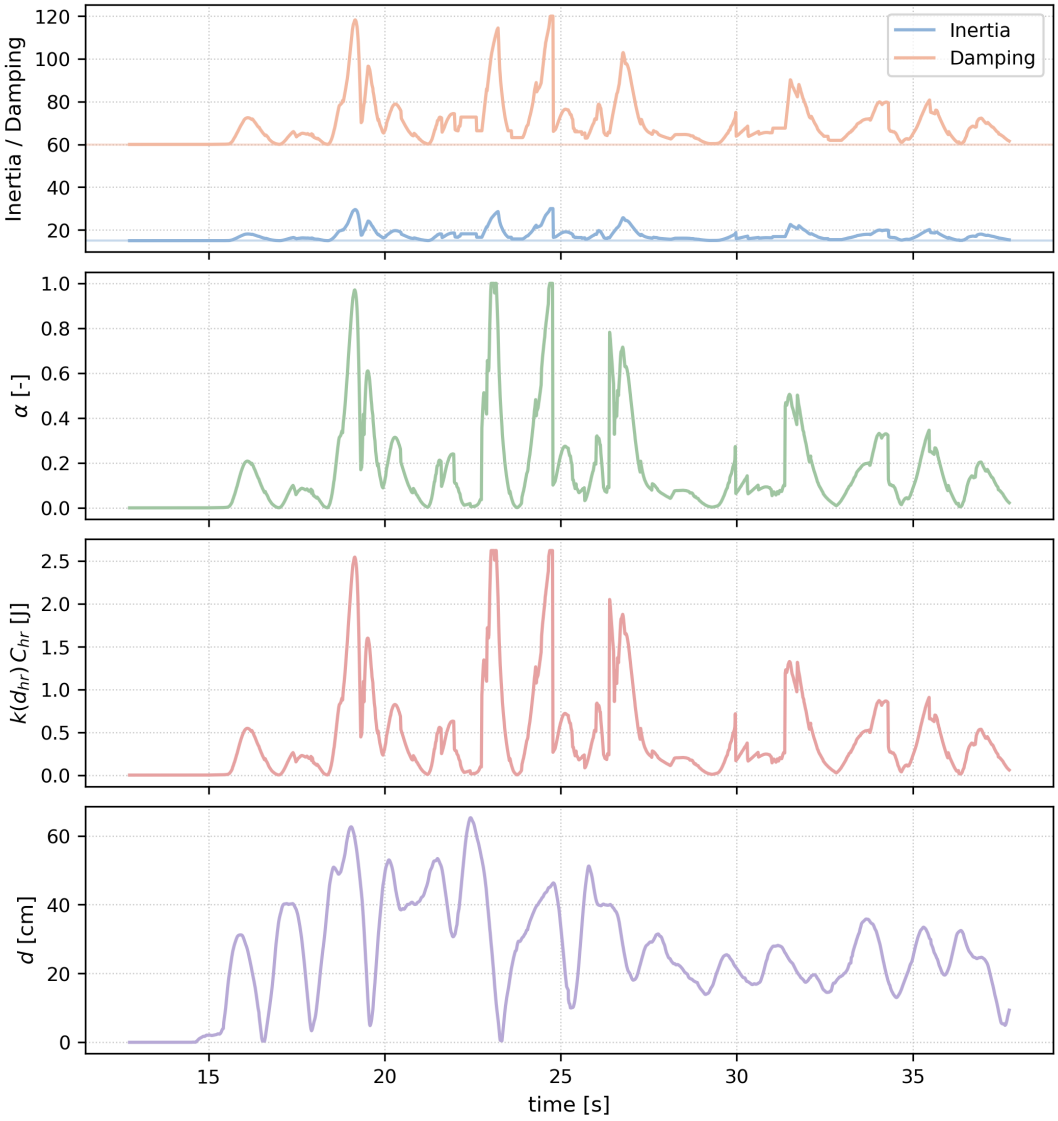


Figure 4.4: Representative time histories during lead-through programming, illustrating the causal chain from safety supervision to interaction behavior: adapted admittance parameters (inertia and damping), resulting scaling coefficient α, distance-modulated relative kinetic energy $k(d_{hr})\, C_{hr}$ for the critical human–robot pair, and minimum human–robot distance d (global diagnostic).

Table 4.2: Qualitative interaction behavior observed during experimental validation.

Condition	Constraint activation	Perceived behavior
Large distance, low speed	None	High transparency
Moderate distance, higher speed	Energy-modulated	Increased damping
Close proximity	Strong modulation	Resistive interaction

Importantly, the experimental data corroborate the design choice of using distance to modulate the kinetic-energy constraint rather than directly competing with it. In preliminary configurations (not reported here), a direct worst-case selection between distance- and energy-based constraints led to overly conservative behavior, with persistent parameter adaptation even at low interaction speeds. The adopted modulation strategy mitigates this effect while preserving sensitivity to both proximity and motion intensity.

While the minimum distance $d = \min_{h \in \mathcal{H},\, r \in \mathcal{R}} d_{hr}$ is reported as a global proximity diagnostic, the supervision acts on the critical primitive pair $(h^\star, r^\star)$ defined in (4.26), which is not necessarily the same pair achieving the minimum distance.

4.6.4 Admittance Parameter Adaptation and Interaction Feel

The evolution of the scaling coefficient α and the corresponding admittance parameters demonstrates that the adaptation mechanism operates smoothly and without discontinuities. As shown in Fig. 4.4, increases in α correlate with reductions in minimum distance and increases in distance-modulated kinetic energy, resulting in higher apparent inertia and damping.

From the operator's perspective, this manifests as a progressive increase in resistance and damping when approaching the robot or inducing faster motions, rather than as sudden stops or unpredictable behavior. No oscillatory or unstable interaction was observed during the experiments, even in conditions of repeated approach and withdrawal, indicating that the parameter adaptation preserves stability of the overall interaction loop.

A qualitative summary of observed interaction behaviors under different proximity and motion conditions is reported in Table 4.2. This table emphasizes the experiential aspect of lead-through programming, which is central to the usability of the proposed approach.

4.6.5 Discussion of Experimental Results

The experimental validation supports the central claim of this chapter: that safe and intuitive lead-through programming can be achieved on industrial robots by embedding safety supervision into the interaction dynamics through constraint-based admittance parameter adaptation. The combined use of geometric distance and model-based kinetic energy provides a physically interpretable and practically effective means of anticipating unsafe conditions without relying on contact-based safety paradigms or discrete stopping actions.

At the same time, the results highlight the importance of careful constraint design and tuning. The role of distance as a modulation factor, rather than as a competing constraint, emerges as a key insight for maintaining usability while enforcing conservative behavior in proximity. These findings motivate the transition, in the next chapter, toward safety paradigms that explicitly incorporate energy exchange and passivity guarantees in collaborative robotic systems.

4.7 Discussion, Scope, and Limitations

This chapter has addressed the problem of safe and intuitive lead-through programming of an industrial manipulator through physical human–robot interaction, under the explicit assumption that intentional contact between the operator and the robot is intrinsic to the task. Within this setting, safety cannot be enforced through contact avoidance alone, nor through paradigms that presuppose intrinsically collaborative hardware. Instead, safety has been framed as an emergent property of the interaction dynamics, shaped by external supervision and conservative modulation of the robot's response to human-applied forces.

4.7.1 Interpretation of the Safety Paradigm

A first point that merits clarification concerns the nature of the adopted safety paradigm. The supervision strategy developed in this chapter is *distance-aware* and *model-informed*, but it is neither a certified implementation of Speed and Separation Monitoring nor a Power and Force Limiting scheme. Distance information is used to characterize unintended proximity between the robot and non-interacting parts of the operator's body, while intentional contact at the hand-guiding interface is explicitly allowed and excluded from distance computation. As a result, the minimum distance d should not be interpreted as a collision-avoidance margin in the classical sense, but as a contextual variable that modulates the admissible interaction dynamics.

Similarly, the kinetic-energy metric introduced in Section 4.5 does not constitute an injury criterion. It provides a physically grounded measure of motion-related risk, derived from the identified robot inertia matrix, and is employed solely to

constrain the aggressiveness of the interaction response. No claim is made regarding biomechanical injury thresholds, pain limits, or post-impact safety. This distinction is essential to avoid conflating the present framework with collaborative safety standards that rely on fundamentally different assumptions.

4.7.2 Distance Modulation versus Distance Domination

An important insight emerging from the experimental validation concerns the interaction between distance- and energy-based constraints. While an initial worst-case combination of the two constraints is conceptually appealing, it was found to be overly conservative in practice, leading to persistent parameter adaptation even in benign interaction conditions. The adopted strategy (using distance to modulate the contribution of kinetic energy rather than to compete with it directly) represents a pragmatic compromise between sensitivity and usability.

This design choice highlights a broader point: in hand-guiding tasks, distance alone is an insufficient proxy for risk, especially when intentional contact is present. Conversely, energy-based metrics alone may fail to capture contextual aspects of proximity. The combined, hierarchical use of these metrics reflects the structure of the task and the industrial constraints under which it is performed.

4.7.3 Architectural Separation and Industrial Compatibility

Throughout the chapter, a strict separation has been maintained between perception, safety supervision, interaction control, and low-level motion execution. This architectural discipline is not merely an implementation detail; it underpins the interpretability and extensibility of the proposed framework. Vision-based perception provides geometric information without issuing control commands. Safety supervision evaluates explicit, inspectable constraints and modulates interaction parameters. The admittance controller interprets human intent through a virtual mechanical model, while the industrial robot controller remains responsible for motion tracking and low-level stability.

Such separation is particularly relevant in industrial contexts, where modifications to certified control layers are often infeasible. By confining safety-related adaptations to a supervisory layer and avoiding direct intervention on motion commands, the framework remains compatible with existing industrial robots while enabling richer forms of physical programming.

4.7.4 Limitations and Outlook

Despite its effectiveness in the considered scenario, the proposed approach has clear limitations. It does not provide formal safety guarantees in the sense of ISO-certified SSM or PFL, nor does it address post-contact impact mitigation. Its validity relies on conservative geometric modeling, appropriate parameter tuning, and the reliability of the perception system. Moreover, the focus on hand-guiding tasks limits its applicability to scenarios where intentional contact is localized and well understood.

These limitations are not shortcomings of the implementation, but rather reflections of the chosen scope. The framework is designed to support safe and intuitive programming of industrial robots through physical interaction, not to replace collaborative safety standards or to enable unrestricted human–robot coexistence.

Crucially, the insights gained in this chapter motivate the transition to more expressive safety paradigms. In particular, scenarios involving sustained close proximity, shared manipulation, or contact-rich interaction require explicit reasoning about energy exchange and passivity. The next chapter builds on the experience developed here by moving toward energy-based safety strategies and Power and Force Limiting concepts in collaborative robotic systems, where the boundaries between interaction control and safety are inherently different.

Chapter 5

Energy-Based Variable Admittance Control for Collaborative Robotics

The adaptive admittance strategies developed in Chapter 4 were conceived within an industrial context in which physical interaction between human and robot is allowed only under strict supervisory conditions. In that setting, safety is enforced externally through distance-aware monitoring mechanisms inspired by Speed and Separation Monitoring (SSM), while the admittance controller serves exclusively as an interaction interface for intuitive trajectory teaching. Contact is neither intended nor tolerated, and any proximity-related risk is mitigated by modulating the robot motion as a function of spatial separation. This paradigm is well aligned with the constraints of non-collaborative industrial manipulators and with existing industrial safety practices governed by ISO 10218 [53, 54].

Collaborative robotics introduces a fundamentally different interaction regime. In Power and Force Limiting (PFL) scenarios, physical contact between the human and the robot is not only admissible but explicitly anticipated as part of normal operation. Safety is no longer achieved by preventing contact through spatial separation; instead, it is ensured by limiting the mechanical energy, forces, and pressures exchanged during interaction in accordance with human tolerance thresholds defined by ISO/TS 15066 [50]. As a consequence, safety can no longer be treated as an external supervisory layer acting on robot motion. Rather, it becomes an intrinsic property of the interaction dynamics themselves.

This shift has profound implications for the role of admittance control. Under SSM-inspired supervision, admittance parameters primarily affect interaction transparency and operator effort, while safety is guaranteed by motion scaling or interruption triggered by distance-based criteria. In contrast, under PFL, the admittance parameters directly shape the robot's apparent mechanical behavior during contact, thereby influencing the amount of kinetic energy that can be transferred to the human. In this context, inertia and damping are no longer neutral tuning parameters but safety-critical quantities whose values must be continuously compatible

with the admissible interaction energy.

The present chapter addresses this conceptual transition explicitly. Rather than extending the distance-modulated framework of Chapter 4, it replaces the underlying safety paradigm with an energy-based formulation rooted in PFL. The admittance controller remains the central interaction interface, but its adaptive mechanisms are redesigned so that safety constraints are enforced intrinsically through energy reasoning, rather than extrinsically through distance supervision.

By reframing safety as an energetic feasibility problem, this chapter establishes the foundation for a unified treatment of stability, interaction quality, and safety within the admittance control framework. The following sections formalize this problem by deriving the PFL energy constraint, exposing the coupling between admittance parameters and apparent mass, and introducing two distinct solution paradigms (algorithmic and optimization-based) for adapting admittance parameters under explicit safety constraints.

5.1 Power and Force Limiting as an Energy Constraint

Power and Force Limiting (PFL) defines safety in collaborative robotics by bounding the mechanical interaction between human and robot to levels that are physiologically tolerable. Unlike distance-based paradigms, PFL explicitly admits physical contact and regulates its consequences by constraining the forces, pressures, and energies exchanged during interaction. Among these quantities, energy provides a compact and physically grounded abstraction that captures, in a single scalar inequality, the coupled effects of relative motion and effective inertial properties. For this reason, energy-based formulations of PFL are widely adopted in physical human–robot interaction (pHRI) [1, 4] and are formalized in ISO/TS 15066 [50].

5.1.1 Transferred-energy bound in ISO/TS 15066

A central quantity in PFL is the energy transferred during a potential inelastic collision between the robot and a human body part. Under commonly adopted impact assumptions, the kinetic energy dissipated during such an event can be expressed with the scalar $\Delta K_e \in \mathbb{R}_{\geq 0}$:

$$\Delta K_e = \frac{1}{2} \frac{m_r m_h}{m_r + m_h} \left\| \dot{x}_{Rn} + \dot{x}_{Hn} \right\|^2, \tag{5.1}$$

where $m_r \in \mathbb{R}_{>0}$ denotes the robot apparent mass along the direction of impact, $m_h \in \mathbb{R}_{>0}$ is the equivalent mass of the involved human body part, and $\dot{x}_{Rn} \in \mathbb{R}$ and $\dot{x}_{Hn} \in \mathbb{R}$ are the signed projected velocities along $\mathbf{n}$, defined so that $\dot{x}_{Rn} + \dot{x}_{Hn}$ equals the relative closing speed (see Section 5.1.2).

The PFL requirement imposes the inequality

$$\Delta K_e \leq E_{\max}^{\mathrm{PFL}}, \tag{5.2}$$

where $E_{\max}^{\mathrm{PFL}} \in \mathbb{R}_{>0}$ is the maximum admissible transferred energy specified by ISO/TS 15066 for a given body region and contact scenario. In particular, $E_{\max}^{\mathrm{PFL}}$ can be computed from the maximum allowable contact force $F_{\max} \in \mathbb{R}_{>0}$ and the effective stiffness $k \in \mathbb{R}_{>0}$ of the considered body part:

$$E_{\max}^{\mathrm{PFL}} = \frac{F_{\max}^2}{2k} = \frac{A^2 p_{\max}^2}{2k}, \tag{5.3}$$

where $A \in \mathbb{R}_{>0}$ is the contact area and $p_{\max} \in \mathbb{R}_{>0}$ is the maximum allowable pressure. These parameters depend on anatomical and contextual factors and are tabulated in ISO/TS 15066 [50].

5.1.2 Projected motion quantities and directionality

The quantities in (5.1) depend on the relative motion of the robot and the human along the direction connecting them. Let $\mathbf{n} \in \mathbb{R}^6$ denote the unit approach direction (with nonzero translational components) pointing from the robot end-effector toward the human body part of interest. The robot Cartesian velocity projected along this direction is defined as:

$$\dot{x}_{Rn} = \mathbf{n}^\top \mathbf{J}(\boldsymbol{q})\, \dot{\boldsymbol{q}}, \tag{5.4}$$

where $\mathbf{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times 6}$ denotes the geometric Jacobian at configuration $\boldsymbol{q} \in \mathbb{R}^6$ and $\mathbf{n} = \{n_x, n_y, n_z, 0, 0, 0\}$.

Sign convention for projected velocities. In this chapter, projected human and robot velocities are expressed in a *closing-speed* convention. Specifically, the human projected velocity is defined with the opposite orientation of the robot projection,

$$\dot{x}_{Hn} \triangleq -\,\mathbf{n}^\top \dot{\boldsymbol{x}}_H, \tag{5.5}$$

so that the scalar sum $\dot{x}_{Rn} + \dot{x}_{Hn}$ represents the *relative closing speed* along the approach direction $\mathbf{n}$. Under this convention, $\dot{x}_{Rn} + \dot{x}_{Hn} \geq 0$ corresponds to approaching motion (increasing collision likelihood), whereas $\dot{x}_{Rn} + \dot{x}_{Hn} < 0$ corresponds to separating motion. $\dot{\boldsymbol{x}}_H \in \mathbb{R}^6$ denotes the estimated human body-part Cartesian velocity expressed in the same frame as $\mathbf{n}$. This choice is adopted to keep all feasibility expressions dependent on a single signed closing-speed quantity.

A key aspect for online enforcement is that the energy-based condition is most relevant when the relative motion tends to increase collision danger, i.e., when the robot moves toward the human along $\mathbf{n}$. This is captured through a directional sign variable $s \in \{-1, +1\}$:

$$s = \begin{cases} +1, & \dot{x}_{Rn} + \dot{x}_{Hn} \geq 0, \\ -1, & \dot{x}_{Rn} + \dot{x}_{Hn} < 0. \end{cases} \tag{5.6}$$

This choice encodes the fact that the safety condition is activated when motion is approaching and naturally relaxed when motion is separating, thereby avoiding unnecessary conservatism during disengagement.

5.1.3 Continuous safety residual and feasibility condition

While (5.2) must be satisfied at the instant of collision, practical controllers benefit from continuously monitoring proximity to the safety boundary during motion. By rearranging (5.1)–(5.2), one can define the scalar PFL residual $h \in \mathbb{R}$:

$$h = s\, m_r m_h \left\| \dot{x}_{Rn} + \dot{x}_{Hn} \right\|^2 - 2E_{\max}^{\mathrm{PFL}}(m_h + m_r), \tag{5.7}$$

which satisfies $h \leq 0$ whenever the admissible transferred-energy condition is respected. The residual (5.7) is not merely a diagnostic quantity: it explicitly reveals that PFL feasibility depends jointly on *relative projected velocity* and *apparent mass*. As a consequence, ensuring safety is inherently a *control-parameter selection problem*: the controller can reduce h either by decreasing the projected velocity term or by reducing the apparent mass term, or by coordinating both.

This dissertation exploits this structure directly. In particular, since the apparent mass term m_r is shaped by the admittance inertia (Section 5.2), and the projected velocity evolution depends on the admittance damping through the admittance dynamics (Sections 5.3–5.4), the PFL condition can be enforced through online adaptation of both $\mathbf{M}_a$ and $\mathbf{D}_a$, rather than by imposing external speed limits. This sets the stage for the two adaptation paradigms developed in the remainder of the chapter: an algorithmic strategy and an optimization-based strategy, both operating on the same feasibility residual (5.7).

5.2 Apparent Mass and Safety Coupling in Admittance Control

The PFL formulation introduced in Section 5.1 expresses safety as a feasibility condition that depends explicitly on two quantities: the apparent mass of the robot along the direction of interaction and the relative velocity between robot and human. In order to actively enforce this condition through control, it is therefore necessary to make explicit how both quantities arise within an admittance-controlled interaction and how they are shaped by the admittance parameters.

As established in Chapter 4 (Section 4.2), physical human–robot interaction in this work is mediated through a Cartesian admittance control law, which defines a virtual mechanical relationship between the interaction wrench applied by the human and the resulting end-effector motion. The detailed formulation of the admittance dynamics has already been introduced and will not be repeated here. Instead, this

section focuses on a specific and safety-critical consequence of that formulation: the fact that the admittance parameters directly determine the apparent mass and the velocity evolution entering the PFL constraint.

5.2.1 Apparent mass induced by admittance inertia

When interaction occurs predominantly along a specific direction, the full Cartesian admittance can be projected onto that direction to obtain a scalar representation of the interaction dynamics. Let $\mathbf{n} \in \mathbb{R}^6$ denote the unit vector defining the direction of potential impact introduced in Section 5.1. The apparent robot mass along the potential impact direction $\mathbf{n}$ is given by

$$m_r = \mathbf{n}^\top \mathbf{M}_a \mathbf{n}, \tag{5.8}$$

where $m_r \in \mathbb{R}_{>0}$ coincides with the apparent mass term appearing in the PFL energy formulation. Equation (5.8) makes explicit a fundamental property of admittance-controlled interaction: the apparent mass governing impact energy is not an intrinsic property of the robot hardware, but a control-dependent quantity shaped by the chosen admittance inertia matrix $\mathbf{M}_a$.

Substituting (5.8) into the PFL energy expression (5.1) reveals that, for a given relative velocity along the potential impact direction $\mathbf{n}$, the transferred energy scales linearly with the projected admittance inertia. As a consequence, increasing $\mathbf{M}_a$ increases the energy that may be transferred during contact. Conversely, reducing $\mathbf{M}_a$ lowers the apparent mass and enlarges the admissible safety margin, at the cost of a lighter and potentially less stable interaction behavior.

5.2.2 Velocity evolution under admittance damping

The second term entering the PFL constraint is the relative velocity projected along the interaction direction. In admittance control, the evolution of the end-effector velocity is governed by the admittance dynamics and is therefore directly influenced by the damping matrix $\mathbf{D}_a$. Projecting the translational component of the admittance equation along $\mathbf{n}$ yields the scalar dynamics

$$m_r \, \ddot{x}_{Rn} + d_r \, \dot{x}_{Rn} = w_n, \tag{5.9}$$

where $\dot{x}_{Rn} \in \mathbb{R}$ and $\ddot{x}_{Rn} \in \mathbb{R}$ denote the projected velocity and acceleration, $w_n \in \mathbb{R}$ is the projected interaction force, and

$$d_r = \mathbf{n}^\top \mathbf{D}_a \mathbf{n}, \qquad d_r \in \mathbb{R}_{>0} \tag{5.10}$$

is the apparent damping along the interaction direction.

Equation (5.9) shows that the projected velocity entering the PFL residual is not an independent quantity, but the outcome of a dynamic process shaped jointly by

apparent mass and apparent damping. Increasing d_r dissipates energy and reduces $\dot{x}_{Rn}$ more rapidly, whereas decreasing d_r improves responsiveness but allows higher velocities to persist. As a result, both $\mathbf{M}_a$ and $\mathbf{D}_a$ directly influence the evolution of the safety residual (5.7).

5.2.3 Implications for safety-oriented parameter adaptation

The expressions above clarify why safety under PFL cannot be enforced by velocity regulation alone. While increasing damping is an effective means of reducing projected velocity, it does not affect the apparent mass term in the PFL constraint and therefore often leads to conservative behavior, characterized by abrupt damping injection and degraded interaction transparency. In contrast, coordinated adaptation of admittance inertia and damping allows the controller to act simultaneously on both terms governing the transferred energy.

Importantly, the coupling expressed in (5.8)–(5.9) does not imply any reliance on an accurate dynamic model of the robot. The apparent mass and damping arise from the desired admittance parameters defined at the control level and remain independent of the robot's physical inertia, which is handled by the low-level controller. Dynamic modeling therefore plays no functional role in enforcing the PFL constraint, but may still be employed for analysis and validation purposes, consistent with the methodological separation established in Chapter 3.

The discussion above establishes that safety feasibility under PFL is inherently intertwined with the selection of admittance parameters. The following sections build on this observation by introducing two distinct strategies for adapting $\mathbf{M}_a$ and $\mathbf{D}_a$ online: an algorithmic approach and an optimization-based approach, both designed to maintain the PFL residual within admissible bounds while preserving passivity and interaction quality.

5.3 Passivity Preservation Under PFL-Driven Admittance Adaptation

The passivity issues associated with time-varying admittance parameters have been rigorously addressed in Chapter 4 through the introduction of an energy-tank mechanism (Section 4.3). That formulation is adopted here *without modification* and will not be re-derived. Instead, the purpose of this section is to clarify how the same passivity-preservation architecture operates when admittance parameters are adapted in response to a *Power and Force Limiting (PFL)* safety objective, rather than to distance-based supervision.

What changes from Chapter 4. In Chapter 4, online variations of the admittance inertia and damping were driven by proximity- and distance-related safety indicators inspired by Speed and Separation Monitoring. In the present chapter, parameter adaptation is instead driven by the PFL residual (5.7), which encodes feasibility of the transferred-energy constraint defined by ISO/TS 15066. As discussed in Sections 5.1 and 5.2, this residual depends explicitly on: (i) the apparent mass $m_r = \mathbf{n}^\top \mathbf{M}_a \mathbf{n}$ shaped by the admittance inertia, and (ii) the projected relative velocity $\dot{x}_{Rn} + \dot{x}_{Hn}$ shaped by the admittance damping. Consequently, PFL-driven safety regulation may require rapid, coordinated, or directional changes of $\mathbf{M}_a$ and $\mathbf{D}_a$, potentially inducing active behavior if not properly supervised.

What does *not* change. Crucially, the energetic mechanism through which time-varying inertia can inject power into the interaction port remains identical. As shown in Chapter 4, the only source of potential activity introduced by parameter adaptation is the term

$$\frac{1}{2}\dot{\boldsymbol{x}}^\top \dot{\mathbf{M}}_a(t)\, \dot{\boldsymbol{x}}, \tag{5.11}$$

which appears in the power balance of the variable-admittance system. The energy-tank formulation introduced in Section 4.3 explicitly accounts for this term by requiring that any inertia-induced power injection be compensated by energy previously dissipated through damping and stored in the tank.

Admissible inertia variation under PFL-driven adaptation. Since the admittance inertia $\mathbf{M}_a(t)$ is diagonal, Chapter 4 derived a component-wise upper bound on the admissible inertia rate that preserves passivity, expressed through the diagonal matrix

$$\dot{\mathbf{M}}^{\text{pass}}_{a,\max}(t), \tag{5.12}$$

whose elements depend on the current tank energy level, prescribed tank bounds $(T_{\min}, T_{\max})$, velocity limits, and energy-distribution weights (see Section 4.3.2). In the present chapter, this bound plays exactly the same supervisory role: any safety-driven request for inertia variation (whether arising from algorithmic adaptation (Section 5.4) or from optimization-based adaptation (Section 5.5) is filtered so as to satisfy

$$\dot{\mathbf{M}}_a(t) \leq \dot{\mathbf{M}}^{\text{pass}}_{a,\max}(t). \tag{5.13}$$

When the tank energy approaches $T_{\min}$, admissible inertia increases vanish, effectively freezing further apparent-mass growth until the tank is replenished through dissipated power.

Architectural separation between safety and passivity. It is important to emphasize that, also in the PFL setting, the energy tank does *not* enforce safety constraints and does *not* provide collision-injury guarantees. Its function is strictly to preserve passivity of the interaction port $(\boldsymbol{w}, \dot{\boldsymbol{x}})$ under time-varying admittance parameters. Safety feasibility is instead determined entirely by the PFL residual

and by the associated adaptation laws that regulate apparent mass and velocity. Architecturally, the tank therefore acts as a passivity supervisor interposed between the safety-driven adaptation logic and the actual applied parameters, enforcing energetic admissibility without interfering with the safety objective itself.

Implications for the adaptation strategies. This separation is what enables the two PFL-oriented adaptation strategies developed in the remainder of the chapter. The algorithmic method (Section 5.4) explicitly computes inertia and damping updates to regulate the PFL residual, while the optimization-based method (Section 5.5) treats inertia and damping as coupled decision variables constrained by feasibility, bounds, and passivity. In both cases, the energy tank guarantees that any admissible solution remains passive, ensuring stable and predictable physical interaction even under aggressive safety-driven parameter reshaping.

5.4 Algorithmic Safety Adaptation: Nominal Interaction vs. Safe Correction

Section 5.1 framed PFL as a *state feasibility* condition expressed through the dissipated collision energy $\Delta K_e \in \mathbb{R}_{\geq 0}$ and the residual $h \in \mathbb{R}$ in (5.7)–(5.2), consistently with standard energy-based PFL treatments and ISO/TS 15066 [1, 4, 50]. Section 5.2 then made explicit the control-relevant coupling: under Cartesian admittance, the apparent mass entering PFL is shaped by the virtual inertia $\mathbf{M}_a \in \mathbb{R}^{6\times 6}$ through (5.8) (see also the explicit coupling discussion in energy-based safe admittance formulations [55]). The present section operationalizes this coupling into an *algorithmic* adaptation mechanism that modifies $\mathbf{M}_a$ and $\mathbf{D}_a$ online with two distinct purposes: (i) in *nominal interaction*, reshape the admittance smoothly as the predicted impact energy increases, while preserving a consistent interaction feel via inertia–damping ratio tracking; (ii) in *safe correction*, apply a targeted and short-lived damping injection to rapidly restore feasibility when the PFL residual becomes positive.

The key design choice is to treat *inertia–damping co-modulation* as the primary lever for apparent-mass shaping (thus acting directly on the PFL term m_r), while treating *damping-only modulation* as the primary lever for fast dissipation when feasibility is violated. Since time variation of inertia can inject energy through the $\dot{\mathbf{M}}_a$ term in the admittance energy balance (Chapter 4, Section 4.3), all inertia updates are constrained by the passivity-aware bound produced by the energy-tank supervisor [55]. Conversely, damping increases are dissipative at the interaction port and therefore remain admissible from a passivity standpoint.

5.4.1 One-step safety gating and energy-loss scaling

The algorithm operates along the potential impact direction $\mathbf{n} \in \mathbb{R}^6$ introduced in Section 5.1. Let the scalar projected velocities be $\dot{x}_{Rn} \in \mathbb{R}$ and $\dot{x}_{Hn} \in \mathbb{R}$, and define the approach/separation indicator $s \in \{-1, +1\}$ as in (5.6) so that the PFL condition is treated as relevant only when the relative motion is approaching along $\mathbf{n}$ (Section 5.1; see also [1, 4, 50]). This directional gating is not a mere implementation detail: it encodes the semantic scope of the anticipatory residual h, which is meaningful as a predictor of *potential* collision severity only when the robot is moving toward the human along the line of approach.

In nominal operation, the adaptation intensity is parameterized by a normalized energy-loss factor $\beta \in [0, 1]$:

$$\beta = \min\left(\frac{\Delta K_e}{E_{\max}^{\mathrm{PFL}}},\, 1\right), \tag{5.14}$$

computed only if $s = +1$ (approaching motion). Here ΔK_e is the dissipated collision energy in (5.1) and $E_{\max}^{\mathrm{PFL}} \in \mathbb{R}_{>0}$ is the admissible PFL bound (ISO/TS 15066) [50]. The role of β is purely *shaping*: it does not enforce feasibility by itself, but provides a monotone mapping from predicted energy loss to a desired inertial reconfiguration.

5.4.2 Nominal interaction mode: ratio-preserving inertia–damping co-adaptation

Let $\mathbf{M}_a^\star \in \mathbb{R}^{6\times6}$ and $\mathbf{D}_a^\star \in \mathbb{R}^{6\times6}$ denote the nominal admittance inertia and damping matrices selected for comfortable hand-guiding. Let $\mathbf{M}_a^{\max} \in \mathbb{R}^{6\times6}$ denote a conservative upper bound on the admittance inertia, introduced to enlarge the safety margin by limiting the maximum apparent mass along the interaction direction.

In nominal interaction, the algorithm constructs a *temporary target inertia* $\mathbf{M}_{a,\mathrm{tmp}} \in \mathbb{R}^{6\times6}$ as a function of the normalized energy-loss factor $\beta \in [0, 1]$:

$$\mathbf{M}_{a,\mathrm{tmp}} = \mathbf{M}_a^\star + \beta\,(\mathbf{M}_a^{\max} - \mathbf{M}_a^\star), \tag{5.15}$$

which interpolates between the nominal and conservative inertia values.

Let $\Delta t \in \mathbb{R}_{>0}$ denote the controller update interval (one control cycle). The desired inertia rate $\dot{\mathbf{M}}_{a,\mathrm{nom}} \in \mathbb{R}^{6\times6}$ is defined as

$$\dot{\mathbf{M}}_{a,\mathrm{nom}} = \frac{\mathbf{M}_{a,\mathrm{tmp}} - \mathbf{M}_a}{\Delta t}, \tag{5.16}$$

and is applied subject to the passivity-aware bound returned by the energy-tank supervisor (Chapter 4, Section 4.3) [55].

In the algorithmic strategy, damping is not adapted independently in nominal conditions. Instead, a fixed inertia–damping ratio is enforced to preserve a consistent

Algorithm 4 Nominal interaction update (ratio-preserving co-adaptation)

Require: $s \in \{-1, +1\}$, $\beta \in [0, 1]$, $\Delta t \in \mathbb{R}_{>0}$
Require: current $\mathbf{M}_a \in \mathbb{R}^{6\times6}$, current $\mathbf{D}_a \in \mathbb{R}^{6\times6}$
Require: nominal $\mathbf{M}_a^\star \in \mathbb{R}^{6\times6}$, nominal $\mathbf{D}_a^\star \in \mathbb{R}^{6\times6}$, bound $\mathbf{M}_a^{\max} \in \mathbb{R}^{6\times6}$
Ensure: updated $\mathbf{M}_a$, updated $\mathbf{D}_a$
if $s = -1$ **then** ▷ separating motion: no safety-driven reshaping
  $\mathbf{M}_a \leftarrow \mathbf{M}_a^\star$
  $\mathbf{D}_a \leftarrow \mathbf{D}_a^\star$
  return
end if
$\mathbf{R}_d \leftarrow \mathbf{D}_a^\star(\mathbf{M}_a^\star)^{-1}$
$\mathbf{M}_{a,\mathrm{tmp}} \leftarrow \mathbf{M}_a^\star + \beta(\mathbf{M}_a^{\max} - \mathbf{M}_a^\star)$
$\dot{\mathbf{M}}_{a,\mathrm{nom}} \leftarrow (\mathbf{M}_{a,\mathrm{tmp}} - \mathbf{M}_a)/\Delta t$
$\dot{\mathbf{M}}_a \leftarrow \textsc{TankBound}(\dot{\mathbf{M}}_{a,\mathrm{nom}})$ ▷ Chapter 4, Sec. 4.3
$\mathbf{M}_a \leftarrow \mathbf{M}_a + \dot{\mathbf{M}}_a \Delta t$
$\mathbf{D}_a \leftarrow \mathbf{R}_d \mathbf{M}_a$

interaction feel (ratio preservation is a standard perceptual objective in rule-based variable admittance) [1, 4]. Define the ratio matrix $\mathbf{R}_d \in \mathbb{R}^{6\times6}$ as:

$$\mathbf{R}_d \triangleq \mathbf{D}_a^\star(\mathbf{M}_a^\star)^{-1}, \tag{5.17}$$

and impose the ratio-preserving update

$$\mathbf{D}_a \leftarrow \mathbf{R}_d \mathbf{M}_a. \tag{5.18}$$

Equation (5.18) should be read as an *assignment rule*: in nominal interaction, the current damping matrix $\mathbf{D}_a$ is set to the ratio-consistent value determined by the current inertia matrix $\mathbf{M}_a$. This makes explicit the intended semantics that were only implicit in the coupling argument of Section 5.2: as $\mathbf{M}_a$ is reshaped to affect the apparent mass m_r, the dissipative behavior is reshaped to change the robot velocity $\dot{x}_{Rn}$ and to preserve a stable, predictable feel.

For completeness and to avoid leaving the nominal mode as a verbal description, Algorithm 4 summarizes the one-step logic executed at each control cycle.

5.4.3 Safe correction mode: feasibility restoration by damping injection

When the PFL residual becomes positive ($h > 0$ under approaching motion), nominal inertia modulation may be insufficient to restore feasibility in a timely manner. In this case, the algorithm switches to a *safe correction mode*, explicitly designed to reduce the residual by increasing dissipation while freezing inertia [55]. As already discussed in Section 5.1, the inequality $h > 0$ does not imply that ISO/TS 15066 is being violated *at that instant* (the standard constrains the exchanged quantities

at contact) but it does indicate that, if a collision were to occur under the current relative approach conditions, the dissipated energy would exceed the admissible bound. The safe correction mode is therefore interpreted as an *anticipatory recovery action*: it reshapes the interaction dynamics so as to drive the system back to the feasibility boundary $h \leq 0$ within a single control step (or a small number of steps) under the discretized admittance model, thereby preventing the interaction from evolving deeper into an unsafe region.

Projecting the admittance dynamics along the interaction direction $\mathbf{n}$ yields the scalar model (5.9), where $m_r \in \mathbb{R}_{>0}$ (5.8) and $d_r \in \mathbb{R}_{>0}$ (5.10) are the projected apparent inertia and damping, and $w_n \in \mathbb{R}$ is the projected wrench component $w_n = \mathbf{n}^\top \boldsymbol{w}$. The strategy adopted in this work follows this constructive procedure: it explicitly computes (i) the *boundary* robot projected velocity that would satisfy the PFL energy constraint with equality, and then (ii) the damping update that makes the robot projected velocity reach that boundary value in *one* sampling period Δt.

Boundary velocity consistent with the PFL energy limit. When $h > 0$, the first step consists in computing a velocity limit $\dot{x}_{\text{lim}} \in \mathbb{R}$ such that the PFL residual evaluated at $\dot{x}_{Rn} = \dot{x}_{\text{lim}}$ lies exactly on the feasibility boundary. Starting from the PFL dissipated energy model (5.1), the equality condition $h = 0$ can be rewritten as

$$m_r m_h \left\| \dot{x}_{\text{lim}} + \dot{x}_{Hn} \right\|^2 - 2E_{\max}^{\text{PFL}} \left(m_h + m_r \right) = 0, \tag{5.19}$$

which can be solved analytically to obtain $\dot{x}_{\text{lim}}$. Among the two solutions of (5.19), the implementation selects the one closer to the currently measured $\dot{x}_{Rn}$, so that the corrective action is minimally invasive while still restoring feasibility.

One-step damping synthesis from the inverted admittance update. Once $\dot{x}_{\text{lim}}$ is available, the objective becomes to compute an updated damping matrix $\mathbf{D}_a^{new} \in \mathbb{R}^{6\times 6}$ such that, after one controller update of duration $\Delta t \in \mathbb{R}_{>0}$, the robot projected velocity along $\mathbf{n}$ reaches $\dot{x}_{\text{lim}}$. This is achieved by inverting the admittance update over one sampling interval. Using the admittance dynamics in (4.1) and a one-step discretization, the projected update can be written in the form:

$$\dot{x}_{\text{lim}} = \dot{x}_{Rn} + \mathbf{n}^\top \left[\mathbf{M}_a^{-1} \left(\boldsymbol{w} - \mathbf{D}_a^{new} \dot{\boldsymbol{x}} \right) \right] \Delta t. \tag{5.20}$$

Equation (5.20) involves $\mathbf{D}_a^{new}$ as a 6×6 unknown, and therefore does not admit a unique closed-form solution without additional structure. The correction is restricted to a uniform damping variation on the translational subspace by imposing

$$\mathbf{D}_a^{new} = \mathbf{D}_a + \dot{d}\, \mathbf{\Phi}\, \Delta t, \tag{5.21}$$

where $\mathbf{\Phi} \in \mathbb{R}^{6\times 6}$ is defined as:

$$\mathbf{\Phi} = \text{diag}(1, 1, 1, 0, 0, 0) \tag{5.22}$$

and where $\dot{d} \in \mathbb{R}$ is a scalar gain to be computed. Note that although the projection direction $\mathbf{n} \in \mathbb{R}^6$ is kept in full, the correction is restricted to the translational

subspace through $\mathbf{\Phi}$ in (5.22), so that the injected damping does not affect rotational dynamics.

Substituting (5.21) into (5.20) yields a scalar equation in $\dot{d}$, which can be solved explicitly as

$$\dot{d} = \frac{\dot{x}_{\lim} - \dot{x}_{Rn} - \mathbf{n}^\top [\mathbf{M}_a^{-1} (\boldsymbol{w} - \mathbf{D}_a \dot{\boldsymbol{x}})] \, \Delta t}{\mathbf{n}^\top \mathbf{M}_a^{-1} \mathbf{\Phi} \dot{\boldsymbol{x}} \, \Delta t^2}. \tag{5.23}$$

The damping update applied during safe correction is then obtained directly from (5.21), i.e. by setting $\mathbf{D}_a \leftarrow \mathbf{D}_a^{new}$. Two properties make this corrective action particularly attractive for real-time use: (i) it enforces feasibility *by construction* in one step (under the assumptions of the discretized model), and (ii) it modifies only damping, which is always dissipative at the interaction port and therefore legitimate from a passivity standpoint, even during abrupt transients.

Recovery phase: avoiding repeated abrupt interaction changes. The safe correction step computed through (5.19)–(5.23) may result in a large instantaneous damping increase, which can substantially alter the perceived interaction feel. To avoid repeated drastic changes, the strategy includes a recovery phase: for a fixed time window $t_{\max} \in \mathbb{R}_{>0}$ after a correction is triggered, the damping is not immediately returned to nominal ratio tracking, but is instead exponentially adapted through

$$\mathbf{D}_a \leftarrow \delta \, \mathbf{D}_a + (1 - \delta) \, \mathbf{D}_a^\star, \qquad \delta \in (0, 1), \tag{5.24}$$

where $\mathbf{D}_a^\star \in \mathbb{R}^{6\times 6}$ denotes the nominal damping matrix. If during this decay phase the PFL residual becomes positive again ($h > 0$), the recovery timer is re-initialized and the safe correction step is re-applied. This re-triggering rule prevents the controller from "relaxing" damping too quickly in the presence of persistent unsafe approach conditions.

Operationally, safe correction proceeds in two phases:

(i) Injection. When $s = +1$ and $h > 0$, inertia is frozen ($\mathbf{M}_a$ held constant) and the boundary velocity $\dot{x}_{\lim}$ is computed from (5.19). The damping matrix is then updated through (5.21)–(5.23) so that the robot projected velocity reaches $\dot{x}_{\lim}$ in one control interval Δt. The resulting $\mathbf{D}_a$ can depart significantly from the ratio-tracking assignment (5.18), and therefore constitutes an explicitly safety-driven deviation from nominal "feel".

(ii) Recovery by exponential decay. To avoid repeated abrupt changes in interaction perception, once feasibility is restored the injected damping is not immediately removed. Instead, for a bounded recovery window $t_{\max}$, the algorithm exponentially decays the damping toward the nominal matrix $\mathbf{D}_a^\star$ using (5.24). If a new violation occurs, injection is re-triggered and the timer reset.

Algorithm 5 Safe correction logic (damping injection and recovery)

Require: $h \in \mathbb{R}$, $s \in \{-1, +1\}$, $\mathbf{n} \in \mathbb{R}^6$
Require: $\Delta t \in \mathbb{R}_{>0}$, $\delta \in (0, 1)$, $t_{\max} \in \mathbb{R}_{>0}$
Require: current time $t \in \mathbb{R}$, stored $t_{\text{start}} \in \mathbb{R}$
Require: current $\mathbf{M}_a \in \mathbb{R}^{6\times6}$, current $\mathbf{D}_a \in \mathbb{R}^{6\times6}$
Require: nominal $\mathbf{D}_a^\star \in \mathbb{R}^{6\times6}$, $m_h \in \mathbb{R}_{>0}$, $E_{\max}^{\text{PFL}} \in \mathbb{R}_{>0}$
Require: measured $\dot{x}_{Rn}(t) \in \mathbb{R}$, $\dot{x}_{Hn}(t) \in \mathbb{R}$, $\boldsymbol{w}(t) \in \mathbb{R}^6$, $\dot{\boldsymbol{x}}(t) \in \mathbb{R}^6$
Ensure: updated $\mathbf{D}_a$ and updated t_{start}

if $s = +1$ **and** $h > 0$ **then** ▷ constraint violation while approaching
  $m_r \leftarrow \mathbf{n}^\top \mathbf{M}_a \mathbf{n}$
  Solve (5.19) for $\dot{x}_{\lim}$ (choose root closest to $\dot{x}_{Rn}(t)$)
  $\dot{d} \leftarrow$ compute from (5.23)
  $\mathbf{D}_a \leftarrow \mathbf{D}_a + \dot{d}\,\boldsymbol{\Phi}\,\Delta t$ ▷ structured injection, (5.21)
  $t_{\text{start}} \leftarrow t$ ▷ reset recovery timer
else
  if $t - t_{\text{start}} \le t_{\max}$ **then** ▷ recovery window active
    $\mathbf{D}_a \leftarrow \delta\,\mathbf{D}_a + (1-\delta)\,\mathbf{D}_a^\star$ ▷ decay, (5.24)
  else
    $\mathbf{D}_a \leftarrow \mathbf{D}_a^\star$
  end if
end if
return $\mathbf{D}_a$, t_{start}

The complete logic is given in Algorithm 5, which is referenced by the text above and should be read as the algorithmic counterpart of the two-mode interpretation.

Algorithm 4 and 5 make explicit the conceptual separation that will matter later when comparing against the optimization-based strategy: the algorithmic method relies on (i) a ratio-preserving nominal modulation that links inertia and damping, and (ii) an explicit mode-dependent corrective action that temporarily breaks that ratio via a computed damping update derived from the boundary velocity $\dot{x}_{\lim}$ and the inverted one-step admittance update. This is exactly the point where the optimization-based approach will diverge: rather than enforcing ratio tracking and then breaking it through a constructive rule, the optimizer will treat inertia and damping as coupled decision variables constrained by feasibility and passivity, and will let the trade-off between apparent mass shaping and dissipation emerge from the optimization structure (Section 5.5; see also [1, 4, 50, 55]).

5.5 Optimization-Based Safety Adaptation Under PFL Constraints

The algorithmic strategy in Section 5.4 enforces PFL feasibility through a constructive two-mode logic: ratio-preserving co-modulation in nominal interaction, and a

damping-only corrective step when $h > 0$. While this provides an interpretable and lightweight solution, it also exposes a structural limitation that becomes evident in sustained or highly dynamic pHRI: the controller must *decide* how to trade off (i) apparent-mass shaping (via $\mathbf{M}_a$) and (ii) dissipation shaping (via $\mathbf{D}_a$) under explicit feasibility constraints, *without* relying on ad-hoc switching rules or fixed ratio assumptions.

The second contribution of this chapter therefore replaces the rule-based switching logic with an optimization layer that, at each sampling instant, selects the *most suitable coordinated update* of the admittance parameters under a set of explicit constraints: (i) PFL feasibility expressed through the residual h in (5.7); (ii) hard bounds on admissible inertia/damping values; (iii) rate bounds (including the passivity-aware bound enforced by the energy tank, Chapter 4, Section 4.3); and (iv) an objective encoding the desired interaction behavior. The key conceptual difference from Section 5.4 is that inertia and damping are treated as *coupled decision variables* rather than being linked by a fixed ratio and then heuristically "broken" during correction: the optimizer is allowed to violate ratio tracking if this improves feasibility restoration or reduces unnecessary dissipation, while still remaining within explicit constraints.

5.5.1 Decision variables and discrete-time parameter updates

In the following implementation only the *translational* admittance parameters are adapted online. Let the translational diagonal blocks for the inertia ($\mathbf{M}_t \in \mathbb{R}^{3\times3}$) and damping ($\mathbf{D}_t \in \mathbb{R}^{3\times3}$) matrices be represented as:

$$\mathbf{M}_t = \mathrm{diag}(\boldsymbol{m}), \quad \mathbf{D}_t = \mathrm{diag}(\boldsymbol{d}), \tag{5.25}$$

with $\boldsymbol{m} \in \mathbb{R}^3$ and $\boldsymbol{d} \in \mathbb{R}^3$ collecting the translational diagonal elements. At each sampling step of duration $\Delta t \in \mathbb{R}_{>0}$, the optimizer selects translational *rate* increments

$$\mathbf{u} = \begin{bmatrix} \dot{\boldsymbol{m}} \\ \dot{\boldsymbol{d}} \end{bmatrix} \in \mathbb{R}^6, \qquad \dot{\boldsymbol{m}} \in \mathbb{R}^3,\ \dot{\boldsymbol{d}} \in \mathbb{R}^3, \tag{5.26}$$

which are applied through the discrete-time update rule

$$\mathbf{M}_t^+ = \mathrm{diag}(\boldsymbol{m}) + \mathrm{diag}(\dot{\boldsymbol{m}})\Delta t, \qquad \mathbf{D}_t^+ = \mathrm{diag}(\boldsymbol{d}) + \mathrm{diag}(\dot{\boldsymbol{d}})\Delta t, \tag{5.27}$$

where $(\cdot)^+$ denotes the value at the next control instant. This representation makes explicit the intended control semantics: the optimization does not "pick" parameters arbitrarily, but selects *admissible rates* that can be safely implemented within the passivity supervision architecture already established in Chapter 4.

5.5.2 One-step prediction of projected robot motion and feasibility

Since the optimization adapts only the translational admittance blocks, we introduce the translational projection of the unit approach direction,

$$\mathbf{n}_t \triangleq [n_x \; n_y \; n_z]^\top \in \mathbb{R}^3, \tag{5.28}$$

i.e., the first three components of the full vector $\mathbf{n} \in \mathbb{R}^6$ defined in Section 5.1.2. All predictive quantities in the optimization layer are therefore evaluated along $\mathbf{n}_t$.

The feasibility residual in (5.7) depends on (i) the relative projected velocity $\dot{x}_{Rn} + \dot{x}_{Hn}$ and (ii) the apparent mass m_r along the interaction direction. In the optimization strategy, feasibility is imposed *predictively* at the next time step as a function of the chosen update $\mathbf{u}$.

Let $\mathbf{J}_t(\boldsymbol{q}) \in \mathbb{R}^{3\times 6}$ denote the translational Jacobian (consistent with the translational-only adaptation assumption). The predicted robot projected velocity along $\mathbf{n}$ is modeled over one step as

$$\dot{x}_{R_n}^+ = \dot{x}_{R_n}(t) + \Big(\mathbf{n}_t^\top \mathbf{M}_t^{+^{-1}} \big(\boldsymbol{w}_t - \mathbf{D}_t^+ \, \dot{\boldsymbol{x}}_t\big)\Big)\Delta t, \tag{5.29}$$

where $\dot{x}_{R_n} \in \mathbb{R}$ is the current projected velocity at the current step, $\boldsymbol{w}_t \in \mathbb{R}^3$ is the measured interaction *force* (translational wrench component) and $\dot{\boldsymbol{x}}_t \in \mathbb{R}^3$ denotes the translational end-effector velocity (e.g., $\dot{\boldsymbol{x}}_t = \mathbf{J}_t(\boldsymbol{q})\dot{\boldsymbol{q}}$). Equation (5.29) is the optimization counterpart of the one-step inversion logic used in safe correction (Section 5.4.3), but here it is embedded directly into a constrained decision problem: the next-step velocity is not "forced" to a boundary value; rather, it emerges from the chosen $(\mathbf{M}_t^+, \mathbf{D}_t^+)$.

The predicted apparent mass entering the PFL residual is

$$m_r^+ = \mathbf{n}_t^\top \mathbf{M}_t^+ \mathbf{n}_t, \tag{5.30}$$

which mirrors the coupling already established in (5.8), but expressed on the predicted translational inertia.

Using these predicted quantities, the PFL residual is evaluated as a function of the decision variable $\mathbf{u}$:

$$h(\mathbf{u}) = s \, m_r^+ m_h \left\| \dot{x}_{R_n}^+ + \dot{x}_{H_n} \right\|^2 - 2E_{\max}^{\mathrm{PFL}} \left(m_h + m_r^+\right), \tag{5.31}$$

where $m_h \in \mathbb{R}_{>0}$ and $E_{\max}^{\mathrm{PFL}} \in \mathbb{R}_{>0}$ are the same quantities introduced in Section 5.1. The directional gating is retained exactly as before:

$$s = \begin{cases} +1, & \dot{x}_{R_n} + \dot{x}_{H_n} \geq 0, \\ -1, & \dot{x}_{R_n} + \dot{x}_{H_n} < 0, \end{cases} \tag{5.32}$$

so that feasibility is enforced primarily when approaching motion makes the residual meaningful as an anticipatory safety indicator (Section 5.1.2).

5.5.3 Constraints: bounds, rates, and passivity-supervised inertia variation

The optimization problem is constrained at three complementary levels.

(i) Parameter bounds. Admittance parameters must remain within prescribed limits. In the optimization formulation, these limits are encoded as component-wise bounds on the translational diagonal parameters:

$$\boldsymbol{m}_{\min} \leq \boldsymbol{m}^{+} \leq \boldsymbol{m}_{\max}, \qquad \boldsymbol{d}_{\min} \leq \boldsymbol{d}^{+} \leq \boldsymbol{d}_{\max}, \tag{5.33}$$

where $\boldsymbol{m}^{+}, \boldsymbol{d}^{+} \in \mathbb{R}^3$ are the predicted diagonal vectors associated with $\mathbf{M}_t^{+}, \mathbf{D}_t^{+}$ (Eq. (5.27)). Here $\boldsymbol{m}_{\min} \in \mathbb{R}^3$ and $\boldsymbol{m}_{\max} \in \mathbb{R}^3$ denote, respectively, the minimum and maximum admissible translational inertia diagonals, while $\boldsymbol{d}_{\min} \in \mathbb{R}^3$ and $\boldsymbol{d}_{\max} \in \mathbb{R}^3$ denote the minimum and maximum admissible translational damping diagonals. These bounds reflect both physical considerations (e.g., avoiding excessively light apparent inertia that would amplify noise and hand tremor, or excessively high damping that would suppress transparency) and implementation constraints (e.g., ensuring numerical conditioning and compatibility with the low-level robot controller).

(ii) Rate bounds. To avoid aggressive transients in the interaction dynamics and to reflect both actuation and implementation limitations, rate constraints are imposed directly on the decision variables governing inertia and damping adaptation:

$$\dot{\boldsymbol{m}}_{\min} \leq \dot{\boldsymbol{m}} \leq \dot{\boldsymbol{m}}_{\max}, \qquad \dot{\boldsymbol{d}}_{\min} \leq \dot{\boldsymbol{d}} \leq \dot{\boldsymbol{d}}_{\max}. \tag{5.34}$$

Here $\dot{\boldsymbol{m}}, \dot{\boldsymbol{d}} \in \mathbb{R}^3$ denote the translational inertia and damping rate vectors treated as decision variables in the optimization. The bounds $\dot{\boldsymbol{m}}_{\min}, \dot{\boldsymbol{m}}_{\max} \in \mathbb{R}^3$ and $\dot{\boldsymbol{d}}_{\min}, \dot{\boldsymbol{d}}_{\max} \in \mathbb{R}^3$ define the minimum and maximum admissible rates of change for inertia and damping, respectively. These constraints serve a dual purpose: they enforce smoothness of the interaction by preventing abrupt parameter variations that would degrade transparency or induce instability.

(iii) Passivity-aware inertia rate bound. Finally, because time variation of the admittance inertia is the only source of potential energy injection in the admittance energy balance (Chapter 4, Section 4.3), the inertia-rate decision is additionally constrained by the passivity supervisor implemented through the energy-tank mechanism:

$$\dot{\boldsymbol{m}}_i \leq \dot{\boldsymbol{m}}_{t,\max}^{\text{pass}}, \qquad \forall i \in \{1, \ldots, 3\}, \tag{5.35}$$

where $\dot{\boldsymbol{m}}_{t,\max}^{\text{pass}} \in \mathbb{R}^3$ denotes the component-wise upper bound on the admissible inertia rate returned by the energy-tank supervisor at the current time step.

This bound is not a tuning parameter of the optimization problem, but a dynamic constraint inherited from the passivity analysis developed in Chapter 4. As

such, it enforces energetic admissibility of the inertia variation independently of the safety or performance objectives considered by the optimizer. The optimization layer may reason about safety feasibility and interaction quality, but it cannot request inertia changes that would violate the passivity condition enforced by the energy tank.

5.5.4 Objective function and problem formulations

The optimizer must encode two competing objectives: (i) remain close to a nominal interaction behavior, so as to preserve transparency and avoid excessive perceived "heaviness" or "stickiness"; (ii) regulate the parameters with minimal variation, so that adaptation does not itself become a source of perceptual discontinuity.

In this dissertation, these requirements are encoded through a quadratic objective function, designed to balance *interaction comfort anchoring* against *adaptation smoothness*:

$$\mathcal{L}(\mathbf{u}) = \lambda_M \left\| \boldsymbol{m}^+ - \boldsymbol{m}_a^\star \right\|^2 + \lambda_D \left\| \boldsymbol{d}^+ - \boldsymbol{d}_a^\star \right\|^2 + \lambda_u \left(\|\dot{\boldsymbol{m}}\|^2 + \|\dot{\boldsymbol{d}}\|^2 \right). \tag{5.36}$$

The first two terms penalize deviations of the predicted admittance parameters from their nominal values and therefore act as *comfort-anchoring* terms: they bias the optimizer toward interaction behaviors that remain perceptually close to the baseline hand-guiding admittance defined by $\mathbf{M}_a^\star$ and $\mathbf{D}_a^\star$. In the absence of active safety constraints, these terms make the nominal admittance an attractive equilibrium of the optimization problem.

The third term penalizes the rates of change of inertia and damping and serves as a *regularization* on the adaptation process itself. Its role is not related to safety feasibility, but to perceptual continuity: it discourages rapid parameter variations that would otherwise introduce abrupt changes in interaction feel, even when such variations are admissible from a constraint standpoint.

Here:

- $\boldsymbol{m}^+, \boldsymbol{d}^+ \in \mathbb{R}^3$ denote the predicted translational diagonal components associated with the updated admittance matrices $\mathbf{M}_a^+, \mathbf{D}_a^+$;
- $\boldsymbol{m}_a^\star, \boldsymbol{d}_a^\star \in \mathbb{R}^3$ are the translational diagonal components of the nominal admittance matrices $\mathbf{M}_a^\star, \mathbf{D}_a^\star$ introduced in Section 5.4.2;
- $\lambda_M, \lambda_D, \lambda_u \in \mathbb{R}_{>0}$ are weighting coefficients that regulate the trade-off between adherence to the nominal interaction behavior and smoothness of parameter adaptation.

Importantly, safety is *not* encoded in the objective, but enforced exclusively through constraints. This separation ensures that feasibility with respect to the

PFL residual is never traded against comfort, but instead acts as a hard (or relaxed) admissibility condition on the optimization outcome.

With this cost, the *hard-constraint* formulation reads

$$\begin{aligned} \min_{\mathbf{u}\in\mathbb{R}^6} \quad & \mathcal{L}(\mathbf{u}) \\ \text{s.t.} \quad & h(\mathbf{u}) \le 0, \\ & (5.33),\ (5.34),\ (5.35). \end{aligned} \tag{5.37}$$

In practice, strict feasibility may be temporarily unattainable due to sensor noise, abrupt human motion, or conflicting bounds. For this reason, this work also introduces a *soft-constraint* variant with slack:

$$\begin{aligned} \min_{\mathbf{u}\in\mathbb{R}^6,\ \sigma\ge 0} \quad & \mathcal{L}(\mathbf{u}) + \rho\,\sigma \\ \text{s.t.} \quad & h(\mathbf{u}) - \sigma \le 0, \\ & (5.33),\ (5.34),\ (5.35), \end{aligned} \tag{5.38}$$

where $\sigma \in \mathbb{R}_{\ge 0}$ relaxes the feasibility constraint and $\rho \in \mathbb{R}_{>0}$ penalizes violations.

The conceptual role of (5.38) is crucial for the thesis narrative. Unlike the algorithmic strategy (Section 5.4), which introduces an explicit corrective mode followed by a recovery window, the optimization-based approach embeds the trade-off between safety enforcement and interaction comfort directly into a single constrained problem. The slack variable provides a principled mechanism for graceful degradation when strict feasibility cannot be achieved within the admissible parameter and rate bounds, rather than forcing discontinuous mode switching.

Interpretation of hard vs. soft feasibility enforcement. The distinction between the hard- and soft-constrained formulations in (5.37)–(5.38) is not merely a numerical convenience, but reflects two fundamentally different interpretations of safety feasibility under PFL-driven interaction.

In the hard-constraint formulation (5.37), the PFL residual $h(\mathbf{u}) \le 0$ is treated as an inviolable condition. The optimizer is allowed to adjust inertia and damping only if a feasible solution exists that simultaneously satisfies the safety constraint, parameter bounds, rate limits, and passivity-aware inertia variation. This formulation is appropriate when the admissible set is sufficiently rich and when measurements are reliable, but it may become overly restrictive in practice: transient infeasibility can arise due to sensor noise, abrupt human motion, or the simultaneous activation of tight bounds on parameters and rates.

The soft-constraint formulation (5.38) relaxes this requirement by introducing the nonnegative slack variable σ, which explicitly quantifies the degree of violation of the PFL feasibility condition. From a physical perspective, σ does *not* correspond to a tolerated violation of ISO/TS 15066 at contact; rather, it represents a controlled

relaxation of the *anticipatory* feasibility condition encoded by h. The optimizer is therefore allowed to temporarily accept $h > 0$ when strict enforcement is incompatible with the admissible parameter variations, but such violations are penalized through the weight ρ in the cost.

This construction has two important implications. First, it guarantees that the optimization problem remains feasible at every time step, avoiding solver failure or discontinuous fallback strategies. Second, it embeds the safety–comfort trade-off directly into the optimization structure: when strict feasibility can be achieved, the slack collapses to zero and the solution coincides with the hard-constrained case; when it cannot, the optimizer selects the least-violating solution that best balances interaction comfort, smoothness, and safety recovery.

This behavior contrasts sharply with the algorithmic strategy of Section 5.4, where feasibility violations trigger an explicit mode switch and a heuristic recovery phase. In the optimization-based approach, no discrete modes are introduced: nominal interaction, corrective behavior, and recovery emerge continuously from the same constrained problem. This unification is a central conceptual advantage of the optimization-based formulation and will be revisited in Section 5.6 when discussing experimental behavior.

5.5.5 Implementation note and optimization-loop structure

This subsection summarizes the per-step execution logic of the optimization-based admittance adaptation strategy. The purpose is not to specify solver-dependent details, but to make explicit the sequence of operations performed at each control cycle and the interface between measurement, prediction, optimization, and passivity supervision.

Optimization-based update loop. At each control cycle t, the controller executes the following steps:

Solver and real-time considerations. The optimization problems (5.37) or (5.38) are solved online at each control cycle. In the experimental implementation described in the second paper, the problem size is kept intentionally small (six decision variables plus one slack variable) and all constraints are convex or convexified, enabling reliable convergence within the available control period.

Key implementation aspects discussed in the paper and relevant for reproducibility include: (i) solver choice and warm-starting strategy; (ii) handling of measurement noise in the evaluation of the PFL residual; (iii) saturation and fallback behavior in case of temporary infeasibility; (iv) interaction between the optimizer output and the passivity supervisor.

Algorithm 6 Optimization-based admittance adaptation (per-step loop)

Require: Current admittance matrices $\mathbf{M}_a(t), \mathbf{D}_a(t)$
Require: Nominal matrices $\mathbf{M}_a^\star, \mathbf{D}_a^\star$
Require: Measured $\dot{x}_{Rn}(t), \dot{x}_{Hn}(t), \boldsymbol{w}(t), \dot{\boldsymbol{x}}(t)$
Require: Bounds (5.33), (5.34), (5.35)
Require: Sampling time Δt
Ensure: Updated matrices $\mathbf{M}_a^+, \mathbf{D}_a^+$
Scene monitoring and projection
Compute $s(t)$ using (5.6)
Evaluate PFL residual $h(t)$ using (5.7)
Prediction and decision-variable setup
Assemble constraint set: feasibility, bounds, rates, passivity
Optimization solve
Solve (5.37)
if solver fails or infeasible **then**
    Solve (5.38)
end if
Parameter update and supervision
Enforce passivity supervision via energy tank (Chapter 4, Section 4.3)
Update $\mathbf{M}_a^+, \mathbf{D}_a^+$ from optimizer output
return $\mathbf{M}_a^+, \mathbf{D}_a^+$

These aspects are not repeated here, but the algorithmic structure above is designed to accommodate them explicitly and to remain consistent with the passivity-preserving architecture established in Chapters 3 and 4.

5.6 Experimental Evaluation and Comparative Discussion

5.6.1 Experimental setup and protocol

The experimental evaluation of the adaptive admittance strategies developed in this chapter is conducted using a *common physical setup and interaction protocol* for both methods, so as to isolate the effect of the adaptation logic itself from confounding hardware, sensing, or task-related factors.

Experiments are performed on a collaborative robotic manipulator (UR10e) equipped with a wrist-mounted force/torque sensor and controlled through Cartesian admittance. The robot end-effector is directly guided by a human operator through physical interaction, while the controller adapts the admittance parameters online in response to the PFL safety residual defined in Section 5.1. The interaction includes both slow exploratory motions and faster, more impulsive approach motions,

Table 5.1: Experimental setup and control parameters used for the comparative evaluation of algorithmic and optimization-based admittance adaptation strategies (from nini2025_algo).

Category	**Value / Description**
Robot platform	Universal Robots UR10e
Degrees of freedom	6
End-effector sensing	Integrated UR e-series 6-axis F/T sensor
Control framework	Cartesian admittance control
Control frequency	$f_c = 500$ Hz
Interaction modality	pHRI Hand-guiding
Safety paradigm	Power and Force Limiting (PFL)
Nominal admittance inertia	$\mathbf{M}_a^\star = \mathrm{diag}(m_x, m_y, m_z, \cdot, \cdot, \cdot)$
Nominal translational inertia	$m_x = m_y = m_z = 5$ kg
Maximum admittance inertia	$\mathbf{M}_a^{\max} = \mathrm{diag}(m_x^{\max}, m_y^{\max}, m_z^{\max}, \cdot)$
Maximum translational inertia	$m_x^{\max} = m_y^{\max} = m_z^{\max} = 15$ kg
Nominal admittance damping	$\mathbf{D}_a^\star = \mathrm{diag}(d_x, d_y, d_z, \cdot, \cdot, \cdot)$
Nominal translational damping	$d_x = d_y = d_z = 25$ Ns/m
Inertia–damping ratio	Preserved in nominal mode: $\mathbf{R}_d = \mathbf{D}_a^\star(\mathbf{M}_a^\star)^{-1}$
Rotational admittance	Fixed (no adaptation)
Sampling time	$\Delta t = 2$ ms
Recovery window (algorithmic)	$t_{\max} = 1$ s
Damping decay factor	$\delta = 0.95$
Human motion	Unconstrained voluntary motion
Safety indicator	PFL residual h evaluated online
Compared strategies	Algorithmic adaptation (Sec. 5.4); Optimization-based adaptation (Sec. 5.5.4)

intentionally designed to challenge the safety mechanisms and trigger parameter adaptation.

At each control cycle, the perception and estimation pipeline provides: (i) the projected robot and human velocities $\dot{x}_{Rn}$ and $\dot{x}_{Hn}$ along the interaction direction $\mathbf{n}$; (ii) the interaction wrench $\boldsymbol{w}$ at the end-effector; (iii) the quantities required to evaluate the PFL residual h in (5.7). All experiments are executed at a fixed control frequency and under identical nominal admittance parameters $\mathbf{M}_a^\star, \mathbf{D}_a^\star$. A summary of the experimental platform, sensing, control architecture, and nominal admittance parameters common to both methods is reported in Table 5.1, and shown in Figure

5.6.2 Results with algorithmic safety adaptation

Representative time histories obtained with the algorithmic safety adaptation strategy are shown in Fig. 5.2. The figure focuses on a zoomed time interval selected to capture both nominal interaction and the onset of a safety-critical approach motion.

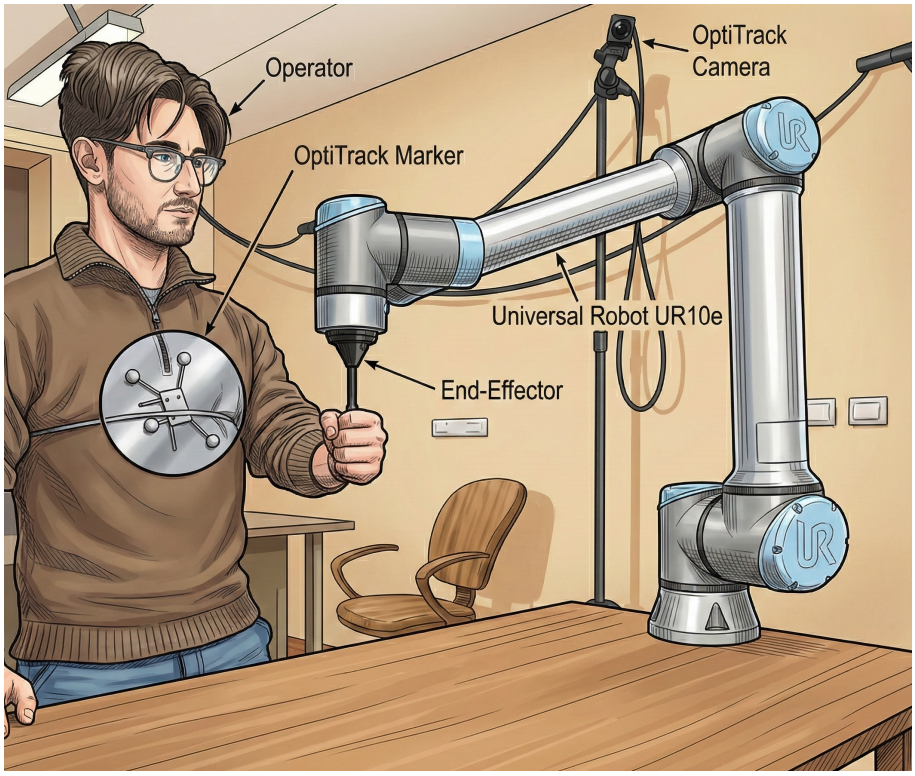


Figure 5.1: Illustration of the experimental setup for physical human-robot interaction. The system comprises a Universal Robot UR10e robotic manipulator and an OptiTrack motion capture system. The human operator holds the robot's end-effector, while an OptiTrack camera tracks a reflective marker cluster attached to the operator's chest to monitor their position during the collaborative task.

During the initial phase, the interaction evolves under the nominal, ratio-preserving admittance behavior. In this regime, the dissipated collision energy ΔK_e increases smoothly in response to operator-induced motion but remains below the admissible PFL bound $E_{\max}^{\mathrm{PFL}}$. As a result, the interaction impedance evolves continuously and is perceived as stable and predictable.

When a faster approach motion is introduced, ΔK_e exceeds the admissible energy threshold, corresponding to a violation of the anticipatory feasibility condition ($h > 0$). This instant is marked by the vertical dotted line in Fig. 5.2. As prescribed by the algorithmic strategy (Section 5.4), the controller immediately switches to the safe-correction mode. This transition is characterized by a sharp increase in the damping component d_1, while the corresponding inertia component m_1 is temporarily held constant. The resulting damping injection rapidly dissipates kinetic energy and drives the system back toward the feasibility boundary.

Once feasibility is restored, the controller enters the recovery phase. Here, the damping component decays exponentially toward its nominal ratio-consistent value, preventing repeated abrupt impedance changes while gradually restoring the nominal interaction feel. Throughout both the corrective and recovery phases, the energy tank level remains strictly positive and bounded, confirming that the aggressive damping action and the associated inertia handling remain compatible with passivity preservation.

The minimum human–robot distance is reported solely as a contextual diagnostic. It confirms that safety-driven parameter adaptation occurs during approaching motion, but it does not directly trigger the adaptation. The evolution of the tank energy T further illustrates the energetic role of inertia variation: whenever the controller increases inertia, energy is withdrawn from the tank to compensate for the associated power injection.

Overall, Fig. 5.2 highlights the defining characteristics of the algorithmic approach: a smooth, ratio-preserving evolution during nominal interaction, followed by a clearly segmented corrective phase that guarantees rapid feasibility restoration at the cost of a short-lived but perceptible discontinuity in interaction impedance.

5.6.3 Results with optimization-based safety adaptation

Representative time histories for the optimization-based safety adaptation strategy are shown in Fig. 5.3, obtained under the same experimental conditions and over a comparable zoomed time window as the algorithmic case. The figure illustrates how PFL feasibility is regulated through a unified constrained optimization process, without explicit mode switching.

At the beginning of the interval, both admittance inertia and damping remain close to their nominal values. This reflects an interaction regime in which the optimizer prioritizes transparency and minimal deviation from the baseline behavior,

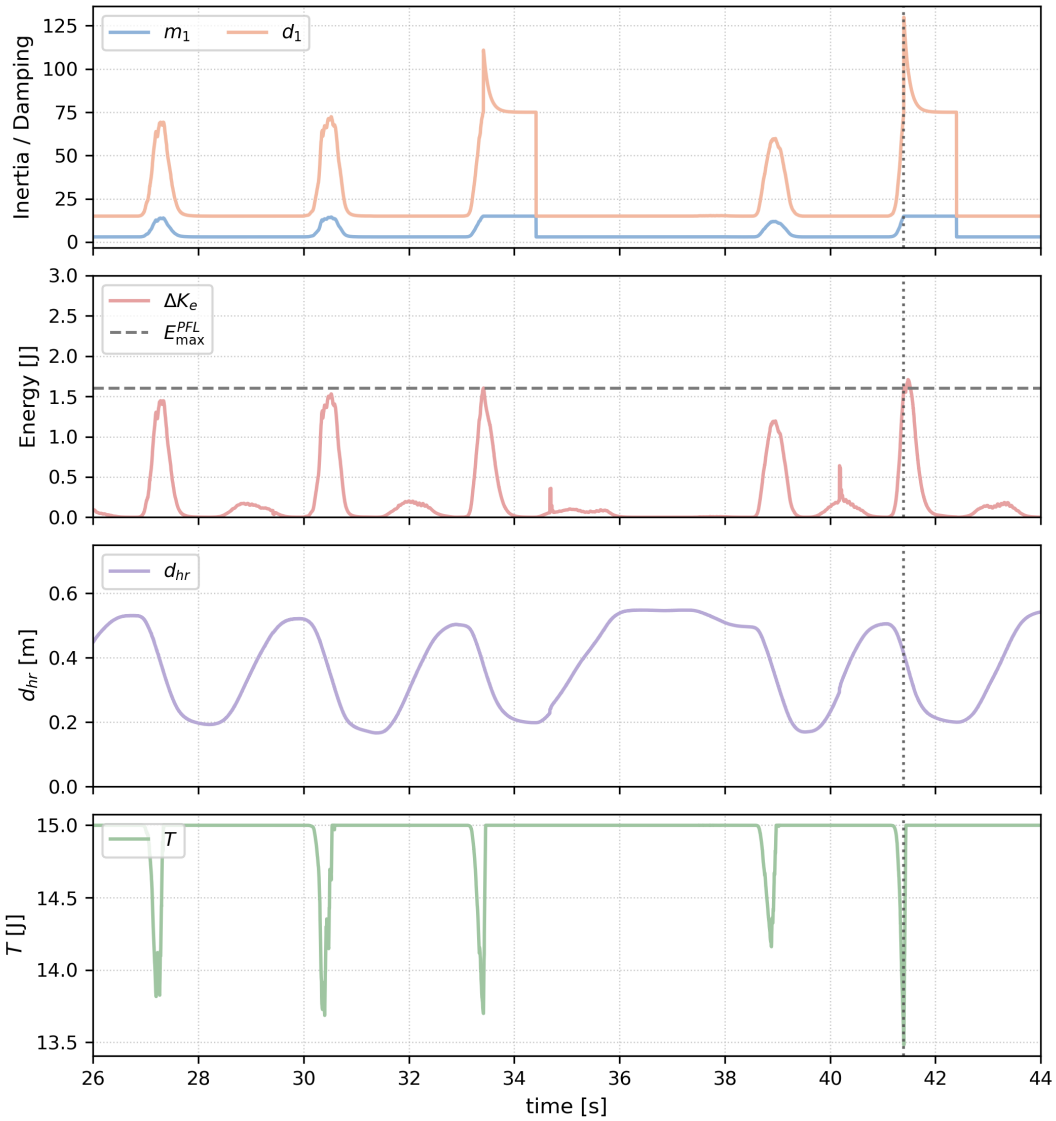


Figure 5.2: Representative time histories for the algorithmic PFL-based admittance adaptation. From top to bottom: evolution of the first diagonal admittance inertia and damping components (m_1, d_1), dissipated collision energy ΔK_e with respect to the PFL admissible bound $E^{PFL}_{\max}$, minimum human–robot distance d_{hr} (diagnostic), and energy tank level T. The vertical dotted line marks the onset of a fast approach motion leading to violation of the anticipatory feasibility condition ($h > 0$) and the switch to the Safe correction mode (5.4.3).

as long as the PFL residual remains well within the feasible region.

As the operator introduces faster approach motions, the PFL residual h increases and approaches the feasibility boundary. In contrast to the algorithmic strategy, no discrete corrective action is triggered. Instead, inertia and damping evolve smoothly and jointly, reflecting the optimizer's continuous trade-off between apparent mass reduction, dissipation increase, and adherence to parameter bounds, rate limits, and passivity constraints.

When strict feasibility cannot be instantaneously enforced due to these constraints, the soft-constraint formulation allows a controlled and penalized relaxation of the residual. This behavior manifests as a bounded and gradual excursion of h, rather than as an abrupt damping spike. As in the previous case, the minimum human–robot distance is reported only as a contextual diagnostic and confirms that parameter adaptation occurs during approaching motion. The evolution of the tank energy T shows that inertia increases are energetically compensated through tank depletion, consistent with the passivity-preserving architecture.

From the operator's perspective, this results in a more homogeneous interaction feel. Parameter adaptations occur less abruptly and with smaller amplitude compared to the algorithmic strategy. However, this smoother behavior is achieved by relaxing strict inertia–damping ratio preservation: inertia and damping are adjusted independently when this improves feasibility regulation under the imposed constraints.

5.6.4 Qualitative comparison and discussion

The experimental results highlight that the two approaches address the same safety objective through fundamentally different control philosophies.

The algorithmic strategy favors determinism and reactivity. Its explicit separation between nominal interaction and safe correction ensures rapid feasibility restoration and transparent cause–effect relationships between safety violations and controller response. However, this comes at the cost of perceptible interaction discontinuities during corrective phases and relies on heuristic mode switching.

The optimization-based strategy embeds safety, comfort, and smoothness into a single decision process. By avoiding explicit mode switching, it yields smoother parameter trajectories and a more uniform interaction feel. Its limitations are primarily practical rather than conceptual, including dependence on solver timing and the need for careful tuning of weights and bounds.

Importantly, neither approach can be declared superior in absolute terms. Rather, they represent complementary design points: the algorithmic method prioritizes predictability and rapid recovery, while the optimization-based method prioritizes continuity and integrated trade-offs. The choice between them depends on application

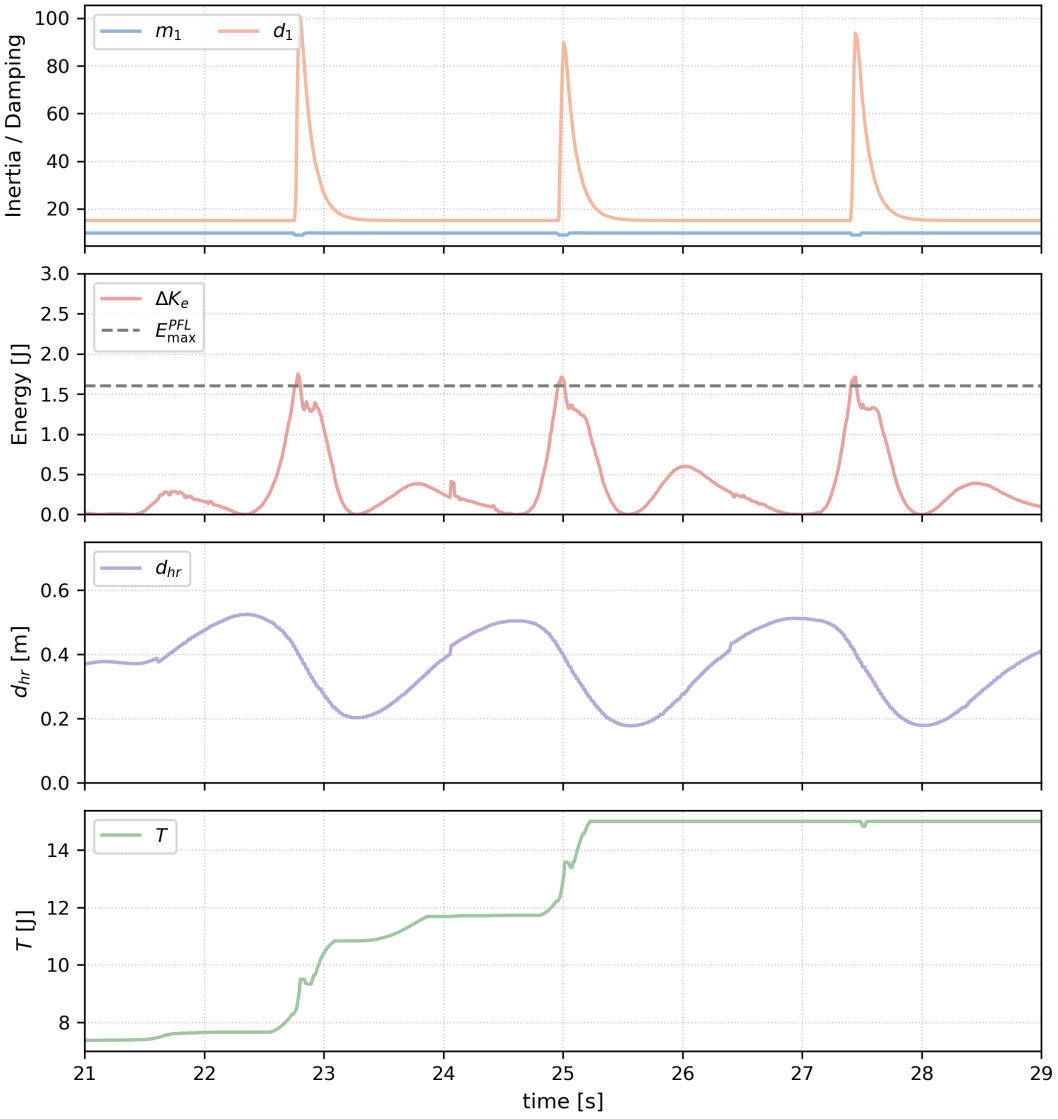


Figure 5.3: Representative time histories for the optimization-based PFL-driven admittance adaptation (zoomed view). From top to bottom: evolution of the first diagonal admittance inertia and damping components (m_1, d_1), PFL feasibility residual h, human–robot distance d_{hr} and energy tank level T. The highlighted interval corresponds to a fast approach motion bringing the system close to the feasibility boundary, without triggering discrete corrective actions.

requirements, computational resources, and acceptable levels of interaction variability.

Chapter 6

Artificial Intelligence for Intuitive Robot Programming

Despite significant advances in robot control, sensing, and physical human–robot interaction, the process of programming industrial and service robots remains a dominant bottleneck in real-world deployment [1, 7]. This bottleneck is not primarily physical, but cognitive: it stems from the mismatch between how humans naturally express task intent and how robots require that intent to be specified in formal, machine-interpretable terms. As a consequence, even robotic systems endowed with advanced compliance, adaptive control, and interaction-aware behaviors remain accessible only to highly trained experts, limiting their applicability in high-mix, low-volume, or rapidly reconfigurable production environments [1].

Traditional robot programming paradigms (such as teach pendant programming, offline programming using CAD-based tools, or script-based interfaces) impose a steep learning curve and require users to reason explicitly in terms of joint configurations, coordinate frames, motion primitives, and vendor-specific abstractions [56]. Several studies have highlighted that these interfaces not only demand specialized expertise, but also introduce significant time overhead during task setup, debugging, and reconfiguration, particularly in scenarios where tasks change frequently or must be adapted on the shop floor [1]. As discussed in the literature on industrial human–robot collaboration, these limitations persist even when physical interaction is available, since lead-through or hand-guiding techniques alone do not eliminate the need for post-processing, validation, and formalization of the taught motion sequences [7].

From a human–robot interaction perspective, this gap can be interpreted as a failure of *intent alignment*: while humans naturally describe tasks at a semantic or goal-oriented level (e.g., "pick the part and place it into the fixture"), robots require low-level, fully specified instructions that encode not only what should be done, but how it should be executed [8, 9]. This disconnect is particularly evident in collaborative and flexible manufacturing contexts, where operators are domain experts in the task, but not necessarily in robotics. As a result, the cognitive load

associated with robot programming becomes a limiting factor for adoption, often outweighing the benefits offered by advanced control and safety mechanisms [1].

Recent research has therefore explored alternative programming modalities aimed at reducing this cognitive barrier, including graphical programming, demonstration-based learning, and multimodal interfaces combining speech, gesture, and vision [7, 57]. While these approaches alleviate some aspects of the problem, they typically remain constrained by rigid interaction grammars or require substantial system-specific customization. Moreover, they often fail to scale beyond narrowly defined tasks, as they lack a general mechanism for interpreting high-level human intent and mapping it onto executable robot behaviors [9].

In this context, large language models (LLMs) have emerged as a promising tool for bridging the semantic gap between human intent and robotic execution [38, 58]. Trained on vast amounts of natural language and code, LLMs exhibit a remarkable ability to parse, contextualize, and structure free-form textual input. This capability has motivated a growing body of work investigating their use as interfaces for robot programming, task specification, and high-level planning [39, 59, 43, 60]. Importantly, and as emphasized in this thesis, the appeal of LLMs in robotics does not lie in their ability to replace control or decision-making algorithms, but in their potential to serve as *cognitive translators*: systems capable of converting human language into structured, machine-interpretable representations of tasks [60].

However, introducing LLMs into robotic systems raises fundamental questions regarding reliability, predictability, and responsibility [20, 42]. Language models are inherently probabilistic, sensitive to prompt formulation, and prone to hallucination, making them unsuitable for direct control or safety-critical decision-making [60]. Consequently, any serious attempt to integrate LLMs into robotic programming must carefully delineate their role and explicitly constrain their authority within the system architecture [43].

This chapter addresses these challenges by framing LLMs strictly as *programming interfaces*, operating at the semantic level and decoupled from execution, control, and safety mechanisms. The focus is not on extending or modifying the safety-aware control architectures developed in Chapters 4 and 5, but on tackling a distinct and complementary problem: how to make robot programming more intuitive, expressive, and accessible, while preserving deterministic execution and limiting the impact of language-model uncertainty [39, 43]. In doing so, the chapter positions language-based programming as an additional interaction modality (orthogonal to physical interaction) aimed at reducing cognitive load rather than enforcing physical safety.

The remainder of this chapter builds on this premise. Section 6.1 formalizes the role of large language models as high-level programming interfaces and explicitly distinguishes them from controllers and planners. Section 6.2 introduces the deterministic behavioral core that mediates between language-derived task representations and robot execution. Section 6.3 discusses semantic grounding through primitive skills and constrained action spaces, while Section 6.4 presents experimental

demonstrations of language-guided robot programming. Finally, Section 6.5 critically examines the limitations and non-claims of the proposed approach.

6.1 Large Language Models as Programming Interfaces, Not Control Systems

The recent surge of interest in large language models (LLMs) within robotics has produced a broad and heterogeneous body of work exploring their integration at different levels of the robotic stack, ranging from natural-language command interfaces to tightly coupled architectures that blend language, perception, planning, and action into a single learning-based pipeline [39, 59, 43, 60, 41]. In many of these approaches, LLMs are implicitly or explicitly entrusted with responsibilities that go beyond semantic interpretation, including task decomposition, plan generation, or even direct action selection [45, 46, 47, 61]. While such systems demonstrate the expressive and compositional capabilities of foundation models, they also introduce significant ambiguity regarding the functional role of the language model within the robotic architecture.

A central concern raised in recent critical analyses is that conflating language understanding with control or decision-making undermines key principles of robotic system design, namely determinism, traceability, and verifiability [20, 42]. Language models are probabilistic systems whose outputs depend on prompt formulation, contextual framing, and sampling strategies, and are therefore inherently non-deterministic [38, 58]. As a result, their direct involvement in control loops, planning pipelines, or safety-related decisions poses substantial challenges in terms of reproducibility, responsibility attribution, and failure analysis [60, 20].

In response to these limitations, several authors have explicitly advocated for architectural separations in which LLMs are treated as high-level interfaces rather than execution authorities [43, 60, 62]. Within this paradigm, the role of the language model is confined to translating human intent (expressed through free-form natural language) into structured, machine-interpretable task representations. These representations are subsequently processed by deterministic software components, such as planners, state machines, or behavior trees, which retain full control over execution [8, 9].

The approach adopted in this chapter follows this line of reasoning and deliberately restricts the LLM to the role of a *programming interface*. The language model operates exclusively at the semantic level and is never granted access to robot state variables, sensor streams, or low-level control interfaces. It does not generate joint trajectories, Cartesian velocities, forces, or torques, nor does it select control gains, thresholds, or safety parameters. Its sole responsibility is to produce structured task descriptions expressed in terms of predefined actions, parameters, and execution logic [39, 59, 43].

This strict scoping distinguishes the proposed framework from end-to-end vision–language–action models and policy-learning approaches that directly map linguistic input to motor commands or learned behaviors [46, 47, 61]. Although such models are attractive from a representation-learning perspective, they remain difficult to reconcile with industrial and collaborative robotics requirements, where system behavior must be explainable, auditable, and bounded by explicit design constraints [1, 7]. In contrast, by preserving a clear separation between intent specification and execution, the present approach aligns with established hybrid robotic architectures that combine symbolic reasoning with deterministic control [8, 9].

Another critical motivation for constraining the role of the LLM lies in the well-documented phenomenon of hallucination. Language models may generate syntactically plausible but semantically invalid or contextually inappropriate outputs, particularly when operating outside narrowly constrained domains [60, 20]. Allowing such outputs to directly influence robot behavior would introduce unacceptable risks. By contrast, when the LLM output is restricted to a symbolic programming layer and validated against predefined schemas and skill libraries, the impact of hallucinations can be significantly mitigated without requiring the language model itself to be reliable [43, 42].

Importantly, confining the LLM to a programming-interface role does not expand the robot's action space. The robot can only execute behaviors that are already implemented and exposed through its primitive skill set, and only according to execution logic defined in deterministic components of the system. The language model cannot invent new skills, synthesize arbitrary control code, or override execution constraints. Consequently, language-based programming modifies how tasks are specified and parameterized, but not what the robot is fundamentally capable of doing [39, 59, 43].

Finally, this architectural choice clarifies the relationship between the present chapter and the safety-oriented control frameworks developed earlier in the thesis. The language-based programming interface introduced here does not reuse, augment, or replace any safety mechanisms, nor does it make claims regarding compliance with industrial safety standards [2, 50]. Its contribution lies entirely at the level of human–robot communication and task specification, addressing the cognitive dimension of robot programming rather than the physical or normative one [43, 20].

In the next section, this separation is operationalized through the introduction of a deterministic behavioral core that mediates between language-derived task representations and robot execution, ensuring that all actions triggered through natural language are executed within a controlled, transparent, and auditable framework.

6.2 AI–ROS2 Interface Design: A Deterministic Behavioral Core Between Language and Action

The central architectural claim of this chapter is that large language models can support intuitive robot programming *only if their role is confined to symbolic specification* and all execution authority remains within deterministic software. This chapter operationalizes this claim by introducing a modular ROS2 architecture in which the LLM produces a structured task plan, while a dedicated *Behavior Core* validates and executes that plan through a finite library of primitives [23, 60, 43]. This architectural separation is therefore treated as a design constraint: the LLM may contribute semantic structure, but only deterministic software may commit actions to the robot [58, 20, 42].

6.2.1 Module Decomposition and ROS2 Communication Contract

The architecture comprises four explicit modules: *Perception*, *User Input / ChatGPT Node*, *Behavior Core*, and *Robot Interface* (Fig. 6.1). Their separation is deliberately chosen so that each module has a narrow contract, and failure can be localized to a specific interface rather than propagating implicitly across the stack [23].

Perception module (open-vocabulary object state). The perception module provides the structured scene information required by the Behavior Core to bind symbolic task plans to concrete objects in the workspace. In the implementation adopted in this chapter, perception is built around *YOLO-World*, an open-vocabulary object detection framework that extends the YOLO family by enabling *language-conditioned* detection: rather than restricting recognition to a fixed closed set of classes, the detector can be queried at runtime with textual labels (e.g., "red box", "battery", "tool") and returns bounding boxes with associated confidence scores for the queried concepts [63]. Conceptually, this shifts the perception interface from "choose among predefined classes" to "request the objects referenced in the task description," thereby supporting flexible task specification without requiring task-specific retraining or a frozen object taxonomy.

To enable downstream deterministic execution, the output of this perception stage is not treated as raw visual data (images, feature maps) but is converted into a *structured world state* published over ROS2. This world state contains semantic identifiers, confidence values, and associated 3D coordinates (e.g., obtained via depth sensing or geometric back-projection), and is encapsulated in a dedicated message type (`WorldState.msg`) consumed by the Behavior Core. In this way, open-vocabulary perception expands the label space that the user can refer to in language,

while the Behavior Core still interacts only with explicit symbolic variables and typed messages, preserving modularity and traceability at execution time.

User Input and ChatGPT Node (language-to-plan compilation). The user specifies the task via natural language (e.g. a text interface). The instruction is sent to the ChatGPT API together with an augmented prompt that constrains the model output to a prescribed JSON structure. The output is not executed directly; it is parsed into a structured task-plan message (denoted `TaskPlan.msg`) and forwarded to the Behavior Core. This node can be viewed as a compiler front-end: it translates linguistic intent into a symbolic intermediate representation that is syntactically constrained but still semantically uncertain, hence requiring downstream validation [58, 20]. In this implementation, the LLM is accessed as an external service via API calls, while all validation and execution remain local within the ROS2 system.

Behavior Core (execution authority). The Behavior Core is the authoritative layer of the architecture. It receives (i) `TaskPlan.msg` from the ChatGPT Node and (ii) `WorldState.msg` from Perception, and it converts them into validated primitive calls dispatched to the Robot Interface. In contrast with LLM-centric approaches that blur planning and execution, the Behavior Core is designed to preserve determinism through explicit validation, explicit state management, and event-driven control logic implemented in conventional software modules rather than in the language model [8, 9, 43].

Robot Interface (primitive implementation). The Robot Interface consists of one or more ROS2 nodes that implement the primitive functions referenced in the task plan (e.g., `move_to`, `grab`, `release/place`, and auxiliary computation primitives such as offset computation). The key invariant is that the interface exposes a *closed set* of primitives: the LLM cannot synthesize new primitives, bypass the Behavior Core, or inject arbitrary control code. This is what makes the system "safe-ish" in the narrow sense of limiting hallucinations: even a perfectly fluent but incorrect language output cannot extend the robot action repertoire beyond what the deterministic system already implements [43, 42].

6.2.2 Internal Structure of the Behavior Core

The Behavior Core is not monolithic; it is implemented as a modular execution pipeline comprising three submodules: (*Behavior Executor*, *Buffer Manager*, and *Trigger Evaluator*) whose separation is essential for transparency and predictable behavior (Fig. 6.2) [8, 9].

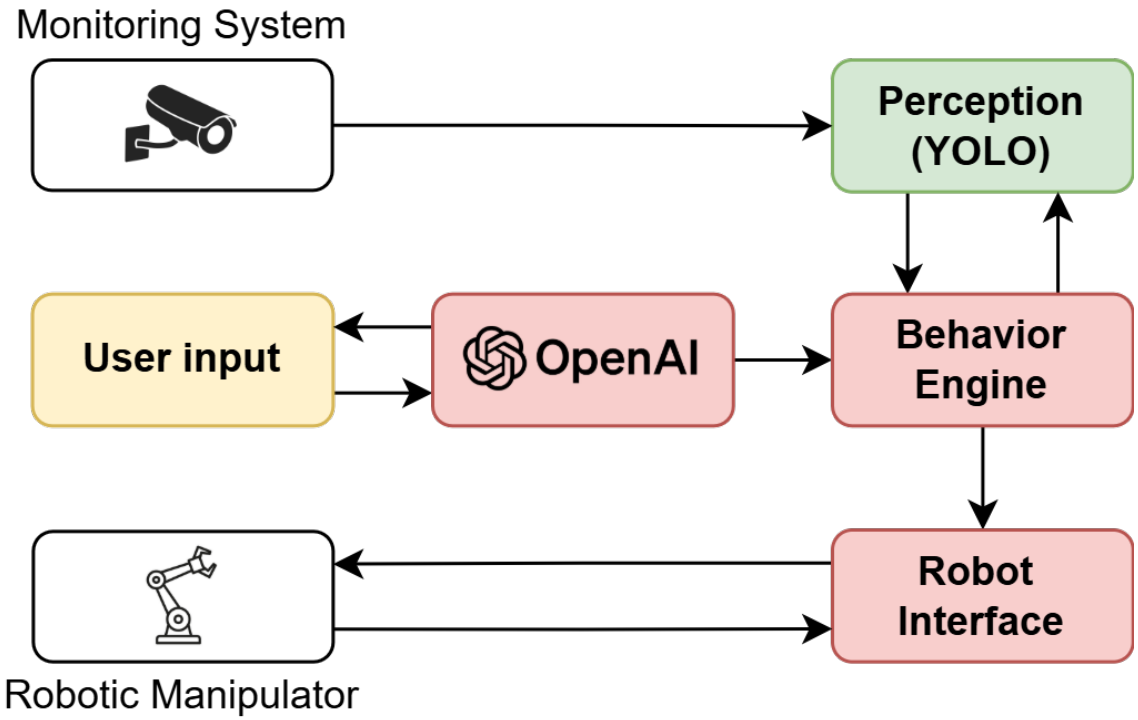


Figure 6.1: Overall ROS2 architecture for language-based robot programming, highlighting the separation between LLM plan generation, deterministic validation/execution, and primitive actuation.

Behavior Executor. The Behavior Executor interprets each primitive object in the task plan and dispatches the corresponding execution request to the Robot Interface. It also interfaces with Perception during execution when verification is required, for example to confirm the presence or pose of an object before committing to the next command. Crucially, these queries do not regenerate the high-level plan; they only confirm preconditions for executing the already validated plan structure [43].

Buffer Manager. The Buffer Manager maintains a short-term memory of symbolic variables and execution-relevant state across multi-step actions. This includes lists of detected object poses, intermediate computed poses (e.g., safe offsets), and execution logs that ensure internal consistency across a repeated cycle. In other words, it is the mechanism by which the system retains coherence across a multi-primitive plan without delegating memory or bookkeeping to the language model [43].

Trigger Evaluator. The Trigger Evaluator implements the event-driven loop semantics encoded in the task plan: after a cycle completes, it evaluates a logical condition over the buffered state (e.g., whether any objects remain to be processed) and determines whether the repeat block should execute again or terminate. This yields a lightweight event-driven execution mechanism that enables reactivity to

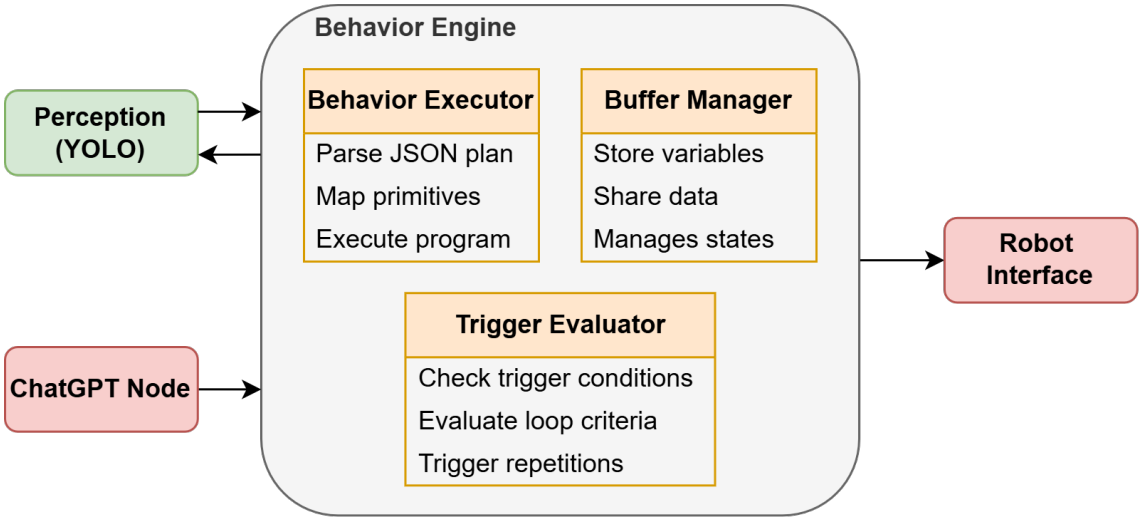


Figure 6.2: Behavior Core and its interfaces, including submodules responsible for primitive dispatch, state buffering, and trigger evaluation.

perception updates without forcing the LLM to continuously replan [43].

6.2.3 Discrete Perception–Execution Cycles and Deterministic Flow

A subtle but crucial feature of the architecture is that it operates in discrete perception–execution cycles. Execution proceeds by consuming a validated primitive sequence; once the sequence (or a cycle) completes, the perception module rescans the workspace to update `WorldState.msg`, which is then used to evaluate the trigger condition for the next iteration. This design choice yields two properties that are easily lost in overly generic descriptions: (i) the Behavior Core remains a deterministic interpreter of a symbolic plan, and (ii) the system can incorporate updated perception between cycles without requiring language-model re-generation of the plan [43].

This cycle-based execution logic is also what makes responsibility attribution meaningful: any executed motion primitive can be traced back to (a) a specific element in `TaskPlan.msg` and (b) a specific deterministic decision of the Behavior Core (buffer update, trigger evaluation, or executor dispatch). This traceability is precisely what the architecture is designed to preserve when incorporating probabilistic language parsing upstream [20, 42].

6.2.4 Limitations due to perception uncertainty

The deterministic nature of the Behavior Core guarantees that, given a valid semantic description of the environment, the executed robot behavior is predictable and reproducible. However, this determinism does not mitigate errors originating

from upstream perception modules. In particular, if the vision system incorrectly classifies an object (for example, misinterpreting a human hand as a manipulable workpiece), the deterministic execution layer would process this incorrect semantic input and generate the corresponding manipulation command. Consequently, the safety properties of this architecture differ fundamentally from those provided by the energetic guarantees of the control framework presented in Chapter 5. In the present AI-assisted programming architecture, safety must therefore be interpreted as probabilistic and dependent on the reliability of the perception system and the redundancy of the sensing pipeline.

6.3 Semantic Understanding and Context Awareness via Structured Task Plans

In this chapter, "semantic understanding" does not mean that the robot acquires an internal language grounding comparable to human semantic competence; rather, it denotes a *controlled translation* from natural language to a bounded symbolic representation that can be executed deterministically. This is achieved by requiring every LLM output to conform to a JSON schema and by treating the resulting JSON object as an intermediate representation (`TaskPlan.msg`) validated and executed by the Behavior Core. This section explicates that representation in detail because it is the technical mechanism that makes the overall approach viable [43, 60].

6.3.1 TaskPlan JSON Schema: Fields, Roles, and Execution Semantics

The task plan is a hierarchical JSON object comprising: (i) an identifier, (ii) a repetition flag, (iii) a trigger condition governing termination, and (iv) two ordered lists of actions: initialization actions executed once and repeat actions executed cyclically. This structure is more expressive than a flat sequence of primitives because it supports iterative, event-driven behaviors while remaining simple enough to validate robustly [43].

Top-level keys. The top-level schema includes the following keys:

- `behavior_id`: a string identifier used for traceability, debugging, and logging.
- `repeat`: a boolean enabling cyclic execution.
- `on_trigger`: an object encoding a logical condition (e.g., over perceived object lists) used by the Trigger Evaluator to determine whether to continue the cycle.

- `init_actions`: a list of primitives executed once before the loop begins (often used to compute auxiliary positions such as pallet layouts or offsets).
- `repeat_actions`: the sequence of primitives that defines the cyclic behavior of the task, executed repeatedly until the trigger condition becomes false.

Primitive action objects. Each action element is itself a structured object that minimally specifies a `function` field naming a primitive implemented by the Robot Interface, and may include optional fields that enable symbolic data flow through the Buffer Manager [43]. This work highlights three particularly important mechanisms:

- `params`: a dictionary of primitive parameters. A key pattern is `from`, which binds the primitive to a buffered variable (e.g., a list of object poses).
- `store_as`: a symbolic variable name into which the primitive output is stored for reuse by subsequent primitives (e.g., saving an "offset" pose as a safe approach point).
- `use_next`: a boolean flag indicating that the primitive should consume the next element of a list-valued variable (e.g., pop the next detected box pose from `box_positions`).

These fields implement a constrained form of program state without allowing arbitrary computation: the plan expresses data dependencies only through variable names and list-consumption flags, while all computation occurs inside deterministic primitives (e.g., offset computation, pallet planning) implemented outside the LLM [43].

6.3.2 Representative Plan Instance (Palletization Loop)

The schema is best understood through a representative example (Listing 6.1), which encodes a palletization loop. Here, the user instruction (e.g., "palletize all boxes") is translated into (i) an initialization block that generates pallet target poses and (ii) a repeat block that iteratively picks and places objects while boxes remain detected [43].

What this plan encodes (and what it deliberately forbids). This plan is simultaneously expressive and restrictive. It is expressive because it encodes a task-level loop with a termination condition driven by perception (`len(box_positions) >= 1`), as well as a structured sequence with intermediate symbolic variables (`pick_safe`, `place_safe`). It is restrictive because every computational operation is delegated to deterministic primitives (`plan_pallet`, `offset_position`) and every physical command is mediated through a finite action set (`move_to`, `grab`, `place`). The LLM does not provide control laws, does not compute kinematics, and does not implement perception; it only emits a schema-conformant plan that references these capabilities.

Listing 6.1: Representative TaskPlan JSON structure produced by the LLM and validated by the Behavior Core (palletization loop).

```
{
  "behavior_id": "palletize_loop",
  "repeat": true,
  "on_trigger": { "condition": "len(box_positions) >= 1" },
  "init_actions": [
    { "function": "plan_pallet", "store_as": "pallet_positions"
       }
  ],
  "repeat_actions": [
    { "function": "offset_position",
      "params": { "from": "box_positions", "dz": 0.10 },
      "store_as": "pick_safe" },

    { "function": "move_to", "params": { "from": "pick_safe" }
     },
    { "function": "move_to", "params": { "from": "box_positions
     ", "use_next": true } },
    { "function": "grab" },

    { "function": "offset_position",
      "params": { "from": "pallet_positions", "dz": 0.10 },
      "store_as": "place_safe" },

    { "function": "move_to", "params": { "from": "place_safe" }
     },
    { "function": "move_to", "params": { "from": "
      pallet_positions", "use_next": true } },
    { "function": "place" }
  ]
}
```

6.3.3 Validation Contract and Failure Containment

Because the plan is generated by a probabilistic model, validation is not optional: it is the mechanism that turns "LLM output" into an admissible program within a closed-world robotic system. The Behavior Core validates both *syntax* (schema conformity) and *semantic admissibility* (primitive closure and parameter integrity) before dispatching any action. This represents the key barrier preventing hallucinated plans from becoming executed motions [58, 20, 42].

At minimum, the validation contract must enforce:

- **Primitive closure:** `function` must belong to the Robot Interface primitive set.
- **Typed structure:** `repeat` is boolean; `init_actions` and `repeat_actions` are lists of objects; `on_trigger` is a condition object.
- **Reference integrity:** variables referenced in `params.from` must exist in the Buffer Manager state or be defined by prior `store_as` bindings.
- **Trigger admissibility:** `on_trigger.condition` must be evaluable by the Trigger Evaluator within a restricted predicate language (e.g., bounded boolean expressions over list sizes and simple comparisons), rather than arbitrary code execution.

Context awareness as bounded fusion. Finally, this chapter's notion of "context awareness" is not a claim about full multimodal reasoning. It is the controlled fusion of (i) a structured perceptual world state from YOLO-World and (ii) a structured symbolic plan from the LLM. The Behavior Core is where these two are combined: it buffers scene variables, checks triggers, and can request perception rescans when required to confirm preconditions. This yields a pragmatic form of context sensitivity while preserving the fundamental constraint that plan authority and execution authority remain distinct [43].

6.4 Experimental Demonstrations of Language-Guided Robot Programming

The experimental activity presented in this section aims to demonstrate the feasibility and practical relevance of language-based robot programming when large language models are employed strictly as programming interfaces. The objective of the experiments is not to evaluate control performance, physical interaction quality, or safety guarantees, but to assess whether natural language can be effectively translated into structured task specifications that are correctly interpreted and

executed by a deterministic robotic system. In line with the scope of this chapter, all experiments focus on cognitive accessibility, execution transparency, and robustness at the interface level [39, 43].

The experimental framework follows the architectural principles introduced in the previous sections. Natural language commands provided by the user are processed by a large language model, which generates a structured task representation compliant with a predefined schema. This representation is subsequently validated and executed by the behavioral core, which orchestrates the execution of predefined primitive skills. At no point does the language model interact directly with low-level control, sensing, or actuation, nor does it participate in real-time decision-making [60, 20].

6.4.1 Experimental Setup and Execution Pipeline

The experimental setup consists of a robotic manipulator equipped with a set of pre-implemented primitive skills, including basic motion primitives and manipulation actions. These skills are exposed to the behavioral core through a deterministic execution interface, while the language model operates externally as a semantic interpreter. Communication between the language interface, the behavioral core, and the robot execution layer is implemented using a modular software architecture, ensuring clear separation of responsibilities and reproducible execution semantics [23, 43].

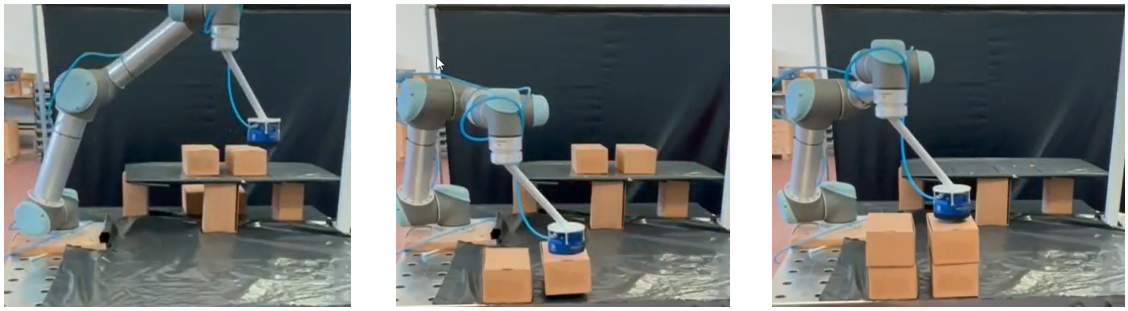
Figure 6.1 summarizes the system architecture adopted in the experiments, from language parsing to deterministic execution through the Behavior Core. A natural language instruction is translated into a schema-conformant task description, validated against the primitive library and parameter constraints, and then executed as a deterministic sequence by the Behavior Core; any invalid or ambiguous model output is rejected during validation and never propagates to actuation.

6.4.2 Representative Language-Guided Programming Scenarios

The experiments include representative task scenarios designed to reflect common programming needs in industrial and laboratory contexts, including pick-and-place operations, object relocation, and simple inspection or positioning tasks. Two illustrative examples are reported in Fig. 6.3 and Fig. 6.4. In the first, the user issues a natural-language instruction to *palletize all boxes in the scene*; in the second, the user requests the system to *sort the boxes by color*. In both cases, commands are intentionally expressed in unconstrained language, allowing the user to specify task intent at a semantic level without explicitly defining robot-centric details such as joint configurations, reference frames, waypoints, or motion parameters [39, 59].

Across all scenarios, the language model successfully generates structured task

representations that are compatible with the behavioral core, provided that the requested actions fall within the available primitive skill set. When commands exceed system capabilities or reference unsupported actions, the framework fails gracefully by rejecting the task or requesting clarification, rather than attempting unsafe or undefined execution. This behavior highlights the importance of constrained action spaces and deterministic validation in mitigating the impact of language-model uncertainty [43, 42].



(a) Two boxes are placed in the picking area; the robot receives the natural-language instruction to palletize all detected boxes.

(b) Two additional boxes are placed while the robot is operating; the system rescans the scene at the next perception–execution cycle and updates the buffered object list accordingly.

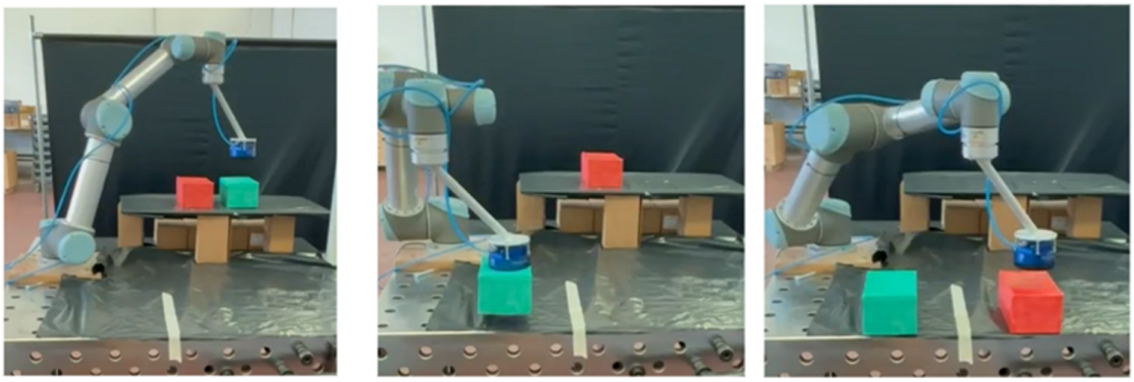
(c) All four boxes have been palletized according to the LLM-generated plan and Behavior Core execution cycle.

Figure 6.3: The system receives a natural-language command to palletize the boxes on the table. As additional boxes appear during execution, the YOLO-World perception module updates the world state at the next perception–execution cycle and the Behavior Core extends the loop accordingly, resulting in all four boxes being successfully palletized.

6.4.3 Qualitative Evaluation and Observations

The primary outcome of the experimental activity is qualitative rather than quantitative. From a usability perspective, language-based programming significantly reduces the cognitive burden associated with task specification, allowing users to express intent directly without intermediate translation into low-level robot commands. This observation is consistent with findings reported in recent studies on natural-language interfaces for robotics, which emphasize improvements in accessibility and task formulation efficiency [57, 64].

From a system perspective, every executed motion can be traced from a specific JSON plan element (Listing 6.1), through a specific primitive invocation, to the corresponding execution log produced by the Behavior Core, which supports debugging and post hoc accountability without requiring interpretability of the language model itself [8, 20].



(a) Two boxes of different colors are placed in the picking area; the robot receives the natural-language instruction to sort all detected boxes based on the color.

(b) The robot sorted one of the boxes in the corresponding area.

(c) All boxes have been sorted according to the LLM-generated plan and Behavior Core execution cycle.

Figure 6.4: The system receives a natural-language command to sort the boxes on the table into two corresponding areas.

It is important to emphasize that no claims are made regarding robustness to adversarial prompts, generalization beyond the tested scenarios, or resilience to language-model failure modes. The experiments are intentionally scoped to illustrate feasibility and architectural soundness rather than completeness or optimality [60]. In particular, the absence of safety supervision, force limitation, or interaction-aware control in these experiments reflects a deliberate design choice aligned with the scope of this chapter.

6.4.4 Discussion of Scope and Limitations of the Experimental Validation

The experimental demonstrations presented here should be interpreted strictly as evidence that language-based programming can be integrated into robotic systems without compromising deterministic execution, provided that the language model is properly constrained. They do not constitute a validation of autonomous task execution, nor do they imply that natural language interfaces are suitable for unstructured or safety-critical environments [20, 42].

By confining the role of the language model to semantic interpretation and grounding all execution in predefined skills, the experiments illustrate a pragmatic pathway for introducing LLM-based interfaces into robotics without overextending their capabilities. This perspective aligns with recent calls for conservative and transparent integration strategies when deploying foundation models in embodied systems [43, 46].

The following section concludes the chapter by explicitly articulating the limitations, risks, and non-claims associated with the proposed approach, reinforcing the distinction between cognitive accessibility and physical safety that underpins the entire contribution of this chapter.

6.5 Limitations, Risks, and Explicit Non-Claims

The language-based robot programming framework presented in this chapter is intentionally scoped and conservative. While it demonstrates the feasibility and practical utility of large language models as programming interfaces, it also exhibits fundamental limitations that must be acknowledged explicitly. These limitations are not incidental shortcomings of the implementation, but rather structural consequences of the design choices adopted to preserve determinism, transparency, and responsibility attribution within the robotic system [20, 42].

A first and central limitation concerns safety. The framework described in this chapter does not implement, reuse, or enforce any physical safety mechanisms, nor does it provide guarantees with respect to human–robot interaction safety, collision avoidance, force limitation, or compliance with industrial safety standards [2, 50]. No assumptions are made regarding speed and separation monitoring, power and force limiting, or any other normative safety paradigm. As a consequence, the proposed approach must not be interpreted as a safety-certified solution, nor as a replacement for established safety architectures developed elsewhere in this thesis or in the broader robotics literature [1].

The use of the term “safe-ish” throughout this chapter refers exclusively to architectural risk mitigation at the programming-interface level. Specifically, it denotes the reduction of semantic and execution-level risks arising from unconstrained natural language input, achieved through the use of primitive skills, constrained action spaces, and deterministic validation layers. This notion does not imply physical safety, normative compliance, or protection against harmful interactions, and should not be conflated with certified safety concepts used in industrial or collaborative robotics [20, 43].

A second limitation concerns the inherent properties of large language models. LLMs are probabilistic systems whose outputs are sensitive to prompt formulation, contextual framing, and decoding strategies [38, 58]. They may produce hallucinated, inconsistent, or contextually inappropriate responses, particularly when operating outside narrowly constrained domains [60]. While the proposed framework mitigates the impact of such failures by restricting the authority of the language model, it does not eliminate them. As a result, language-based programming remains vulnerable to upstream misinterpretations that must be handled through validation, user supervision, and conservative system design [42].

Another important limitation lies in the expressiveness of the action space.

By design, the robot can only execute behaviors that are already implemented as primitive skills. While this constraint is essential for preserving determinism and transparency, it also limits the range of tasks that can be specified through natural language. The framework does not support the synthesis of novel skills, the generation of new control policies, or autonomous learning from language input [39, 59]. Consequently, the system should be understood as a programming aid rather than an autonomy-enhancing mechanism.

From an evaluation standpoint, the experimental results presented in this chapter are qualitative and illustrative. They demonstrate feasibility and architectural coherence, but do not constitute a comprehensive assessment of usability, robustness, or scalability. No user studies are conducted to quantify reductions in programming time or cognitive load, and no benchmarks are provided to compare the proposed approach against alternative programming interfaces [64]. These aspects represent important directions for future investigation, but fall outside the scope of the present work.

It is also important to emphasize that the framework does not address broader ethical, legal, or organizational implications associated with the deployment of large language models in industrial settings. Issues such as accountability for errors, intellectual property concerns, data privacy, and long-term system maintenance are acknowledged but not resolved here [20]. Addressing these dimensions would require interdisciplinary analysis beyond the technical focus of this thesis.

Finally, the chapter advances several explicit non-claims that are critical to its correct interpretation. The proposed approach does not claim to enable autonomous robot behavior, does not claim robustness to adversarial or malicious inputs, and does not claim suitability for safety-critical or unstructured environments. It does not assert compliance with any industrial safety standard, nor does it suggest that natural language interfaces can replace traditional engineering practices in robot programming. Instead, the contribution of this chapter lies in demonstrating that, when carefully constrained, large language models can serve as effective programming interfaces that reduce cognitive barriers without compromising deterministic execution [43, 60].

By articulating these limitations and non-claims explicitly, this chapter positions language-based robot programming as a complementary interaction modality rather than a disruptive replacement for existing control and safety frameworks. The results presented here should therefore be interpreted as a step toward more accessible and expressive robot programming interfaces, grounded in architectural rigor and epistemic humility, rather than as an endpoint in the pursuit of intelligent or autonomous robotic systems.

Chapter 7

Results, Discussion, and Synthesis

This chapter performs the integrative work that the preceding chapters deliberately postpone. Chapters 3–6 introduced three programming modalities (physical lead-through on an industrial robot, PFL-oriented physical interaction in collaborative settings, and language-based programming mediated by a Large Language Model (LLM)) because they embody three distinct answers to the same research question: *how can robot programming be made substantially easier for non-expert users without quietly weakening the safety and interpretability assumptions that industrial deployment requires?*

The key claim defended here is that the dissertation is unified not by a single algorithm reused everywhere, but by a consistent logic of *claims and guarantees* across programming modalities. This chapter therefore (i) consolidates empirical outcomes, (ii) compares the approaches along shared axes, (iii) isolates the theoretical and practical contributions, and (iv) states limitations that delimit what is (and is not) claimed.

7.1 Summary of Experimental Outcomes

A recurring difficulty in synthesizing results across modalities is that each modality is evaluated with metrics that are native to its safety logic: torque reconstruction error for identification, distance-based indicators for SSM-inspired supervision, residual-based energy inequalities for PFL, and plan validity together with qualitative execution demonstrations for LLM-mediated programming. Rather than forcing these into a single scalar score (which would be methodologically misleading), this section summarizes outcomes at the level of *validated claims*: what each stage demonstrates, under which assumptions, and with what observable evidence.

Throughout this chapter, "validation" is used in a modality-dependent sense: identification is validated against measured torque signals, physical-interaction safety

reasoning is validated through satisfaction of the corresponding supervisory or PFL constraints in experimental trials, and LLM-based programming is validated through plan admissibility and deterministic executability rather than physical safety certification.

7.1.1 Dynamic identification outcomes: accuracy as an epistemic instrument, not a control dependency

Chapter 3 establishes an identified rigid-body model whose purpose in the dissertation is *analytical*: it supports physical interpretation, torque-level validation, and safety-relevant reasoning (e.g., torque- and power-related diagnostic quantities), while remaining explicitly *non-functional* with respect to the admittance controller. In particular, the admittance-based programming methods of Chapters 4–5 remain operationally independent of this model; the model is instead used to quantify how the system behaves and why certain safety margins shrink or expand under different dynamic conditions.

Empirically, model fidelity is validated through torque reconstruction. Static validation yields joint-wise errors summarized in Table 3.1, with larger residuals concentrated in the proximal joints where coupling and unmodeled mechanisms are most influential. Dynamic validation, shown qualitatively in Fig. 3.3, demonstrates that the reconstructed torque profiles track measured signals with consistent phase and amplitude trends across excitation segments. The interpretive value of these results is twofold: they justify using the model as a diagnostic reference.

7.1.2 Industrial lead-through outcomes: distance-supervised compliance with energy-consistent adaptation

Chapter 4 demonstrates that physical lead-through programming on a non-collaborative industrial manipulator can be made substantially more accessible by embedding a compliant interaction layer (variable/adaptive admittance) while supervising motion through a *distance-inspired* safety logic. The key outcome is not the claim of normative compliance, but the demonstration of a coherent architecture in which (i) user guidance is interpreted through a compliant dynamic relation, (ii) safety is enforced through supervision and constraint-based modulation, and (iii) parameter adaptation is constrained by passivity-inspired mechanisms intended to preserve energetic consistency under time-varying admittance.

The framework is *SSM-inspired* in the limited sense that distance-to-human information is used to modulate compliance and motion supervision; it is *not* presented as ISO-certified SSM, nor as a replacement for a certified safety controller.

7.1.3 Collaborative interaction outcomes: PFL safety reasoning with energy-based variable admittance

Chapter 5 shifts the safety paradigm from distance supervision to *Power and Force Limiting* (PFL), where the core question is not separation but contact tolerance under bounded transferred energy. The central empirical outcome is that variable/adaptive admittance can be used as a *safety-relevant knob* because the desired inertia shapes the apparent mass along the approach direction, and thereby constrains the admissible velocity under a fixed energy limit. In the PFL formulation adopted throughout the thesis, safety is monitored through a residual quantity h that compares predicted transferred energy against the limit $E_{\max}^{PFL}$, and parameter updates are selected (algorithmically or via optimization) to keep the system on the safe side of the inequality.

A crucial practical outcome is that safety-aware adaptation is not merely reactive damping injection after violation, but can be posed as a *one-step predictive* update selection problem: the controller evaluates whether the *post-update* state would violate the PFL constraint, and rejects/adjusts candidate parameters accordingly (cf. the predictive residual construction used in the optimization-oriented formulation). This anticipatory logic is precisely what enables variable admittance to be safety-oriented rather than purely performance-oriented.

7.1.4 LLM-based programming outcomes: bounded action space and deterministic execution, not certified safety

Chapter 6 provides a third modality of programming accessibility: the user expresses intent in natural language, while execution is mediated by a structured interface that explicitly separates natural-language specification from deterministic execution. The LLM (ChatGPT Node) is treated as a compiler from language to a constrained task plan; it is neither a controller nor an online planner. Execution authority is assigned to the deterministic Behavior Core, which enforces primitive closure, typed structure, reference integrity, and admissible triggers before any motion primitive can be invoked.

Here, "validation" refers to structural and semantic admissibility of the generated plan with respect to the closed primitive set and trigger language, rather than validation of physical safety properties in the ISO sense.

The empirical and architectural outcome to retain for synthesis is therefore not a safety claim in ISO terms, but a *boundedness claim*: the system is "safe-ish" only insofar as it restricts what the LLM can ask the robot to do and validates plans before execution. This is a different kind of guarantee from SSM/PFL: it is a guarantee about the *space of commands* and the *structure of execution*, not about human injury thresholds or certified safety functions.

Table 7.1: Empirical outcomes and validation evidence across programming modalities.

Modality	**Observed quantity / property**	**Validation evidence**
Identification	Joint torque reconstruction accuracy (RMSE_i)	Static and dynamic torque comparisons (Ch. 3)
Industrial lead-through	Distance-based safety indicators; compliant response to guidance	Proximity experiments and constraint activations (Ch. 4)
Collaborative interaction (PFL)	Energy residual h; admissible velocity envelopes	PFL-constrained interaction trials (Ch. 5)
LLM-based programming	Primitive closure; plan validity; deterministic execution flow	Schema validation, execution logs, task demonstrations (Ch. 6)

Table 7.1 summarizes these outcomes in a modality-agnostic format: it records what was empirically observed and what artifact substantiates the claim, without forcing different paradigms into a single scalar metric. This summary then becomes the input to the comparative analysis in Section 7.2, where the modalities are contrasted in terms of guarantees, assumptions, and failure modes.

7.2 Comparative Analysis

A meaningful comparison across Chapters 3–6 requires resisting an attractive but incorrect framing: that the thesis progresses from "model-based control" to "AI control." The dissertation does not argue for replacing physics with learning. Instead, it argues for *stacking programming modalities* while preserving explicit boundaries between (i) physical interaction laws, (ii) safety paradigms, and (iii) execution authority.

Across these modalities, the unifying architectural stance is to separate the human-facing interface (physical guidance or language) from the execution authority, and to make explicit what is bounded and under which safety paradigm the bound is interpreted.

7.2.1 Shared axes of comparison

To prevent "apples vs. oranges" objections, we compare modalities along four axes:

Table 7.2: Comparative matrix across modalities using the shared axes defined in Section 7.2.1.

Axis	**Industrial lead-through**	**Collaborative PFL**	**LLM programming**
Intent channel	Physical guidance	Physical interaction	Natural language
Bounded quantity	Distance-supervised modulation of motion/compliance	Transferred energy / residual	Primitive/action space
Core guarantee	Conservative supervision of compliant motion	Energy-based contact-tolerance reasoning	Deterministic execution + validation
Primary risk	Sensing uncertainty; unmodeled effects	Feasibility under real-time constraints	Miscompiled plans; perception mismatch
Explicit non-claim	ISO-certified SSM	Universal safety beyond model assumptions	ISO safety / SSM / PFL compliance

1. **Where intent enters**: physical guidance forces/velocities vs. language.
2. **What is bounded**: distance-conditioned modulation of motion/compliance vs. transferred energy (PFL) vs. discrete primitive/action space.
3. **What the controller assumes**: (no dynamic model dependency) vs. energy limit tables and approach-direction projections vs. typed schemas and validated triggers.
4. **Failure modes**: unmodeled dynamics and sensing uncertainty vs. residual conservatism and real-time feasibility vs. semantic miscompilation and perception mismatch.

7.2.2 What "safe and easy programming" means in this dissertation

The comparative matrix clarifies an important conceptual point: in this thesis, "safe" is not a single predicate. It is a family resemblance term whose meaning depends on which part of the stack is being discussed. In the physical-interaction modalities, safety is tied to physically grounded quantities (distance margins, transferred energy bounds) and is evaluated in continuous time. In the language-based modality, safety is tied to *command admissibility* and *execution structure*; it is evaluated in discrete

cycles with a deterministic executive. The dissertation's coherence depends on making this polysemy explicit rather than trying to collapse it.

7.3 Theoretical and Practical Contributions

The contributions of the dissertation are best stated as contributions to a design methodology for HRI programming, not as isolated algorithmic novelties. The thesis contributes:

7.3.1 Structured parameter identification via static–dynamic separation

A first methodological contribution of the dissertation lies in the structured treatment of dynamic parameter identification, where static and dynamic effects are explicitly separated rather than estimated through a single monolithic regression. Instead of treating gravity, friction, and inertial parameters as a homogeneous parameter vector, the identification procedure isolates static contributions (gravity and friction) from velocity- and acceleration-dependent dynamics, and estimates them through excitation trajectories tailored to their physical role.

This separation is not merely a numerical convenience. It reflects a physical modeling stance in which different classes of parameters are associated with different observability conditions, noise sensitivities, and validation criteria. Static terms are identified under quasi-static conditions where inertial effects are negligible, while dynamic parameters are identified using persistently exciting motions designed to expose inertial coupling. As a result, the identified model supports meaningful physical interpretation and joint-wise diagnostic analysis, rather than acting as a black-box torque predictor.

Crucially, this identification structure aligns with the broader philosophy of the dissertation: models are introduced to *explain* and *validate* robot behavior, not to serve as hidden dependencies of interaction control laws. By separating static and dynamic effects at the identification level, the resulting model becomes a transparent analytical reference for subsequent chapters, supporting energy estimation, safety-related reasoning, and experimental interpretation without imposing unrealistic accuracy requirements on the control architecture.

7.3.2 Energy-consistent adaptation: variable admittance as a safety-relevant degree of freedom

A second contribution is the systematic use of energy-based reasoning to enable time-varying admittance without destabilizing energy injection. In the industrial case, this appears as the insistence that adaptation must not break energetic consistency when parameters change online. In the collaborative PFL case, this is sharpened into a safety-oriented view of admittance: since desired inertia shapes apparent mass, and apparent mass shapes admissible velocities under a fixed $E_{\max}^{PFL}$, parameter adaptation is not merely "feel tuning" but a lever that trades responsiveness against bounded collision energy. The dissertation's contribution is to make that coupling operational in real-time adaptation laws (algorithmic and optimization-based) while retaining passivity-inspired constraints.

7.3.3 Bounded autonomy for language programming: deterministic execution as the safety-relevant boundary

A third contribution is architectural: the LLM component is intentionally demoted from executive authority to a constrained plan generator, while the Behavior Core enforces correctness properties that are checkable (schema validity, typed parameters, reference integrity, restricted triggers, and primitive closure). This yields a principled "safe-ish" interface: it does not claim physical safety compliance, but it does claim that the LLM cannot directly issue arbitrary motion commands, and that execution is mediated by a closed set of validated primitives. The contribution is therefore not "LLMs control robots," but "LLMs can be used as programming interfaces when the execution boundary is deterministic and formally checkable."

7.4 Limitations and Lessons Learned

The thesis makes progress by narrowing claims to what is supportable; this section states the remaining gaps that delimit generalization.

7.4.1 Limits of distance-based supervision in industrial lead-through

Distance-aware supervision is only as credible as the sensing and geometric abstraction that supports it. Occlusions, latency, conservative geometric proxies, and calibration drift all tend to bias the system either toward unsafe over-optimism (unacceptable) or toward excessive conservatism (reduced usability). The main lesson is that, in an

industrial context, supervision-based approaches should be framed as *architecture-compatible* strategies that can enhance safe interaction, but they should not be rhetorically conflated with certified safety functions unless the complete safety chain is implemented and validated.

7.4.2 Limits of PFL reasoning under modeling and contact assumptions

Energy-based PFL reasoning relies on assumptions about equivalent human mass, effective stiffness, contact area, and the relevance of approach-direction projections. These assumptions are defensible as engineering abstractions and align with the logic of ISO/TS-like tabulations, but they are not universally precise for all contacts, postures, tools, or transient impacts. The lesson is that PFL safety margins should be treated as *structured conservatism*: valuable for shaping interaction envelopes, but necessarily dependent on anatomical/contextual parameterization and on the quality of velocity estimation.

7.4.3 Limits of LLM "safe-ish" programming

The language-based programming pipeline is intentionally not framed as physically safe. Its main limitation is that bounded primitives and validation constrain *what* can be executed, but do not certify *how safely* execution unfolds in human proximity. Moreover, plan validity does not imply semantic correctness: a schema-valid plan may still be contextually wrong when perception is uncertain or when the user instruction is underspecified. The principal lesson is that language interfaces become credible in robotics not by inflating autonomy, but by strengthening execution boundaries, observability, and failure handling.

In combination, these limitations motivate the concluding chapter: future research must focus on bridging these modalities without collapsing their guarantees, i.e., integrating richer safety monitors and predictive reasoning into execution cores while preserving the architectural separation that prevents overclaiming.

Chapter 8

Conclusions and Future Outlook

This dissertation set out to address a long-standing tension in industrial and collaborative robotics: the need to make robot programming accessible and intuitive to human operators, while preserving physically grounded, explicitly bounded safety guarantees. Rather than proposing a single unifying control framework or a monolithic notion of safety, the thesis deliberately developed and analyzed three complementary programming modalities (physical lead-through programming on industrial robots, physical interaction on collaborative platforms, and language-based programming via Large Language Models) each grounded in a distinct safety paradigm and associated with a clearly delimited set of claims.

The central contribution of the thesis is not a new controller, algorithm, or interface in isolation, but a coherent design doctrine for *safe and easy robot programming*. Across all modalities, ease of programming is achieved by lowering the abstraction barrier at the human–robot interface, while safety is preserved by explicitly bounding a modality-specific physical or structural quantity. Crucially, these bounds are neither implicit nor rhetorical: they are formulated, monitored, and validated within architectures that enforce a strict separation between intent specification, execution authority, and safety enforcement.

In the industrial lead-through case, ease of programming emerges from variable admittance control that allows intuitive physical guidance of a non-collaborative manipulator. Safety, however, is not claimed through force or power limitation, nor through certified Speed and Separation Monitoring. Instead, it is achieved through distance-aware supervision inspired by SSM principles, coupled with passivity-inspired energy reasoning that ensures energetic consistency under time-varying admittance parameters. The identified dynamic model plays a strictly analytical role in this context: it supports interpretation, validation, and the computation of diagnostic quantities.

The second modality extends these insights to collaborative and research robots operating under a Power and Force Limiting paradigm. Here, physical interaction is inherently contact-tolerant, and safety is formulated in energetic terms through

measurable residuals and bounded energy exchange. Variable and adaptive admittance strategies (both algorithmic and optimization-based) are developed within a passivity-consistent framework, where energy tanks serve as the central mechanism for regulating parameter adaptation. The safety claim is thus indexed to a specific, experimentally verifiable quantity: the satisfaction of PFL-related energetic constraints during interaction. As in the industrial case, the thesis avoids conflating modeling accuracy with safety guarantees; the emphasis remains on bounded energy flow rather than on exact dynamic inversion.

The third modality, language-based robot programming using Large Language Models, occupies a deliberately different position in the safety landscape. Here, the contribution lies in demonstrating how natural-language interaction can be integrated into robot programming without granting the language model any authority over execution, planning, or control. The system is explicitly "safe-ish": safety does not derive from compliance with ISO standards, nor from physical metrics such as force, power, or separation distance. Instead, it is enforced structurally, through constrained action spaces, schema-validated task plans, admissible trigger logic, and a deterministic behavior core that retains full execution authority. By making these limitations explicit, the thesis resists the increasingly common but often unsubstantiated tendency to attribute physical safety properties to language models themselves.

Taken together, these three modalities articulate a unified but non-reductive view of safe and easy robot programming. They are not interchangeable, nor do they offer identical guarantees. Rather, each modality earns trust by bounding a quantity that is appropriate to its interaction channel: distance-based energy-modulated control in industrial lead-through programming, energy exchange in collaborative physical interaction, and structural admissibility in language-based task specification. The thesis' primary claim is that such explicit bounding, combined with architectural separation between intent, execution, and safety, is a necessary condition for deploying intuitive programming interfaces without inflating safety claims beyond what the system can substantiate.

8.1 Industrial Implications

From an industrial perspective, the results of this dissertation suggest a pragmatic path toward more human-centered automation without requiring disruptive changes to existing robotic infrastructures. In particular, the industrial lead-through framework demonstrates that intuitive physical programming can be introduced even on non-collaborative manipulators, provided that safety is framed realistically and communicated precisely. Variable admittance control, when coupled with distance-aware supervision and energy-consistent parameter adaptation, enables operators to interact physically with the robot in a controlled and analyzable manner, without implying compliance with safety paradigms that the system does not implement.

More broadly, the work reinforces the idea that human-centered automation is fundamentally an architectural problem, not merely an interface design challenge. Across all modalities, the systems developed in this dissertation emphasize modularity, observability, and explicit responsibility boundaries. Safety logic is not diffused across components, and user-facing interfaces are prevented from bypassing validation or execution constraints. This architectural clarity is essential not only for technical robustness, but also for accountability and certification processes, where the distinction between "inspired by" and "compliant with" specific safety standards must be unambiguous.

Finally, the thesis highlights the importance of transparent communication of safety scope in industrial deployments. By explicitly stating what a system does *not* guarantee (whether in terms of ISO compliance, contact tolerance, or predictive safety) the designer reduces the risk of misuse and misinterpretation. In this sense, restraint becomes an enabling factor: systems that are honest about their limits are more likely to be trusted, integrated responsibly, and extended in the future.

8.2 Future Research Directions

The limitations acknowledged throughout the dissertation naturally delineate several directions for future research, each of which can be pursued without undermining the core separations and non-claims that structure this work.

A first avenue concerns predictive safety reasoning. In both industrial and collaborative contexts, the proposed frameworks rely primarily on instantaneous or one-step evaluations of distance, energy, or residual quantities. Extending these approaches toward short-horizon prediction (while remaining paradigm-specific) could improve responsiveness and robustness in dynamic environments. Importantly, such extensions must respect the distinction between distance-based supervision and PFL-based energetic reasoning, rather than attempting to merge them into a single, ambiguous safety metric.

In the domain of language-based robot programming, future work may focus on enriching the notion of structural safety beyond static validation. Runtime monitoring, failure semantics, and recovery primitives could be integrated into the deterministic behavior core to handle unexpected world-state changes or partial task execution failures. Such extensions would move the system toward being safer by construction, while preserving the fundamental constraint that language models remain interfaces rather than execution authorities.

Finally, digital twins and simulation environments offer promising opportunities as evidence generators rather than safety surrogates. High-fidelity simulations could be used to stress-test interaction policies, validate parameter adaptation strategies, and perform regression testing on language-generated task plans. When framed correctly, such tools can accelerate development and validation cycles without

conflating simulated success with certified safety.

In conclusion, this dissertation argues that safe and easy robot programming is achievable not by collapsing safety, control, and intent into a single layer, but by carefully separating them and bounding their interactions. By embracing this separation across physical and language-based modalities, the work provides both a set of concrete technical contributions and a disciplined methodological stance that can guide future research and industrial practice alike.

Nomenclature

$\boldsymbol{q} \in \mathbb{R}^6$	Joint position vector
$\dot{\boldsymbol{q}} \in \mathbb{R}^6$	Joint velocity vector
$\ddot{\boldsymbol{q}} \in \mathbb{R}^6$	Joint acceleration vector
$\boldsymbol{\tau} \in \mathbb{R}^6$	Joint torque vector
$\mathbf{B}(\boldsymbol{q}) \in \mathbb{R}^{6\times6}$	Joint-space inertia matrix
$\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{6\times6}$	Coriolis and centrifugal matrix
$\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^6$	Gravity torque vector
$\boldsymbol{\tau}_f(\dot{\boldsymbol{q}}) \in \mathbb{R}^6$	Friction torque vector
$\mathbf{F}_v \in \mathbb{R}^{6\times6}$	Viscous friction coefficient matrix
$\mathbf{F}_s \in \mathbb{R}^{6\times6}$	Coulomb friction coefficient matrix
$\mathbf{Y} \in \mathbb{R}^{6\times p}$	Regression matrix
$N \in \mathbb{N}$	Number of identification samples
$p \in \mathbb{N}$	Dimension of full dynamic parameter vector
$p_{\min} \in \mathbb{N}$	Dimension of minimum identifiable parameter vector
$\boldsymbol{\pi} \in \mathbb{R}^p$	Full dynamic parameter vector
$\boldsymbol{\pi}_{\min} \in \mathbb{R}^{p_{\min}}$	Minimum identifiable parameter vector
$\mathbf{Y}_{\text{tot}} \in \mathbb{R}^{6N\times p}$	Global regression matrix
$\mathbf{Y}_{\min} \in \mathbb{R}^{6N\times p_{\min}}$	Reduced regression matrix
$\boldsymbol{\tau}_{\text{tot}} \in \mathbb{R}^{6N}$	Stacked joint torque vector
$t \in \mathbb{R}$	Continuous time variable
$T \in \mathbb{R}^+$	Duration of an excitation trajectory
$\text{RMSE}_i \in \mathbb{R}^+$	Root mean square error of torque reconstruction for joint i
$\boldsymbol{\tau}_{\text{ff}} \in \mathbb{R}^6$	Feedforward joint torque
$d \in \mathbb{R}_{\geq 0}$	Minimum human–robot distance (global diagnostic)
$\mathbf{M}_a \in \mathbb{R}^{6\times6}$	Admittance inertia matrix
$\mathbf{D}_a \in \mathbb{R}^{6\times6}$	Admittance damping matrix
$\boldsymbol{w} \in \mathbb{R}^6$	Interaction wrench at the end-effector
$\boldsymbol{q}_r \in \mathbb{R}^6$	Joint position reference
$\dot{\boldsymbol{q}}_r \in \mathbb{R}^6$	Joint velocity reference
$\ddot{\boldsymbol{q}}_r \in \mathbb{R}^6$	Joint acceleration reference
$\boldsymbol{x} \in \mathbb{R}^6$	End-effector Cartesian pose
$\dot{\boldsymbol{x}} \in \mathbb{R}^6$	End-effector Cartesian velocity
$\ddot{\boldsymbol{x}} \in \mathbb{R}^6$	End-effector Cartesian acceleration
$\dot{\boldsymbol{x}}_r \in \mathbb{R}^6$	Cartesian velocity reference
$\mathcal{H}$	Set of human geometric primitives
$\mathcal{R}$	Set of robot geometric primitives

$\mathcal{D}$	Set of pairwise distances d_{hr}
$N_H \in \mathbb{N}$	Number of human geometric primitives
$N_R \in \mathbb{N}$	Number of robot geometric primitives
$d_{hr} \in \mathbb{R}_{\geq 0}$	Distance between primitives h and r
$\boldsymbol{p}_h \in \mathbb{R}^3$	Closest point on human primitive
$\boldsymbol{p}_r \in \mathbb{R}^3$	Closest point on robot primitive
$\mathcal{SC}_c$	Set of kinetic-energy–based constraints
$SC_{c,hr} \in \mathbb{R}$	Energy-based constraint for pair (h, r)
$SC_c^{\min} \in \mathbb{R}$	Worst-case energy-based constraint
$SC_c^U \in \mathbb{R}$	Upper saturation bound for $SC_c^{\min}$
$SC_c^l \in \mathbb{R}$	Lower saturation bound for $SC_c^{\min}$
$n_{hr} \in \mathbb{R}^3$	Unit vector along closest-point direction
$\mathbf{J}_r^p(\boldsymbol{q}) \in \mathbb{R}^{3\times 6}$	Translational Jacobian associated with robot primitive r
$m_{hr} \in \mathbb{R}_{>0}$	Equivalent mass along n_{hr}
$v_{hr} \in \mathbb{R}$	Relative velocity component along n_{hr}
$C_{hr} \in \mathbb{R}_{\geq 0}$	Relative kinetic energy along n_{hr}
$C_{\max} \in \mathbb{R}_{>0}$	Admissible bound for distance-modulated kinetic energy
$k(d_{hr}) \in \mathbb{R}$	Distance-to-risk modulation function (gain on C_{hr})
$k_1 \in \mathbb{R}$	Modulation gain
$k_2 \in \mathbb{R}$	Modulation offset
$\alpha \in [0, 1]$	Normalized scaling coefficient for admittance adaptation
$\mathbf{M}_a^{\min} \in \mathbb{R}^{6\times 6}$	Minimum admittance inertia matrix
$\mathbf{M}_a^{\max} \in \mathbb{R}^{6\times 6}$	Maximum admittance inertia matrix
$\mathbf{D}_a^{\min} \in \mathbb{R}^{6\times 6}$	Minimum admittance damping matrix
$\mathbf{D}_a^{\max} \in \mathbb{R}^{6\times 6}$	Maximum admittance damping matrix
$f_s \in \mathbb{R}^+$	Perception system sampling frequency
$P_{\text{int}} \in \mathbb{R}$	Interaction power at the human–robot port
$\mathcal{H}_a \in \mathbb{R}_{\geq 0}$	Admittance energy storage function
$\dot{\mathbf{M}}_a \in \mathbb{R}^{6\times 6}$	Time derivative of admittance inertia
$x_t \in \mathbb{R}_{\geq 0}$	Energy tank state
$T(x_t) \in \mathbb{R}_{\geq 0}$	Energy stored in the tank, $T(x_t) = \frac{1}{2}x_t^2$
$T_{\min} \in \mathbb{R}_{>0}$	Minimum tank energy bound
$T_{\max} \in \mathbb{R}_{>0}$	Maximum tank energy bound
$\varphi \in \{0, 1\}$	Tank saturation gate (prevents energy beyond $T_{\max}$)
$\gamma \in \{0, 1\}$	Tank depletion gate (prevents dropping below $T_{\min}$)
$P_D \in \mathbb{R}_{\geq 0}$	Dissipated power, $P_D = \dot{\boldsymbol{x}}^\top \mathbf{D}_d \dot{\boldsymbol{x}}$
$P_M \in \mathbb{R}$	Power due to inertia variation, $P_M = \frac{1}{2}\dot{\boldsymbol{x}}^\top \dot{\mathbf{M}}_d \dot{\boldsymbol{x}}$
$\dot{\mathbf{M}}_a \in \mathbb{R}^{6\times 6}$	Time derivative of desired admittance inertia
$\dot{\boldsymbol{m}}_i^{\text{pass}}(t) \in \mathbb{R}^3$	Diagonal element of the passivity bound on inertia rate
$\dot{\boldsymbol{m}}_{\max}^{\text{pass}} \in \mathbb{R}^3$	Diagonal element of the passivity upper bound on inertia rate
$\dot{x}_{i,max} \in \mathbb{R}^+$	velocity bound of the i-th Cartesian component
$\omega_i \in \mathbb{R}^+$	weights used to distribute the tank energy across the six DOFs
$\dot{\mathbf{M}}_a^{\text{pass}} \in \mathbb{R}^{6\times 6}$	Passivity-aware bound on inertia rate (matrix form)
$\dot{\mathbf{M}}_{a,\max}^{\text{pass}} \in \mathbb{R}^{6\times 6}$	Passivity-aware upper bound on inertia rate (matrix form)
$\dot{\boldsymbol{M}}^U(t) \in \mathbb{R}^{6\times 6}$	Maximum threshold that limit excessive energy injection
$\Delta K_e \in \mathbb{R}_{\geq 0}$	Kinetic energy dissipated during a potential collision

$m_r \in \mathbb{R}_{>0}$	Apparent robot mass along direction $\mathbf{n}$
$m_h \in \mathbb{R}_{>0}$	Equivalent mass of the human body part involved in contact
$\dot{x}_{Rn} \in \mathbb{R}$	Robot velocity component projected along $\mathbf{n}$
$\dot{x}_{Hn} \in \mathbb{R}$	Signed Human velocity component projected along $\mathbf{n}$
$E_{\max}^{\mathrm{PFL}} \in \mathbb{R}_{>0}$	Maximum admissible energy under PFL constraints
$F_{\max} \in \mathbb{R}_{>0}$	Maximum allowable contact force (ISO/TS 15066)
$k \in \mathbb{R}_{>0}$	Effective stiffness of the considered human body part
$A \in \mathbb{R}_{>0}$	Effective contact area
$p_{\max} \in \mathbb{R}_{>0}$	Maximum allowable contact pressure (ISO/TS 15066)
$\mathbf{n} \in \mathbb{R}^6$	Unit vector defining the direction of potential impact
$\mathbf{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times6}$	Geometric Jacobian at configuration $\boldsymbol{q}$
$s \in \{-1, +1\}$	Directional sign encoding approach vs separation motion
$h \in \mathbb{R}$	PFL safety residual (energy feasibility indicator)
$d_r \in \mathbb{R}_{>0}$	Apparent damping along the interaction direction $\mathbf{n}$
$w_n \in \mathbb{R}$	Interaction force projected along $\mathbf{n}$
$\mathbf{M}_a^\star \in \mathbb{R}^{6\times6}$	Nominal (comfortable) admittance inertia matrix
$\mathbf{D}_a^\star \in \mathbb{R}^{6\times6}$	Nominal (comfortable) admittance damping matrix
$\mathbf{M}_{a,\mathrm{tmp}} \in \mathbb{R}^{6\times6}$	Temporary target inertia matrix from energy-loss scaling
$\Delta t \in \mathbb{R}_{>0}$	Controller sampling time
$\dot{\mathbf{M}}_{a,\mathrm{nom}} \in \mathbb{R}^{6\times6}$	Nominal desired inertia rate command
$\dot{\boldsymbol{m}}_{t,\max}^{\mathrm{pass}} \in \mathbb{R}^3$	Translational diagonal of $\dot{\mathbf{M}}_{a,\max}^{\mathrm{pass}}$
$\mathbf{R}_d \in \mathbb{R}^{6\times6}$	Ratio matrix enforcing nominal damping-to-inertia tracking
$\delta \in (0, 1)$	Exponential decay factor for damping recovery after correction
$t_{\max} \in \mathbb{R}_{>0}$	Maximum duration of the damping recovery window
$t_{\mathrm{start}} \in \mathbb{R}$	Timestamp marking the start of the recovery window
$\dot{x}_{\lim} \in \mathbb{R}$	Boundary robot projected velocity enforcing $h = 0$
$\mathbf{D}_a^{new} \in \mathbb{R}^{6\times6}$	Updated damping matrix computed during safe correction
$\boldsymbol{\Phi} \in \mathbb{R}^{6\times6}$	Translational selector matrix for structured damping injection
$\dot{d} \in \mathbb{R}$	Scalar damping-rate used in $\mathbf{D}_a^{new} = \mathbf{D}_a + \dot{d}\boldsymbol{\Phi}\Delta t$
$\mathbf{M}_t \in \mathbb{R}^{3\times3}$	Translational admittance inertia adapted by the optimizer
$\mathbf{D}_t \in \mathbb{R}^{3\times3}$	Translational admittance damping adapted by the optimizer
$\boldsymbol{m} \in \mathbb{R}^3$	Translational inertia diagonal vector, $\mathbf{M}_t = \mathrm{diag}(\boldsymbol{m})$
$\boldsymbol{d} \in \mathbb{R}^3$	Translational damping diagonal vector, $\mathbf{D}_t = \mathrm{diag}(\boldsymbol{d})$
$(\cdot)^+$	Next-step (predicted/applied) under discrete-time update
$\mathbf{u} \in \mathbb{R}^6$	Optimization decision vector $\mathbf{u} = [\dot{\boldsymbol{m}}^\top \ \dot{\boldsymbol{d}}^\top]^\top$
$\dot{\boldsymbol{m}} \in \mathbb{R}^3$	Translational inertia-rate decision variables
$\dot{\boldsymbol{d}} \in \mathbb{R}^3$	Translational damping-rate decision variables
$\boldsymbol{m}_a^\star \in \mathbb{R}^3$	Nominal translational inertia diagonal components
$\boldsymbol{d}_a^\star \in \mathbb{R}^3$	Nominal translational damping diagonal components
$\mathbf{J}_t(\boldsymbol{q}) \in \mathbb{R}^{3\times6}$	Translational Jacobian used in one-step velocity prediction
$\boldsymbol{w}_t \in \mathbb{R}^3$	Translational interaction force (measured) used in prediction
$\boldsymbol{m}_{\min}, \boldsymbol{m}_{\max} \in \mathbb{R}^3$	Lower/upper bounds on translational inertia diagonals
$\boldsymbol{d}_{\min}, \boldsymbol{d}_{\max} \in \mathbb{R}^3$	Lower/upper bounds on translational damping diagonals
$\dot{\boldsymbol{m}}_{\min}, \dot{\boldsymbol{m}}_{\max} \in \mathbb{R}^3$	Lower/upper bounds on translational inertia rates
$\dot{\boldsymbol{d}}_{\min}, \dot{\boldsymbol{d}}_{\max} \in \mathbb{R}^3$	Lower/upper bounds on translational damping rates
$\mathcal{L}(\mathbf{u}) \in \mathbb{R}$	Cost function of the optimization problem

$\lambda_M, \lambda_D, \lambda_u \in \mathbb{R}_{>0}$	Weights in the optimization cost (5.36)
$\sigma \in \mathbb{R}_{\geq 0}$	Slack variable relaxing PFL feasibility in (5.38)
$\rho \in \mathbb{R}_{>0}$	Slack penalty weight in (5.38)

Publications

Journal articles

[66] Annalisa Bertoli et al. “Leveraging Digital Twin Technology with a Human-Centered Approach to Automate a Workstation in the Logistics Sector of Made in Italy: CHIMAR Use Case”. In: *Machines* 13.4 (2025), p. 303.

Conference papers

[21] Matteo Nini et al. “Parameter Identification of a 6-DoF Serial Manipulator with Coupled Joints and Load-Assisting Springs for Industrial Applications”. In: *2024 7th Iberian Robotics Conference (ROBOT)*. IEEE. 2024, pp. 1–8.

[65] Mattia Bertuletti et al. “Towards Safe and Efficient Walk-Through Programming in Actual Industrial Environments”. In: *International Conference on Informatics in Control, Automation and Robotics*. Springer. 2023, pp. 303–341.

[67] Matteo Nini et al. “The Art of Not Getting Smacked: ISO/TS 15066-Compliant Variable Admittance Control for Safe Human-Robot Interaction”. In: *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2025, pp. 6336–6343. DOI: 10.1109/IROS60139.2025.11247254.

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Cesare Fantuzzi, for overseeing this research and for providing the academic context within which this work was developed.

I am grateful to Professor Federica Ferraguti for her support and insightful discussions during the early stages of this research, which contributed to shaping its initial direction.

A special acknowledgment goes to Dr. Andrea Pupa, whose constant support, technical expertise, and patience in navigating my less-than-linear research process have been fundamental throughout this work. His guidance has had a decisive impact on both the development and the experimental realization of the contributions presented in this thesis.

I would also like to thank my colleagues at ARS Control and at IT-I for their collaboration and for the stimulating (and entertaining) working environment in which this research was carried out.

Finally, I would like to thank my family for their continuous support, my friends for their encouragement (and for keeping me grounded outside the lab), and Federica for her presence and support during the final phase of this journey.

Bibliography

[1] Valeria Villani et al. "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications". In: *Mechatronics* 55 (2018), pp. 248–266.

[2] *Robots and robotic devices — Safety requirements for industrial robots — Part 1 and 2*. Geneva, Switzerland: International Organization for Standardization, 2025. URL: `https://www.iso.org/standard/5915511.html`.

[3] *Robots and robotic devices — Collaborative robots*. Geneva, Switzerland: International Organization for Standardization, 2016. URL: `https://www.iso.org/standard/62996.html`.

[4] Bruno Siciliano and Oussama Khatib. "Springer Handbook of Robotics". In: (2016).

[5] Andrea Pupa et al. "A safety-aware kinodynamic architecture for human-robot collaboration". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4465–4471.

[6] Federico Benzi, Federica Ferraguti, and Cristian Secchi. "Energy Tank-based Control Framework for Satisfying the ISO/TS 15066 Constraint". In: *arXiv preprint arXiv:2304.14059* (2023).

[7] Aude Billard and Danica Kragic. "Trends and challenges in robot manipulation". In: *Science* 364.6446 (2019), eaat8414.

[8] R. P. Bonasso et al. "Experiences with an architecture for intelligent, reactive agents". In: *Journal of Experimental & Theoretical Artificial Intelligence* 9.2-3 (1997), pp. 237–256.

[9] Rachid Alami et al. "Toward Human-Aware Robot Task Planning". In: *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*. 2006, pp. 39–46.

[10] Arvid QL Keemink, Herman Van der Kooij, and Arno HA Stienen. "Admittance control for physical human–robot interaction". In: *The International Journal of Robotics Research* 37.11 (2018), pp. 1421–1444.

[11] Roberto Rossi et al. "A pre-collision control strategy for human-robot interaction based on dissipated energy in potential inelastic impacts". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 26–31.

[12] Robin Jeanne Kirschner et al. "Experimental analysis of impact forces in constrained collisions according to iso/ts 15066". In: *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE. 2021, pp. 1–5.

[13] Matteo Ragaglia et al. "Accurate sensorless lead-through programming for lightweight robots in structured environments". In: *Robotics and Computer-Integrated Manufacturing* 39 (2016), pp. 9–21.

[14] Isura Ranatunga et al. "Adaptive admittance control for human–robot interaction using model reference design and adaptive inverse filtering". In: *IEEE transactions on control systems technology* 25.1 (2016), pp. 278–285.

[15] Federica Ferraguti et al. "A variable admittance control strategy for stable physical human–robot interaction". In: *The International Journal of Robotics Research* 38.6 (2019), pp. 747–765.

[16] Alessandro Palleschi et al. "Fast and safe trajectory planning: Solving the cobot performance/safety trade-off in human-robot shared environments". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5445–5452.

[17] Cristian Secchi, Stefano Stramigioli, and Cesare Fantuzzi. "Position drift compensation in port-hamiltonian based telemanipulation". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 4211–4216.

[18] Federica Ferraguti et al. "An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1073–1088.

[19] Sai Vemprala et al. *ChatGPT for Robotics: Design Principles and Model Abilities*. Feb. 2023. DOI: 10.48550/arXiv.2306.17582.

[20] Jesse Atuhurra. *Leveraging Large Language Models in Human-Robot Interaction: A Critical Analysis of Potential and Pitfalls*. 2024. URL: https://api.semanticscholar.org/CorpusID:269502160.

[21] Matteo Nini et al. "Parameter Identification of a 6-DoF Serial Manipulator with Coupled Joints and Load-Assisting Springs for Industrial Applications". In: *2024 7th Iberian Robotics Conference (ROBOT)*. IEEE. 2024, pp. 1–8.

[22] Open Robotics. *ROS 2 Citations (Jazzy)*. 2025. URL: https://docs.ros.org/en/jazzy/Citations.html.

[23] Steven Macenski et al. "Robot Operating System 2: Design, Architecture, and Uses in the Wild". In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074.

[24] J Ernesto Solanes et al. "Human–robot collaboration for safe object transportation using force feedback". In: *Robotics and Autonomous Systems* 107 (2018), pp. 196–208.

[25] Silvia Proia et al. "Control techniques for safe, ergonomic, and efficient human-robot collaboration in the digital industry: A survey". In: *IEEE Transactions on Automation Science and Engineering* 19.3 (2021), pp. 1798–1819.

[26] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*. Vol. 200. Springer, 2008.

[27] Gianvito Azzolini. *Identificazione dei Parametri Dinamici e Controllo Adattativo al Variare del Carico per Manipolatori Robotici*. 2010.

[28] Li Ding et al. "Dynamic Model Identification for 6-DOF Industrial Robots". In: *Journal of Robotics* 2015 (Oct. 2015), e471478. ISSN: 1687-9600. DOI: 10.1155/2015/471478. (Visited on 07/25/2022).

[29] L. Ding et al. "Nonlinear Friction and Dynamical Identification for a Robot Manipulator with Improved Cuckoo Search Algorithm". In: *Journal of Robotics* 2018 (2018). ISSN: 1687-9600. DOI: 10.1155/2018/8219123.

[30] John YS Luh, Michael W Walker, and Richard PC Paul. "On-line computational scheme for mechanical manipulators". In: (1980).

[31] Alexander Dietrich et al. "Practical Consequences of Inertia Shaping for Interaction and Tracking in Robot Control". In: *Control Engineering Practice* 114 (Sept. 2021), p. 104875. ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2021.104875. (Visited on 01/25/2024).

[32] Federica Ferraguti et al. "A Control Barrier Function Approach for Maximizing Performance While Fulfilling to ISO/TS 15066 Regulations". In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020), pp. 5921–5928. ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3010494.

[33] Stavros Grafakos, Fotios Dimeas, and Nikos Aspragathos. "Variable admittance control in pHRI using EMG-based arm muscles co-activation". In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2016, pp. 001900–001905.

[34] Niccolò Lucci et al. "Combining speed and separation monitoring with power and force limiting for safe collaborative robotics applications". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6121–6128.

[35] Federica Ferraguti et al. "Safety Barrier Functions and Multi-Camera Tracking for Human–Robot Shared Environment". In: *Robotics and Autonomous Systems* 124 (Feb. 2020), p. 103388. ISSN: 09218890. DOI: 10.1016/j.robot.2019.103388. (Visited on 09/07/2023).

[36] Andrea Pupa, Marco Minelli, and Cristian Secchi. "A dynamic planner for safe and predictable human-robot collaboration". In: *IEEE Robotics and Automation Letters* 9.1 (2023), pp. 507–514.

[37] Kazuhiro Kawamura et al. "Design philosophy for service robots". In: *Robotics and Autonomous Systems* 14.2 (1995), pp. 187–198.

[38] Tom et al. Brown. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020).

[39] Hongmei Mary He. "Robotgpt: From chatgpt to robot intelligence". In: (2023).

[40] Yixiang Jin et al. "RobotGPT: Robot Manipulation Learning from ChatGPT". In: *IEEE Robotics and Automation Letters* PP (Mar. 2024), pp. 1–8. DOI: 10.1109/LRA.2024.3357432.

[41] Yeseung Kim et al. “A survey on integration of large language models with intelligent robots”. In: *Intelligent Service Robotics* 17 (Aug. 2024), pp. 1091–1107. DOI: 10.1007/s11370-024-00550-5.

[42] Wenxiao Zhang et al. *Enhancing Reliability in LLM-Integrated Robotic Systems: A Unified Approach to Security and Safety*. 2025. arXiv: 2509.02163 [cs.RO]. URL: https://arxiv.org/abs/2509.02163.

[43] Anis Koubaa, Adel Ammar, and Wadii Boulila. “Next-generation human-robot interaction with ChatGPT and robot operating system”. In: *Software: Practice and Experience* 55.2 (2025), pp. 355–382.

[44] Jonas Bode et al. *A Comparison of Prompt Engineering Techniques for Task Planning and Execution in Service Robotics*. 2024. arXiv: 2410.22997 [cs.RO]. URL: https://arxiv.org/abs/2410.22997.

[45] Jacky Liang et al. “Code as Policies: Language Model Programs for Embodied Control”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 9493–9500.

[46] Kento Kawaharazuka et al. “Vision-Language-Action Models for Robotics: A Review Towards Real-World Applications”. In: *IEEE Access* (2025). Received Aug 6, 2025; accepted Sep 10, 2025. DOI: 10.1109/ACCESS.2025.3609980. URL: https://vla-survey.github.io.

[47] Yueen Ma et al. “A Survey on Vision-Language-Action Models for Embodied AI”. In: *arXiv* (2025). eprint: 2405.14093v5. URL: https://arxiv.org/abs/2405.14093v5.

[48] International Standards Organization. *ISO 10218-1 - Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots*. https://store.uni.com/uni-en-iso-10218-1-2012. 2012.

[49] International Standards Organization. *ISO 10218-2 - Robots and robotic devices — Safety requirements for industrial robots — Part 2: Robot systems and integration*. https://store.uni.com/uni-en-iso-10218-2-2011. 2011.

[50] *ISO/TS 15066:2016(E). Robots and robotic devices–Collaborative robots*. Technical Specification. Geneva, CH, Feb. 2016.

[51] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. “A fast procedure for computing the distance between complex objects in three-dimensional space”. In: *IEEE Journal on Robotics and Automation* 4.2 (1988), pp. 193–203. DOI: 10.1109/56.2083.

[52] Alessandro De Luca, Bruno Siciliano, and Loredana Zollo. “PD control with on-line gravity compensation for robots with elastic joints: Theory and experiments”. In: *Automatica* 41.10 (2005), pp. 1809–1819. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2005.05.009. URL: https://www.sciencedirect.com/science/article/pii/S000510980500186X.

[53] *ISO 10218-1:2011(E). Robots and Robotic Devices–Safety Requirements for Industrial Robots–Part 1: Robots*. Geneva, CH, July 2011.

[54] *ISO 10218-2:2011(E). Robots and Robotic Devices–Safety Requirements for Industrial Robots–Part 2: Robot systems and integration*. Geneva, CH, July 2011.

[55] Federica Ferraguti et al. "An Energy Tank-Based Interactive Control Architecture for Autonomous and Teleoperated Robotic Surgery". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1073–1088. DOI: 10.1109/TRO.2015.2455791.

[56] *https://www.augmentus.tech/blog/the-true-cost-of-teach-pendant-programming-in-a-high-mix-manufacturing-landscape/*.

[57] Sherry Ruan et al. *Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices*. Aug. 2016. DOI: 10.48550/arXiv.1608.07323.

[58] OpenAI. "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774* (2023).

[59] Yixiang Jin et al. "Robotgpt: Robot manipulation learning from chatgpt". In: *IEEE Robotics and Automation Letters* 9.3 (2024), pp. 2543–2550.

[60] Sai H Vemprala et al. "Chatgpt for robotics: Design principles and model abilities". In: *Ieee Access* 12 (2024), pp. 55682–55696.

[61] Yifan Zhong et al. "A Survey on Vision-Language-Action Models: An Action Tokenization Perspective". In: *arXiv* (2025). eprint: 2507.01925. URL: https://arxiv.org/abs/2507.01925.

[62] Anis Koubaa. *ROSGPT: Next-Generation Human-Robot Interaction with ChatGPT and ROS*. Apr. 2023. DOI: 10.20944/preprints202304.0827.v1.

[63] Shuyun Cheng, Xian Liu, Kaiyang Jia, et al. "YOLO-World: Real-Time Open-Vocabulary Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024. eprint: 2401.17270.

[64] Yang Ye, Hengxu You, and Jing Du. "Improved Trust in Human-Robot Collaboration with ChatGPT". In: *IEEE Access* PP (Jan. 2023), pp. 1–1. DOI: 10.1109/ACCESS.2023.3282111.

[65] Mattia Bertuletti et al. "Towards Safe and Efficient Walk-Through Programming in Actual Industrial Environments". In: *International Conference on Informatics in Control, Automation and Robotics*. Springer. 2023, pp. 303–341.

[66] Annalisa Bertoli et al. "Leveraging Digital Twin Technology with a Human-Centered Approach to Automate a Workstation in the Logistics Sector of Made in Italy: CHIMAR Use Case". In: *Machines* 13.4 (2025), p. 303.

[67] Matteo Nini et al. "The Art of Not Getting Smacked: ISO/TS 15066-Compliant Variable Admittance Control for Safe Human-Robot Interaction". In: *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2025, pp. 6336–6343. DOI: 10.1109/IROS60139.2025.11247254.