ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 38

Settore Concorsuale: 09/G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF/04 - AUTOMATICA

DESIGN, CONTROL AND COMPUTATIONAL STRATEGIES FOR UNSTRUCTURED ROBOTIC APPLICATIONS

Presentata da: Andrea Govoni

Coordinatore Dottorato

Michele Monaci

Supervisore

Gianluca Palli

Co-supervisore

Claudio Melchiorri

Esame finale anno 2026

"Se funziona al primo colpo, probabilmente non funziona… e neanche ci divertiamo."

Luca Antolini

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Abstract

Faculty of Engineering
DEI – Department of Electrical, Electronic and Information Engineering "G. Marconi"

Doctor of Philosophy in Biomedical, Electrical and System Engineering

Design, Control and Computational Strategies for Unstructured Robotic Applications

by Andrea Govoni

Modern robotics has moved from tightly scripted, structured workcells to unstructured, open-world settings. This shift replaces fixed fixtures and repeatable parts with deformable objects, occlusions, and changing contacts. As a result, manipulation now hinges on perception-driven control, uncertainty handling, and learning from data. We embrace this transition and show how combining model-based priors with adaptive policies enables reliable performance beyond the lab.

The main contribution focuses on deformable linear objects manipulation: industrial applications are developed for wire harness assembly, hose manipulation, and cable storage using a multimodal system that combines vision and tactile sensing. Then, a tactile approach is proposed, where tactile readings are given physical meaning through a Hertzian contact model with Bayesian parameter estimation, enabling reliable force regulation and alignment during grasping.

This work explores 3D-printing applications integrated with Wire Arc Additive Manufacturing . A 14-degree-of-freedom climbing robot is developed: it can anchor to the printed structure with variable geometries to preserve process continuity.

A library for capability maps is developed, encompassing maps of reachability, manipulability, and collision that directly feed motion planning.

In unstructured environments, bridging the Sim2Real gap is essential to transition from laboratory tuning to stable production. We adopt parameter identification, targeted randomization, and progressive validation to make simulation predictive within known limits.

The final work to guide robot in unstructured environments converges into a Task-Priority control framework that actively manages robot limits and constraints while leveraging Robotic Distance Functions, a differentiable distance representations of geometry, to achieve gradient-consistent collision avoidance. In unstructured scenarios, this framework becomes crucial: dynamic task activation and coherent gradient use make planning mistakes less likely and execution more robust.

Contents

List of Figures

List of Tables

List of Abbreviations

DLO	**D**eformable **L**inear **O**bjects
SDF	**S**igned **D**istance **F**unction
RDF	**R**obotic **D**istance **F**unction
C/P	**C**ANDECOMP/**P**ARAFAC tensor decomposition
Go/No-Go	**Go**/**No-Go** test for dimensional inspection
Sim2Real	**Sim**ulation to **Real**ity transfer
Real2Sim2Real	**Real**ity to **Sim**ulation to **Real**ity transfer
F/T	**F**orce/**T**orque
DoF	**D**egrees **O**f **F**reedom
URDF	**U**nified **R**obot **D**escription **F**ormat
CAD	**C**omputer **A**ided **D**esign
PCA	**P**rincipal **C**omponent **A**nalysis
TP	**T**ask **P**riority
IK	**I**nverse **K**inematics
FK	**F**orward **K**inematics
WAAM	**W**ire **A**rc **A**dditive **M**anufacturing
RL	**R**einforcement **L**earning
BO	**B**ayesian **O**ptimization

To my family and my love, who support me every day.
In loving memory of my family members and Ambro.
To everyone who has believed in me.
You're in every step that brought me here.

0.1 Introduction

In the last decades, robotics has undergone a profound transformation, moving from highly structured industrial environments to increasingly unstructured and dynamic scenarios. Modern robotic systems are expected to autonomously operate in everyday life environments characterized by uncertainty in geometry, material properties, and external conditions, where the assumptions of rigid, deterministic, and repetitive processes no longer hold. Applications such as *deformable object manipulation* [31, 35, 32, 62], *multi-robot coordination* [91, 3, 68], and *additive manufacturing on irregular geometries* [79, 77] exemplify this new paradigm, demanding systems that are not only precise but also *adaptive*, *robust*, and computationally efficient in real time.

Traditional industrial manipulators, originally conceived for repetitive and deterministic operations, are ill-suited for these tasks. They are designed to execute predefined trajectories in predictable workspaces, typically with rigid parts and fixed tooling. However, when confronted with contact-rich interactions, deformable materials, or unknown obstacles, these systems lack the necessary feedback and reasoning mechanisms to adapt their behaviour. In particular, the manipulation of Deformable Linear Objects (DLOs), such as cables, hoses, and wiring harnesses, presents unique challenges due to nonlinear elasticity, complex deformation modes, frictional effects, and self-contact[59]. The shape evolution of these objects complicates trajectory generation and make the final configuration intrinsically unpredictable under sparse observations. These characteristics make explicit modelling extremely difficult and introduce uncertainties that propagate into both sensing and control.

Given the unmodeled elasticity, friction, and self-contact, a reliable forward model of cable dynamics is not attainable in practice. We therefore close the loop with multimodal sensing, using vision [10, 9, 27] and interaction sensing (6-axis wrist F/T [12, 17]) and in-hand tactile feedback[84, 13], to estimate state and adapt actions in real time. Vision-based perception, though widely used, struggles in such contexts: occlusions, varying illumination, and large deformations degrade the accuracy of object tracking and state estimation. This limitation has driven increasing attention toward *tactile sensing* as a complementary or alternative modality. Tactile sensors embedded in grippers can directly measure local pressure distributions, shear forces, and slip, providing rich feedback during contact [84, 13, 100, 12], but interpreting these in voltage domains requires meticulous calibration. Among the available physical formulations, Hertzian contact mechanics [57, 89] offers a powerful yet interpretable foundation to relate indentation depth, contact area, and reaction force [12]. Such analytical models enable a more grounded understanding of the contact process, moving beyond black-box data regression. Nevertheless, their application requires calibration of material parameters and sensor nonlinearities, an area in which data driven optimization has recently emerged as a robust approach to identify stiffness coefficients and optimize the tactile model directly from experimental data [52, 70]. Integrating these physics-based models with data-driven calibration closes the loop between sensing and control, allowing grippers to adapt grasping and manipulation strategies to changing contact conditions.

Modeling a DLO from first principles remains challenging; however, simulation and large-scale datasets can approximate the behaviors that matter for control [99]. Data generation and policy testing require simulators that capture DLO physics with sufficient fidelity while remaining tractable. Existing models range from mass–spring–damper systems [59] to Cosserat rods [60] and reduced-order FEM [43], each balancing accuracy, stiffness handling, and numerical stability. Bridging the sim-to-real gap demands calibrated parameters (e.g., Young's modulus, damping, friction) [29, 5], domain randomization [83], realistic sensor/actuator noise [40], and closed-loop controllers that retain performance under model error. Parallel to these developments, the recent explosion of *reinforcement learning (RL)* and physics-based simulation tools has fostered data-driven approaches to robot control[81]. By training policies in simulated environments, robots can learn complex behaviours from interaction data while avoiding the risks

and costs of physical experiments. However, these approaches need to deal with the *sim-to-real gap* [2]: the discrepancy between simulated and real-world dynamics, which remains one of the main obstacles to deployment [41, 105]. This gap arises from imperfect modeling of contact dynamics, friction, or sensor noise, and from differences among simulation engines themselves. To mitigate it, recent works explore techniques such as *domain randomization*, parameter identification, and automatic tuning of dynamics [76, 20, 88].

While Sim2Real remains an open challenge on the *dynamics* side, on the *geometry* side, computer graphics provides accurate discretizations such as watertight meshes. Yet meshes are not directly amenable to fast, differentiable interaction: repeated distance, collision, and gradient queries over high-resolution triangulations are computationally expensive [74] and ill-suited to real-time planning and control. This exposes a complementary open problem: designing compact *functional* representations that approximate mesh-based geometry while supporting cheap evaluations and analytical gradients[36, 74, 7].

Historically, motion planning and collision avoidance have relied on discrete sampling and occupancy grids, which scale poorly and lack differentiability, limiting integration with optimization based control. Sphere based models can approximate robot and obstacle geometries for efficient distance computations [11]. In contrast, *Signed Distance Functions (SDFs)* provide a continuous representation that encodes, at each point, the signed distance to the closest surface, enabling efficient gradient computation for obstacle avoidance and contact reasoning [74, 19].

Building on SDFs, *Robotic Distance Functions (RDF)* adopt a robot-centric formulation [58, 47, 22, 54, 55, 64] in which each link is modeled (e.g. with Bernstein polynomial bases) for analytical distance evaluation and smooth gradients with respect to joint configurations . These properties make RDFs well suited to control architectures that require real-time updates and differentiable reasoning. Moreover, advances in *GPU acceleration* now enable massive parallel evaluation of SDFs and RDFs, supporting millions of distance queries per second . Complementary tensor-decomposition techniques, such as CANDECOMP/PARAFAC [7], compress the high-dimensional data underlying these functions, maintaining high fidelity while reducing computational load.

This convergence of accurate geometric modeling and efficient parallel computation provides a foundation for scalable, real-time robot reasoning in complex environments, where capability maps [102], including reachability and inverse reachability [98], support workspace analysis and base placement [61].

RDFs thus emerge as natural primitives for real-time, differentiable reasoning within optimization and task-priority controllers. To preserve responsiveness in high-dimensional spaces, leveraging GPU parallelism is essential, enabling continuous collision checking and gradient-based optimization without sacrificing control-loop frequency.

From the control perspective, complex and unstructured environments require robots to handle multiple, often conflicting, objectives, maintaining stability, avoiding collisions, tracking targets, and optimizing manipulability, while respecting system constraints. The *Task-Priority (TP)* framework offers a natural formulation, managing concurrent objectives via hierarchical decomposition [11, 8, 86]. In TP control, primary objectives (e.g., contact-force regulation or path tracking) are guaranteed by projecting lower-priority tasks (e.g., joint-limit and collision avoidance) into the null space of higher-priority ones, ensuring task independence, modularity, and real-time feasibility. Extensions for collaborative and mobile robotics introduce activation matrices to modulate priorities dynamically in response to context and sensory feedback [6, 3]. However, most implementations still rely on simplified geometric models and CPU-bound computation, limiting scalability and responsiveness in cluttered or dynamic scenes. Integrating differentiable geometry (via RDF) with GPU-accelerated control provides a direct path to overcome these limitations, enabling continuous collision awareness and reactive adaptation without compromising control stability.

Beyond manipulation, large–scale 3D printing has gained momentum in recent years. Starting from desktop systems for rapid prototyping, the field has moved toward meter–scale printers for structural components in construction and aerospace [79, 77]. These systems introduce complex geometries, overhangs, and supports that strain conventional slicing and path planning. When printing on irregular or evolving surfaces, maintaining torch orientation and stand–off is critical for bead quality and structural integrity. Robotics has therefore converged with Wire Arc Additive Manufacturing (WAAM), in which the part *evolves* during deposition. Yet, open questions remain on how to optimize metal structures under robotic constraints. Designing a self-adapting, mobile system that fabricates objects without fixed workspace constraints, remains a central challenge addressed in this thesis.

The remainder of this thesis takes a pragmatic route through the issues raised above.

Chapter 1 addresses industrial DLO manipulation. Sec. 1.1 reviews tactile sensing and its mathematical aspects; Secs. 1.2–1.3 apply it to wire-harness and medical-hose handling; Sec. 1.4 introduces a Hertzian contact model with data-driven calibration to map voltages to physically interpretable quantities. Finally, Sec. 1.5 presents a *cable warehouse* bridging cable production and cabinet wiring via organized storage and a dedicated gripper for deterministic retrieval.

Chapter 2 addresses robotic 3D printing on irregular, evolving geometries. The goal is to realize a printing system that can *climb* the growing structure while coping with WAAM process constraints and the geometric irregularities induced by layer-by-layer deposition.

To address modeling and transfer (sim-to-real) applications, *Chapter 3* studies cable modeling via a mass–spring–damper approach (Sec. 3.1), together with domain randomization. The objective is to shape model errors so that policies trained with reinforcement learning in simulation behave predictably on hardware, particularly in the presence of uncertainty in stiffness, damping, and friction. This topic is extended in Sec. 3.2, which discusses identification and digital-twin pipelines for torque- and velocity-controlled manipulators.

Object geometry is treated as a computational tool for control. *Sec. 3.3* introduces *Robotic Distance Functions (RDF)* based on Bernstein polynomials, providing analytical distances and gradients with respect to joint states and enabling batched GPU evaluation. To close the gap posed by high tensor costs, we reduce memory and bandwidth via low-rank tensor decompositions (CP), which is essential when many links and obstacles are queried continuously. To support workspace-level decisions, *Sec. 3.4* builds large-scale capability maps (e.g., reachability, manipulability, and collision maps) using batched inverse kinematics and database-backed storage, leveraging parallel GPU computation.

Finally, for robot control, *Chapter 4* presents a general Task-Priority controller for mobile bases and single or dual manipulators. Primary objectives (tracking and contact-force regulation) are preserved while secondary behaviors (joint-limit avoidance, collision avoidance) are projected into their null spaces. RDF are exploited to compute distances and gradients and to maintain continuous collision awareness at high control rates, while activation matrices adjust priorities according to context and measurements.

Chapter 1

Deformable Linear Object Manipulation in Industrial Scenarios

Robotic manipulation of *Deformable Linear Objects* (DLOs), such as cables, medical hoses, and wiring harnesses, remains one of the most challenging topics in modern robotics. Unlike rigid bodies, DLOs exhibit nonlinear dynamics, high flexibility, and low stiffness, making them difficult to model, perceive, and control. Vision-based approaches often fail due to occlusions, deformation, and lighting variability, while conventional force sensors are typically not sensitive enough to capture the subtle contact interactions that characterize their manipulation.

Within this scenario, the industrial handling of cables stands as a paradigmatic example of unstructured robotic applications. In sectors such as switchgear, automotive, and electronics manufacturing, wiring processes still rely heavily on human operators, mainly because cable management and routing require complex dexterity and high-level decision making. Even though modern cable production systems, such as the Komax ZETA 640, achieve remarkable automation in cutting, crimping, and labeling, the subsequent phases, cable collection, storage, and retrieval, remain largely manual. This discontinuity breaks the automation chain and represents a critical bottleneck for achieving fully autonomous robotic wiring.

Bridging this technological gap demands not only advanced perception and control algorithms but also the design of intelligent and adaptive hardware capable of creating structured environments out of inherently unstructured tasks. In this context, tactile sensing plays a crucial role, as it enables the estimation of local contact forces and pressure distributions, supporting feedback-based strategies for robust and adaptive manipulation of flexible materials.

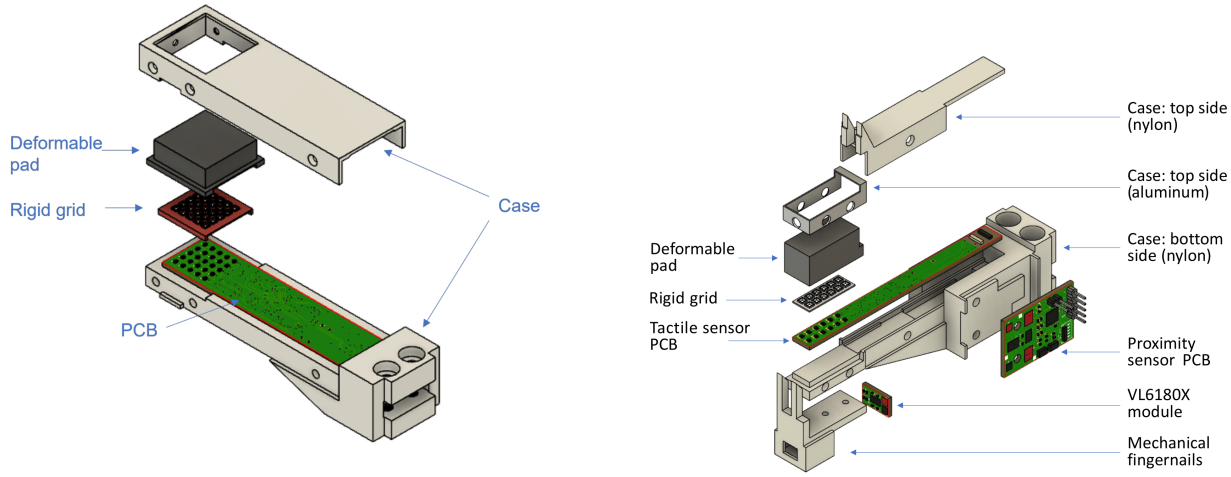
Building upon these concepts, this chapter presents a robotic warehouse system for automated cable management, designed to integrate directly with production machines and operate as an intermediary stage between cable fabrication and robotic wiring. The system combines a customized gripper, equipped to handle cables of varying diameters, with a modular warehouse composed of adaptive clips that allow deterministic storage and retrieval. By transforming chaotic post-production layouts into organized, robot-readable structures, the proposed framework exemplifies how mechanical design, control strategies, and computational reasoning can converge to extend robotic autonomy in unstructured industrial environments.

The chapter is organized as follows:

- Section 1.1 provides a theoretical overview of tactile sensing and its role in DLO manipulation;
- Sections 1.2 and 1.3 review previous applications of tactile sensing in the manipulation of wiring harnesses and medical hoses; these works are published in [51, 35] and [32], respectively;
- Section 1.4 addresses the literature gap and, building on Section 1.1, introduces the development of a physics-based contact model based on plane–cylinder Hertzian contact mechanics. This work is under submission at RAL [31];

- Section 1.5 introduces the proposed robotic warehouse system, detailing its mechanical design, integration, and control framework; this work is published in [33];

1.1 Tactile Sensing Overview for Parallel Grippers



(a) Tactile sensor 5x5 integrated in the Robotiq Hand-E parallel gripper [13].

(b) Tactile sensor 2x6 taxels [51].

Figure 1.1. Tactile sensors used in this work.

Figure 1.1 illustrates the tactile sensors adopted in this work. The first device (Fig. 1.1a) consists of a 5×5 optoelectronic array integrated into the active finger of the Robotiq Hand-E parallel gripper.
The second device (Fig. 1.1b), instead, is a 2×6 tactile array specifically designed for robotic manipulation of deformable linear objects in wiring harness assembly tasks.

The two devices share a similar layered structure: a deformable silicone pad with a reflective bottom surface ensures reliable infrared reflection, while black optical barriers between adjacent taxels minimize crosstalk. Beneath the pad, each taxel combines an infrared LED and a co-packaged phototransistor operating in reflection mode. The phototransistor outputs are sampled by dedicated microcontrollers equipped with multi-channel ADCs, enabling real-time acquisition of tactile maps for integration into robotic control loops. These voltage signals, denoted as v_i for each taxel i, form a tactile map that represents the spatial distribution of contact-induced deformation.

Despite these common design principles, the two sensors differ significantly in their geometry and functional integration. The 5×5 array provides a nearly square sensing area of approximately 21×21 mm^2, with a spatial resolution of 3.55 mm.
This configuration favors compactness and uniform resolution, making it particularly effective for reconstructing detailed local pressure distributions and estimating contact forces in general manipulation tasks. In contrast, the 2×6 array extends the sensitive surface along one axis, covering an area of about 22.6×8.4 mm^2 with elongated geometry. Furthermore, this sensor integrates additional elements to support industrial scenarios: a proximity module based on a time-of-flight (ToF) infrared sensor allows non-contact distance measurement, while mechanical "fingernails" assist in separating, guiding, and stabilizing wires during routing operations.

In summary, both tactile devices are based on the same optoelectronic principle and share the fundamental architecture of silicone pad, reflective layer, and LED–phototransistor taxels. However, they embody different design trade-offs: the 5×5 array emphasizes high-resolution and compact force estimation, whereas the 2×6 array integrates multimodal sensing and structural features specifically optimized for the manipulation of deformable linear objects.

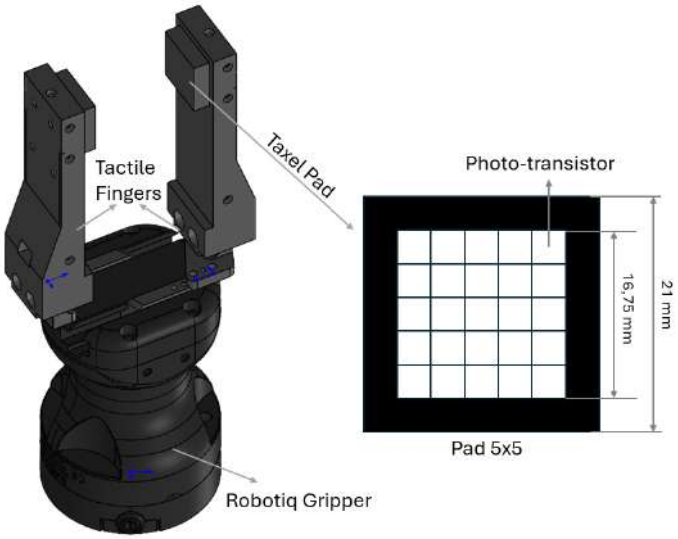


Figure 1.2. Tactile sensor integrated in the Robotiq Hand-E parallel gripper [13].

1.1.1 Principle of Operation

The tactile sensors employed in this thesis are based on **optoelectronic technology**. They consist of:

- a deformable silicone pad with reflective bottom surface,
- a matrix of photo-reflectors (LED + phototransistor pairs), typically arranged in 5×5 or 2×6 configurations,
- an electronic board with 12-bit ADC channels sampling up to 500 Hz.

When an object deforms the pad, the reflected IR light intensity changes, producing voltage signals v_i that are proportional to local pressure. These sensors are embedded in the active fingers of a parallel gripper (Robotiq Hand-E) or in custom-designed robotic fingers, enabling real-time interaction with DLOs.

1.1.2 Mathematical Representation of Tactile Maps

Let N be the number of taxels. Each taxel has coordinates (x_i, y_i) and an associated voltage v_i. The tactile map is:

$$\mathcal{T} = \{(x_i, y_i, v_i) \mid i = 1, \ldots, N\}.$$

When the taxels are arranged on a 2D grid, we will also refer to them through a pair of indices (i, j) with coordinates (x_i, y_j), which is equivalent to using a single linear index.

Normal Force Estimation. In the continuous case, the total normal force can be described by extending the classical 1D formulation of contact pressure (e.g., [89], Eq. (4)) to a 2D configuration:

$$F_N = \int_{-\frac{L_y}{2}}^{\frac{L_y}{2}} \int_{-\frac{L_x}{2}}^{\frac{L_x}{2}} p(x) dx dy \tag{1.1}$$

The total normal force is estimated as:

$$F_N \approx \sum_{i=1}^{N} p(x_i, y_i)\, \Delta x\, \Delta y \tag{1.2}$$

where $p(x_i, y_i)$ is the pressure mapped from v_i, and $(\Delta x, \Delta y)$ are the taxel coordinates.

Tactile Centroid. The centroid of the tactile activation is:

$$x_c = \frac{\sum_{i=1}^{N} v_i x_i}{\sum_{i=1}^{N} v_i}, \qquad y_c = \frac{\sum_{i=1}^{N} v_i y_i}{\sum_{i=1}^{N} v_i} \tag{1.3}$$

It should be noted that the tactile centroid does not correspond to a physical point within the sensor. Rather, it is a mathematical indicator that summarizes the distribution of contact-induced deformations, providing a compact feature for estimating the location and movement of the applied load.

Centroid Displacement. Its displacement with respect to the unloaded centroid (x_{c0}, y_{c0}) is:

$$\Delta x_c = x_c - x_{c0}, \qquad \Delta y_c = y_c - y_{c0} \tag{1.4}$$

The centroid displacement does not measure tangential forces directly, but it provides a reliable indicator of their presence, since lateral loading skews the tactile activation pattern and shifts its centroid relative to the unloaded state.

Mean Pressure Indicator. To compute the tactile indicator of the mean pressure p_{mean} acting over the tactile array, we define:

$$p_{\text{mean}} = \frac{1}{N_{\text{active}}} \sum_{(i,j)\in\mathcal{A}} p(x_i, y_j) \tag{1.5}$$

where $\mathcal{A}$ is the set of indices (i, j) corresponding to active taxels (i.e., those registering non-zero or valid pressure values), while N_{active} is the cardinality of $\mathcal{A}$, i.e., the number of active taxels.

1.2 Dual Arm for Wire Harness Manufacturing

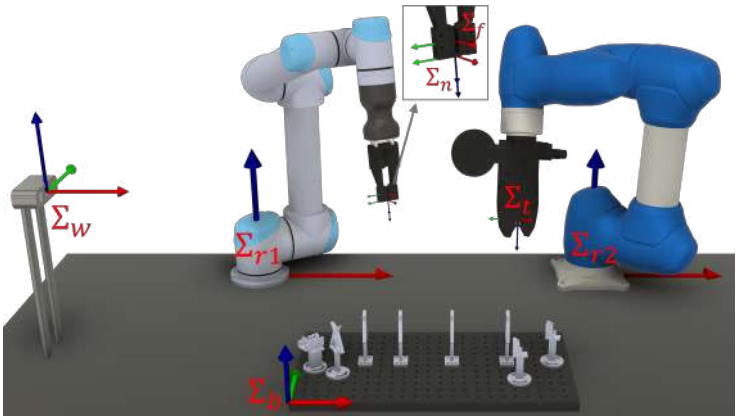


Figure 1.3. Dual-arm robotic cell for wiring harness assembly. The manipulation arm (left) is equipped with tactile–proximity fingers, while the auxiliary arm (right) holds the harness and applies tape.

Wire harness assembly is a key manufacturing step in several industrial domains, where connectors must be inserted into housings and wires must be routed and secured through clips according to a prescribed layout. Despite being repetitive, the task remains largely manual

(a) External view **(b)** Internal view

Figure 1.4. CAD model of the taping gun: (a) external and (b) internal views.

in practice [90, 69, 71], mainly because it involves deformable, lightweight components, tight clearances, and frequent occlusions in cluttered workspaces.

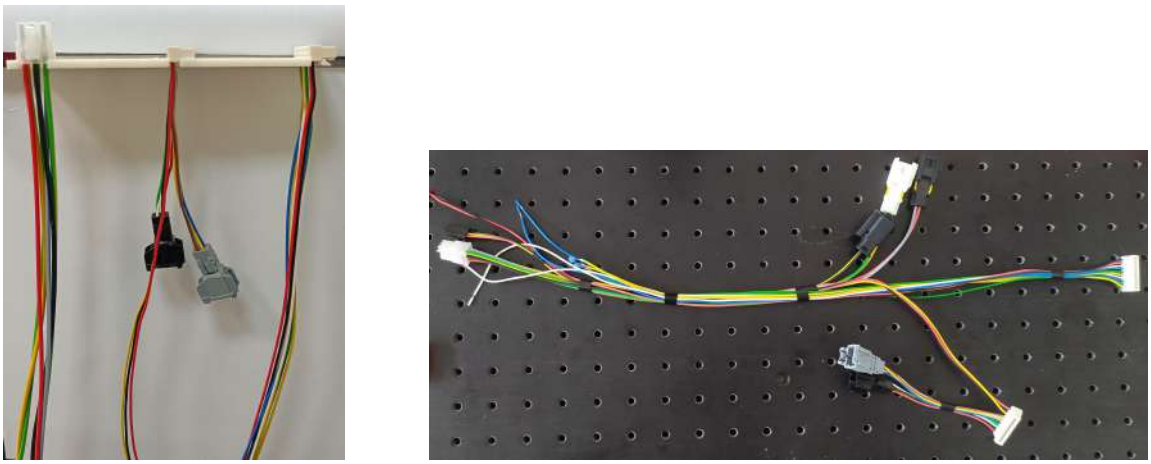
From an automation perspective, wiring harnesses are a representative example of deformable linear objects (DLOs): small geometric variations and elastic effects can lead to large differences in behavior during routing and insertion. These characteristics challenge conventional sensing and control. Vision-based pipelines, while informative at a global scale, degrade when the interaction happens close to the end-effector, where occlusions, reflections, and variable lighting are common. Conversely, standard wrist-mounted force/torque sensing is often insufficient with lightweight wires and snap-fit features, because the relevant interaction forces may be comparable to the sensor noise floor and because contact is highly localized.

For these reasons, recent research has increasingly explored richer sensing and task-specific strategies. Tactile sensing, in particular, is well suited for cable manipulation: tactile fingers can support contour following and routing [66], and tactile-equipped grippers can maintain reliable contact while sliding along cables of different materials and diameters [84]. At the cell level, additional operations are typically required to obtain a stable, production-ready harness, such as taping to stabilize branches [80] or CAD-driven approaches that align routing with design intent, especially in aerospace contexts [30]. However, existing solutions often address individual sub-tasks (e.g., sensing, routing, insertion, or taping) in isolation, and do not provide an integrated and reconfigurable robotic cell that can autonomously execute the full assembly sequence with online verification and recovery across variant-rich production.

This section presents a unified approach based on a dual-arm robotic workcell equipped with multi-modal end-effector sensing. The primary arm performs the fine manipulation steps using gripper fingers embedding tactile and proximity sensing (Fig. 1.1b), enabling robust perception of local geometry and contact conditions during connector insertion and wire routing. The second arm cooperates by holding the harness to improve stability and by executing spot taping operations at selected locations. Finally, a vision system inspects the completed assembly (Fig. 1.5b), enabling error detection and recovery at the end of the process.

We focus on a representative assembly workflow that includes: (i) grasping harness branches from a warehouse, (ii) inserting connectors into snap-fit holders on the workbench, (iii) guiding wires along predefined paths and securing them into custom routing clips, and (iv) applying spot taping to bundle and stabilize the harness. The rest of the section details the system setup and calibration, the design of dedicated fixtures (holders and routing clips), and an analysis of the forces involved in connector insertion. Overall, our goal is to provide a robust and adaptable alternative to prior task-specific solutions by combining dual-arm cooperation, tactile–proximity sensing at the end-effector, and vision-based final verification.

This work builds upon and extends our previous publications [51], which describe the design of the dual-arm cell and the experimental validation of the wiring harness assembly process.



(a) Cable stored in the warehouse.

(b) Final wiring harness.

Figure 1.5. (a) A wiring harness branch stored in the warehouse; (b) the final wiring harness after assembly.

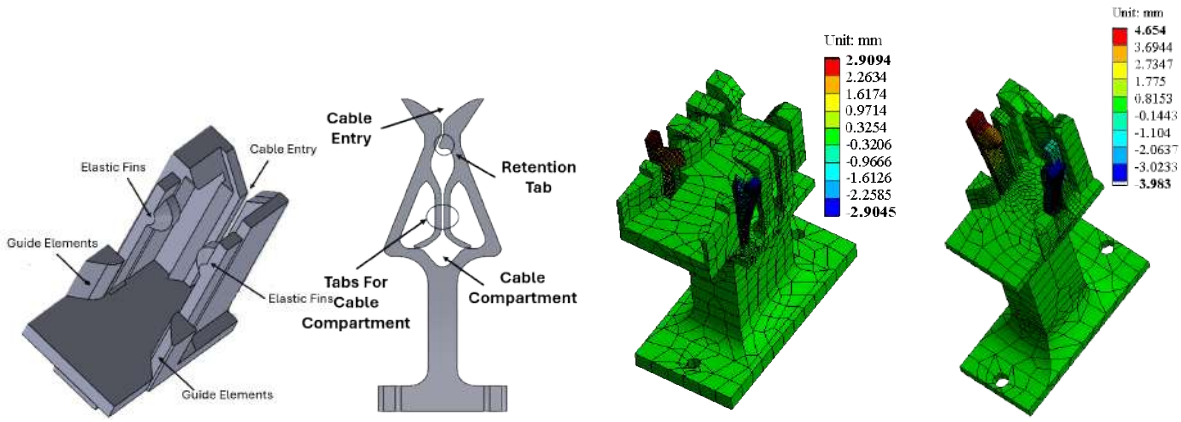
1.2.1 System Overview

The robotic cell, shown in Fig. 1.3, consists of two 6-DOF robotic arms (a UR5 and a Doosan M0609), with complementary roles. A manipulation arm (left-Σ_{r1}), equipped with tactile 2x6–proximity fingers shown in Fig.1.1b, used for grasping, routing, and positioning the cables;
An auxiliary arm (right-Σ_{r2}), equipped with a lightweight taping tool shown in Fig. 1.4, which holds the harness steady and applies adhesive tape at designated points.
In front of the robots, a workbench hosts the assembly area, with snap-fit holders and routing clips fixed to its surface. While on the left side a vision system inspects the final assembly for quality control.

The overall assembly process is organized as the repetition of three main subtasks: pick and place of the branches, routing of the cables, and spot taping. The two arms must work in a coordinated manner, starting from disassembled components stored in the warehouse (Fig. 1.5a) and culminating in the fully assembled wiring harness (Fig. 1.5b).

1.2.2 Assembly and Rooting Setup Design

The targeted assembly setup, indicated by Σ_b in Fig. 1.3, is designed to hold three different wire cables, each equipped with a 6, 10, or 11 pins connector (Fig. 1.5a). To guarantee proper positioning and stability of these connectors during cable routing, dedicated *connector holders* and *routing clips* were designed and manufactured using 3D printing. The holders (Fig. 1.6) are tailored to the geometry of the 6-pin (Fig. 1.6c) and 10/11-pin (Fig. 1.6d) connectors, providing a reliable snap-fit mechanism that locks them onto the workbench, while the routing clips (Fig. 1.6b) guide the wires along predefined paths and keep them bundled throughout the assembly process.



(a) 11/10-pin holder (CAD). **(b)** Routing clips (CAD). **(c)** 6-pin holder (CAD/FEA). **(d)** 11-pin holder (CAD/FEA).

Figure 1.6. Custom snap-fit holders and routing clips designed for the wiring harness assembly.

Connector holder functionality.

The design of the connector holder (see Fig. 1.6a) was carefully engineered to simplify insertion while ensuring secure locking. Referring to Fig. 1.7 its working procedure requires the following steps; once the cable connector is properly inserted on its track, the robot pulls it into the cage, where dedicated mechanical features guide and constrain its motion. Two *elastic fins* with small teeth elastically deform during insertion and subsequently resist motion in the opposite direction, preventing accidental release from the cage. During the insertion process two *guiding elements* compensate for small positional errors due to cable deformation, ensuring proper guidance inside the holder. During the task the robot monitors the tangential force indicator coming from tactile, expressed by Eq.(1.4), that estimates the force of the cable on the fins, to detect the moment when the connector is fully inserted and locked in place.

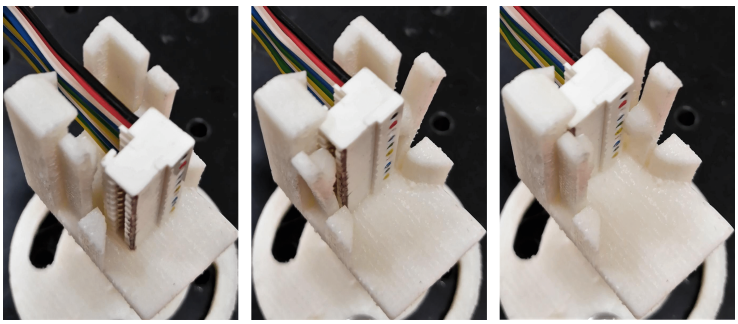


Figure 1.7. Insertion and locking of the 11-pin connector into its holder

Connector Holder Design and FEA Analysis Recall Fig.1.6a, to be sure that the connector is securely locked in place after insertion, the elastic fins must keep connectors firmly during robot routing operations. At the same time, excessive stiffness could lead to difficulties during insertion or even damage to the connector. For this reason, their geometry must be optimized to allow easy insertion by the robot, ensuring that the required force remains within acceptable limits in order to avoid slipping during the pulling phase.

To validate the design, Finite Element Analysis (FEA) was performed considering ABS as printing material, with mechanical parameters depending on the extrusion direction [63]. The simulations quantified the force required to deform the elastic tabs and allow insertion:

- $\approx 15\,\mathrm{N}$ for the 6-pin connector holder;

- $\approx 11\,\text{N}$ for the 10/11-pin connector holders.

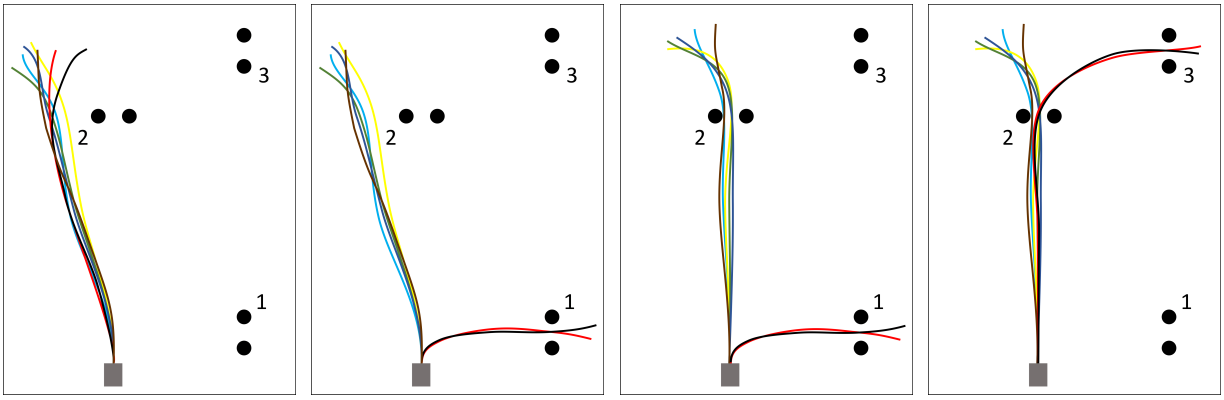
These values provide reference for programming the robot's insertion strategy.

Routing Clips Design

The routing clips (Fig. 1.6b) have a curved geometry that ensures cable retention while allowing easy insertion and removal. Key features include:

- a V-shaped cable entry guiding the wires into the compartment (**Cable Entry**);
- a flexible retention tab preventing accidental release (**Retention Tab**);
- deformable tabs that keep the cables bundled once inserted (**Cable compartment**).

A brief functionality description of the routing clips is shown in Fig. 1.8: the robot positions the connector in the cage (Fig. 1.8a); once the cable is constrained, it separates the wires (Fig. 1.8b); finally it routes and secures them through the clips (Fig. 1.8c and Fig. 1.8d).



(a) Connector insertion. **(b)** Validation and wire separation. **(c)** Routing through clips. **(d)** Final branch securing.

Figure 1.8. Schematic steps of the wiring harness assembly process.

1.2.3 CAD information

To make the system general and not linked to a single wiring harness design, all the trajectories executed by the robots during the task are derived from configuration files containing the routing information. With reference to Fig. 1.8 that shows the main steps of the wiring harness assembly process, the robot uses the CAD information to:

- **Step 1 (Fig. 1.8a)**: extract the position and orientation of the clips and holders, to plan the approach and insertion trajectories;
- **Step 2 (Fig. 1.8b)**: retrieve the geometric parameters of the wires (number of pins, pitch, length, etc.) to compute the grasping height and separate the wires without damaging them;
- **Step 3 (Fig. 1.8c)**: build the routing paths from linear and circular segments, with optional offsets for parallel insertion;
- **Step 4 (Fig. 1.8d)**: route the second branch in its specific clips and secure the harness with spot taping.

Figure 1.9 shows the position of the clips and holders in the robot workspace, as extracted from the CAD model of the wiring harness.

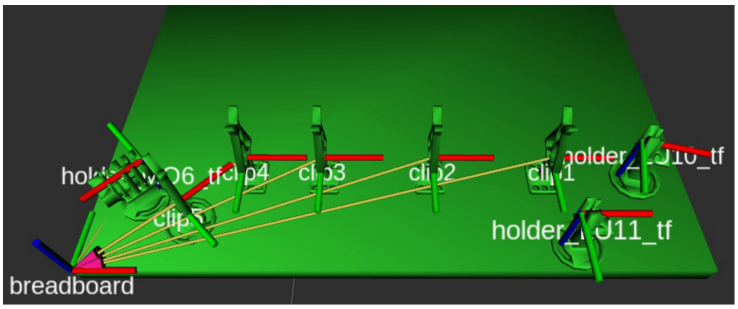


Figure 1.9. Position of the clips and holders in the robot workspace. The frames are extracted from the CAD model of the wiring harness.

1.2.4 Visual inspection

During the execution of the operations described in Sec. 1.2.3, the system acquires a 3D point cloud of the pinboard with a Zivid One depth camera to verify that each routing operation has been correctly executed. Only depth data are used; RGB information is not exploited in this work and is left for future extensions. Our approach follows the scalable strategy of [27].

Network. Adopting a *shared-encoder, multi-head* architecture to balance precision and scalability in wiring-harness assembly, where multiple clips share geometry but differ in visibility and required wire count. The encoder is based on PointNet++ [78] and maps the cropped point cloud to a latent feature vector. For each clip, a lightweight decoder—three linear MLP layers—outputs the probability that the cables are correctly inserted. Adding a new clip only requires training its decoder while keeping the shared encoder frozen, which keeps fine-tuning fast and parameter-efficient.

For each clip we collected 90 labeled samples (45 *Positive*, 45 *Negative*, with negatives stratified by increasing difficulty). We further augmented the dataset with 500 labeled samples acquired during real executions on the assembly board (both *Positive* and *Negative*).

Given the known clip poses, we crop a $10 \times 10\,\text{cm}$ region around the target clip to reduce irrelevant points and speed up inference, align the cloud to a canonical orientation, and normalize coordinates to $[0, 1]$ before feeding them to the network.

1.2.5 Results

Task-Level Pipeline.

The complete dual-arm pipeline partially shown in Fig1.10, is modular and draws task parameters from configuration files (CAD of clips/holders, component data, operation sequence). A typical cycle comprises:

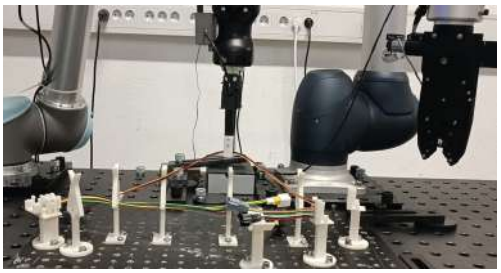
1. **Connector insertion.** The connector is retrieved and inserted into a snap-fit holder, then pulled until a calibrated current threshold confirms correct activation.

2. **Wire grasp & separation.** Geometric parameters define the grasp height, enabling selective splitting of branches without affecting neighbors.

3. **Routing.** Wire paths are built from linear/circular segments, with optional offsets for parallel insertion; tensioning is achieved via controlled finger motion.

4. **Pushing.** Wires are vertically pressed into the clips beyond the retention tab. No further analysis is provided, as the robot points for this operation are strategically defined through experimental tuning.

(a) Pick and rout the connector 6 poles.



(b) Pick and rout the connector 11 poles, with wire separation.

(c) Pick and rout the connector 10 poles.

(d) Taping the harness.

Figure 1.10. Main schematic steps of the wiring harness assembly process, routing and securing two branches.

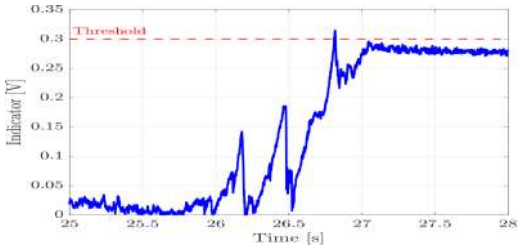
5. **Spot taping.** An auxiliary robot arm uses a taping gun to wrap wires with alignment defined relative to the clip frame.

6. **Inspection.** A depth camera and PointNet++ classify insertions; errors trigger local corrective actions.

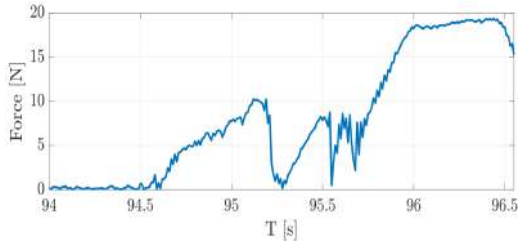
Connector Insertion

Using the CAD information (Sec. 1.2.3), the robot picks each branch from the warehouse and inserts the corresponding connector into a snap-fit holder fixed on the workbench (see Fig. 1.7, Fig. 1.10a, and Fig. 1.10b). The first step toward enabling reliable harness manipulation was presented in [35], where tactile sensing was exploited to estimate tangential forces during grasping. In this operation the robot executes a controlled pulling motion: it moves in the insertion direction with constant speed, while the tactile sensor monitors the tangential forces exerted on the grasped wires. From the displacement of the tactile centroid (Eq. 1.4) an indicator proportional to the shear force is computed, and the robot motion is stopped once this indicator exceeds a predefined threshold, as illustrated in Fig. 1.11b, confirming that the connector is securely locked in place.

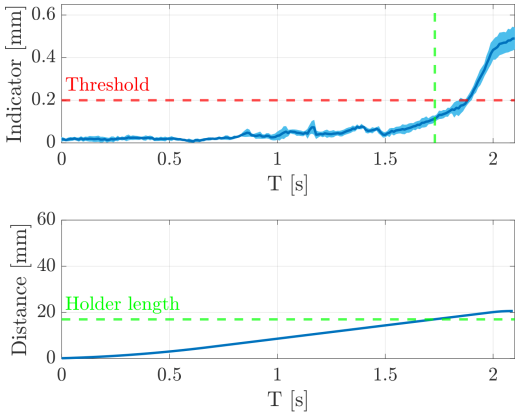
This method, based on the centroid displacement of the tactile pad, enables the detection of incipient slip during cable grasping, the estimation of shear forces without the need for additional force/torque sensors, and the closed-loop control of grasp forces in exploratory and alignment tasks. The connector insertion and locking procedure has been validated by comparing tactile indicator data with FEM predictions (Fig. 1.6) and with measurements from an external F/T sensor. As reported in Fig. 1.11a, the chosen threshold on the centroid-based indicator (0.2 mm, corresponding to $15 \sim 20$ N shown in Fig. 1.11c) ensures complete insertion, guarantees robustness against uncertainties, and remains well below the slipping limit of 30 N. The experimental results confirm consistency with both FEM simulations and the reference sensor readings, validating the effectiveness of the proposed approach.



(a) Centroid displacement during LU10 connector insertion. The red dashed line indicates the threshold for stopping the robot motion.

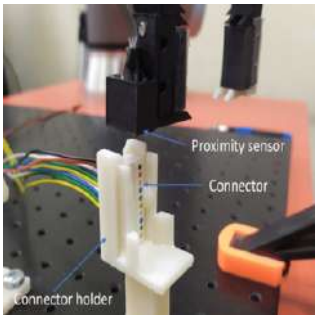


(c) Force measured during LU11 connector insertion, showing a maximum force of about 10 N before complete locking.

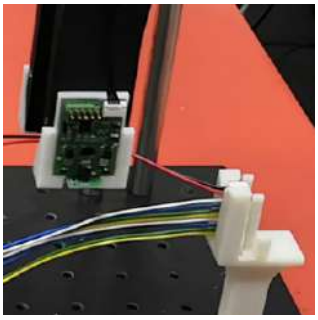


(b) Tactile indicator trend (mean and variance) during five LU11 insertions. The threshold of 0.2 mm is activated only after the holder length (dashed green line) is traveled.

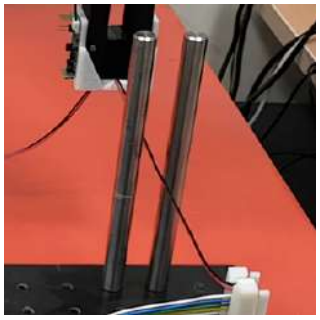
Figure 1.11. Experimental validation of the insertion task: (a) centroid displacement, (b) force profile, (c) tactile indicator response.



(a) Proximity measurement during wire approach.



(b) Wire separation using tactile fingers.



(c) Move the wire branch separated and start routing the other one.

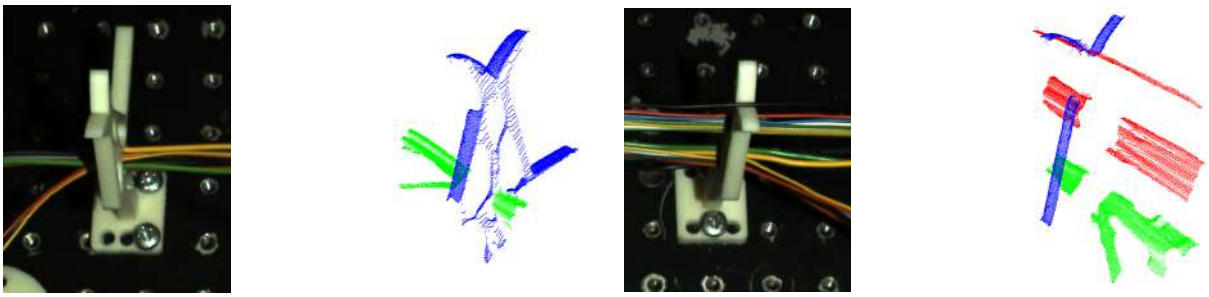
Figure 1.12. (a) Proximity measurement during wire approach; (b) wire separation using tactile–proximity fingers; (c) tensioned wire branch before routing completion.

Cable Separation and routing

Once the connector is securely locked in its holder, the robot separates the wires and routes them through the guiding clips, again exploiting CAD information (Sec. 1.2.3). In this phase, the tactile–proximity fingers are crucial:

- the **proximity sensor** measures the distance to the wires during approach, ensuring accurate finger positioning before grasping;
- the **tactile sensor** confirms secure contact, allowing the robot to grasp the wires firmly without applying excessive force that could damage them.

By combining these two sensing modalities, the robot can reliably isolate individual wires and guide them along the predefined routing paths. Figure 1.12 illustrates key moments of this



(a) Correct Insertion **(b)** Incorrect Insertion

Figure 1.13. Qualitative examples after clip cropping: (a) two views with correct insertion (high *OK* probability); (b) two views with incorrect insertion (cables out of seat → low *OK* probability). Colors: blue = clip, green = correct cables, red = misplaced cables.

process: distance measurement during approach (Fig. 1.12a), wire separation after grasping (Fig. 1.12b), and insertion into the clips (Fig. 1.12c).

Pushing and Spot Taping

This operation of **pushing** is carried out to force the wires in the cable compartment of the clip by pushing them down. The only required information is the clip where the pushing operation is required. The executed trajectory is a vertical linear movement, from top to bottom, at the specified side of the selected clip.

The last operation reported is **taping**, that is the application of tape at predefined points usually located in the proximity of at least one clip to ensure that the wires are locked in position. The trajectory is generated starting from the following parameters: clip frame, Δx, Δz, and x_{rot}. The taping gun frame, Σ_t, is aligned with the clip frame Σ_{curr}, but with an offset Δx in the x-direction, which sets the desired distance from the clip, and an offset Δz in the z-direction. Furthermore, a rotation equal to x_{rot} about the x-axis of Σ_{curr} is applied.

Visual Inspection

Figure 1.13 shows two pairs of cropped views per clip. On the left (a) are two examples of *correct insertion*: cables follow the intended path inside the clip, and the model assigns a high *OK* probability. On the right (b) are two *incorrect* cases: cables are misplaced or out of the seat, yielding a low *OK* probability. Colors (when present): blue = clip, green = correct cables, red = misplaced cables.

1.3 Medical Hose Quality Inspection

Automating the manipulation and inspection of Deformable Linear Objects (DLOs) such as medical hoses is difficult due to millimetric diameters, high deformability, low mass, and variable contact dynamics. Under these conditions, vision-only or purely model-based strategies become brittle, and force/torque cues approach sensor noise. Clean-room constraints further limit fixturing and throughput-friendly human supervision.

Prior work in quality control (QC) for medical devices has largely relied on calibrated vision pipelines and rigid workholding, e.g., stent inspection and extrusion monitoring with line-scan or multi-axis imaging [39, 28], with only a few reports of robotic pilots on production lines. Tactile sensing has enabled precise contact reasoning for rigid or semi-constrained tasks using both commercial and research devices (BioTac [24], GelSight [101], [84]), including peg-in-hole and wire handling [15], and there is growing interest in alternative transduction and low-pressure

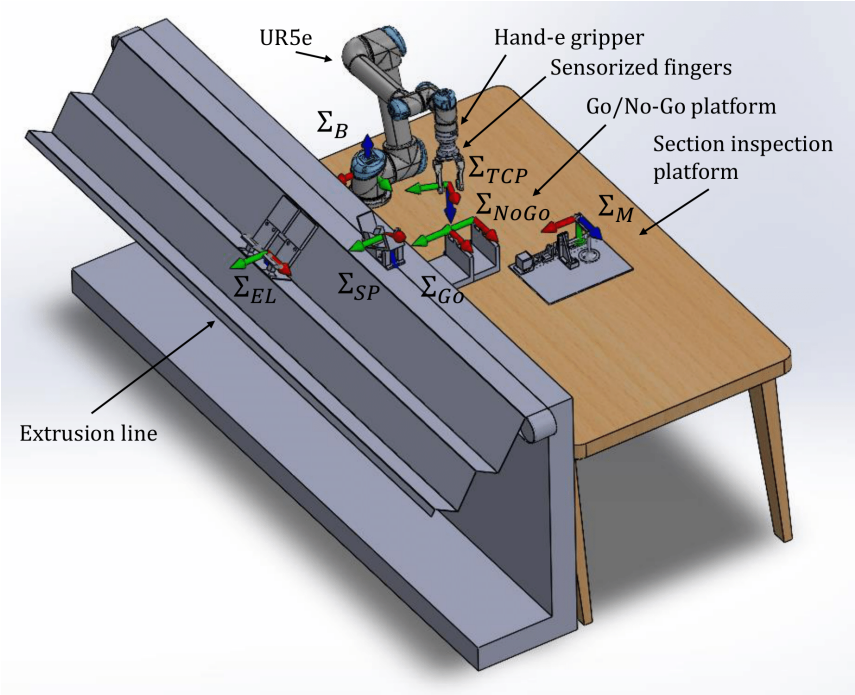


Figure 1.14. Medical hose quality inspection setup.

characterization. Search strategies under uncertainty (e.g., spiral or hybrid policies) have been proposed to localize features without precise geometry [65]. However, these strands rarely converge on the specific setting of soft, lightweight hoses in clean-room QC, where contact signatures are weak, geometry is poorly known, and safety margins are tight.

This work closes that gap with a robotic platform that integrates optoelectronic tactile fingertips and a dedicated cutting unit for sample preparation, coupled with tactile indicators and blind-search policies that detect, align, and evaluate contact without precise geometric priors. For section inspection under microscopy, we exploit RGB imagery to improve generalization and apply color-based binarization and no-reference sharpness metrics when a pristine reference is unavailable [104, 72, 23, 53, 1]. We demonstrate the approach on three QC tasks, sample preparation, Go/No-Go inner-diameter verification, and micro-section inspection, showing that the synergy of mechatronic design, tactile sensing, and adaptive control yields safe, repeatable manipulation in unstructured, variable conditions.

1.3.1 Setup Description

The developed robotic setup is composed of a UR5e collaborative robotic arm by Universal Robots, equipped with a Robotiq Hand-E parallel gripper featuring custom sensorized fingers, as described and shown in Fig. 1.1a. Referring to Fig. 1.14 and 1.15, the robotic cell is composed of:

- an **extrusion line** where the robot picks medical hose samples where the hoses are produced and collected for quality inspection (reference frame Σ_{EL}),
- a **quality inspection station** (named Go/No-Go Σ_{NoGo} testing platform) where the robot performs dimensional tests on medical hoses using tactile feedback (reference frame Σ_{NoGo}),
- a **cutting and microscope station** (named) where the robot cuts the hose samples and prepares them for visual inspection under a digital microscope (reference frame Σ_M).

(a) Pick from extrusion line.

(b) Move to alignment support.

(c) Place on alignment support.

Figure 1.15. Robotic pick-and-place sequence from extrusion line to alignment support.

1.3.2 Custom collector

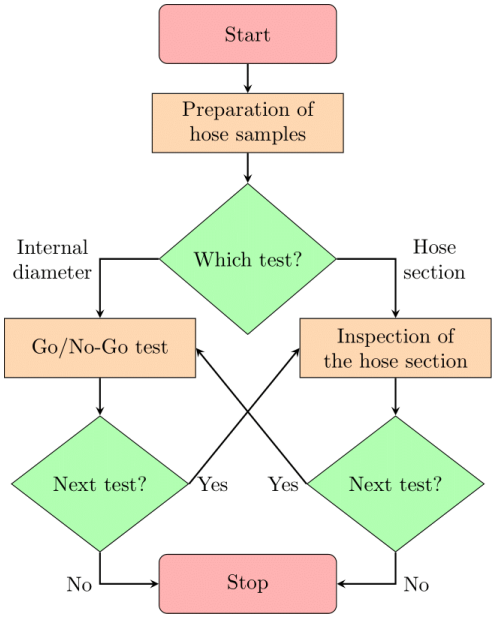
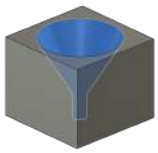


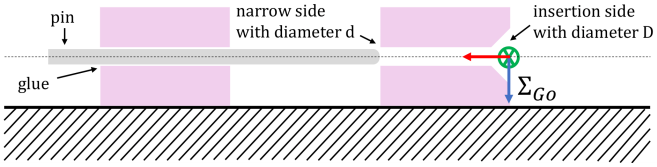
Figure 1.16. Medical hose quality inspection workflow.

A custom collector was designed to interface the robotic system with the extrusion line and to ensure reliable sample acquisition. The collector receives the hoses that are blown out from the extrusion line through a controlled air flow and temporarily stores them before the robotic pick-up.

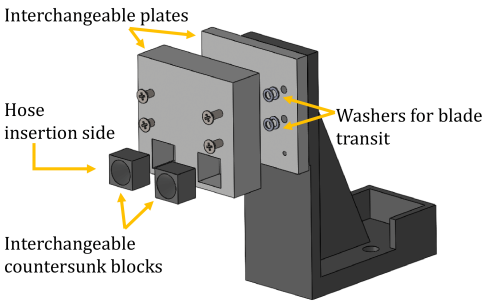
The sequence shown in Fig. 1.15 illustrates the complete pick-and-place procedure from the extrusion line with reference frame Σ_{EL} to the alignment support Σ_{SP}. The robot first collects the hose directly from the extrusion outlet (Fig. 1.15a), then transfers it toward the alignment fixture (Fig. 1.15b), and finally releases it onto the inclined alignment support (Fig. 1.15c). Once released, each hose slides down the sloped alignment support, denoted as Σ_{SP} in Fig. 1.14, until it reaches a mechanical stop that ensures a repeatable and well-defined resting pose. This setup allows a deterministic re-pick by the robotic gripper, guaranteeing consistent alignment between the hose and the reference frames of the robotic cell. Consequently, subsequent tasks, such as the Go/No-Go test and the section inspection, can be executed with accurate initial positioning, minimizing uncertainty in the overall workflow.



(a) Resin block.



(b) Go/No-Go test pins.



(c) Usage of resin block in cutter station.

Figure 1.17. Countersunk resin block and its applications in the Go/No-Go test and cutter station.

This preparation phase, corresponds to the initial stage of the diagram in Fig. 1.16. After this phase, the process is baranched into two independent inspection branches according to the specific testing requirements: one dedicated to the Go/No-Go test, which evaluates the internal diameter, and another dedicated to the section inspection, which analyzes the cut surface of the hose. Each branch can be executed individually or sequentially, depending on production needs, before the process returns to the initial state or terminates.

1.3.3 Countersunk resin block

Manipulation of medical hoses introduces significant uncertainty in positioning and contact interaction, due to their intrinsic flexibility and low stiffness. To overcome this issue, a *countersunk resin block* was developed as a passive mechanical aid, as shown in Fig. 1.17. The block features a conical cavity that guides the hose during insertion and constrains its motion once inside, effectively increasing its local rigidity. This simple yet effective design allows the robot to perform insertion tasks with higher accuracy, compensating for the unpredictable deformation of the hose. As illustrated in Fig. 1.17b, the resin block is used in combination with the Go/No-Go test pins, facilitating alignment and guiding the hose toward the proper contact region for internal-diameter evaluation. Moreover, the same design principle is applied at the cutting station (Fig. 1.17c) and during section inspection, where the block ensures stable positioning and prevents unwanted hose bending during the cutting and imaging phases.

Insertion inside the resin block

However, when the hose tip is significantly misaligned with the block opening, direct insertion may fail due to surface collisions. To address this limitation, a *blind search strategy* based on an Archimedean spiral was implemented to automatically locate the countersink entrance before starting the *Go/No-Go* test. The spiral trajectory follows the polar equation

$$\rho = b\,\theta, \qquad b = \frac{r}{2\pi}, \tag{1.6}$$

where r is the distance between two consecutive spiral turns, chosen equal to half the countersink diameter ($r = D/2$). At each angular step, the position is incremented according to

$$\theta_{k+1} = \theta_k + \Delta\theta, \tag{1.7}$$

where $\Delta\theta$ varies progressively to ensure uniform spacing between testing points. This condition guarantees that the distance between consecutive positions is always smaller than D, ensuring convergence toward the hole center within a finite number of attempts.

An example of the explored spiral path is shown in Fig. 1.18. At each testing point, the robot performs a short insertion movement and evaluates contact through tactile feedback. If contact is detected, the robot retracts and moves to the next point along the spiral until the cavity is successfully located. This preparatory step ensures robust and repeatable centering of the hose within the resin block, enabling accurate execution of both *Go/No-Go* and section inspection tasks.

The *Go/No-Go* test automates the verification of the internal diameter of deformable medical hoses by combining tactile sensing with a calibrated mechanical setup. The robot uses tactile feedback to assess whether the hose can be inserted into a reference pin or if contact occurs prematurely, indicating an undersized internal diameter. This approach allows the system to reproduce the manual inspection procedure while providing quantitative and repeatable measurements based on sensor data.

Each testing unit, shown in Fig. 1.17b, consists of a perforated resin block and a countersunk one, aligned and fixed at a proper distance, so that the pin, glued into the perforated block, reaches the narrow side of the countersunk block. The hose is inserted from the countersunk side, which facilitates alignment and stabilizes the interaction during insertion. Tactile indicators are monitored throughout the task to detect whether the hose smoothly fits onto the pin or becomes blocked due to a mismatch in internal diameter.

During the insertion process, the robot continuously estimates the contact centroid from the tactile data to monitor the interaction dynamics. Two thresholds are defined: the traveled distance s and the contact indicator C_I already explained in Eq.(1.3) and Eq.(1.4), which quantifies the displacement of the tactile centroid. The test logic relies on comparing these quantities against predefined thresholds.

In nominal conditions, the robot aligns the hose with the testing pin and performs a linear insertion. A *Go* test is considered successful when the hose reaches the target depth s_t without exceeding the contact threshold C_{It}:

$$\textit{Go test passed} \iff (s > s_t) \wedge (C_I < C_{It}). \tag{1.8}$$

Conversely, a *No-Go* test is passed when contact is detected before the insertion depth is reached:

$$\textit{No-Go test passed} \iff (s < s_t) \wedge (C_I > C_{It}). \tag{1.9}$$

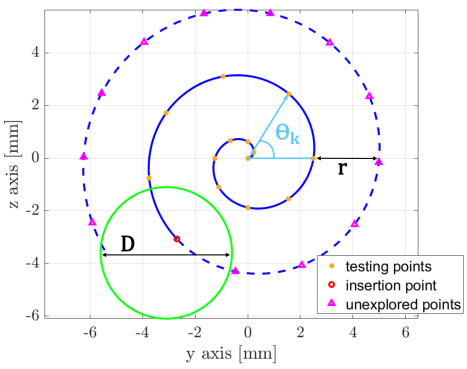


Figure 1.18. Spiral search strategy for blind insertion.

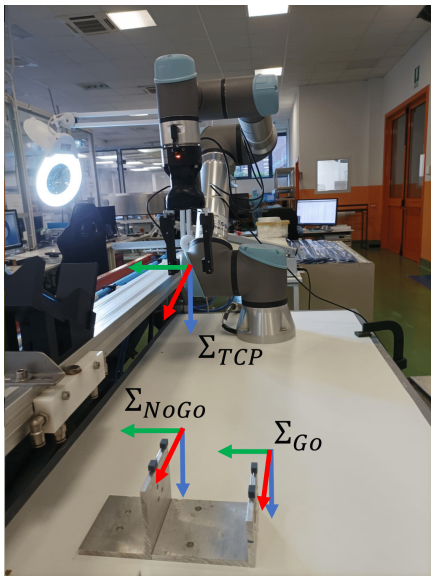


Figure 1.19. Go and No-Go test.

The overall diameter verification is considered successful only if both conditions are satisfied. This threshold-based approach enables a fully autonomous and tactile-driven evaluation of the hose internal diameter, ensuring repeatability and robustness even under small alignment or shape variations.

1.3.4 Cutting Machine and Vision-Based Inspection

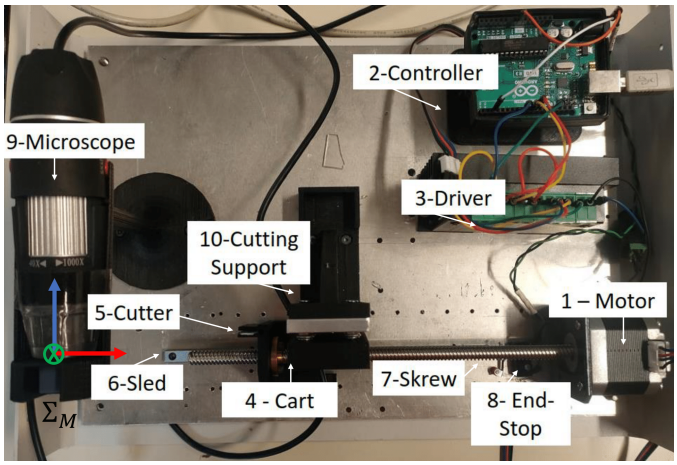


Figure 1.20. Cutting and microscope inspection station, top view.

The cutting machine, shown in Fig. 1.20, automates the manual and repetitive process of hose cutting, enhancing both productivity and operator safety. Its design prioritizes simplicity, cost-efficiency, and precision, resulting in a versatile and affordable tool for modern manufacturing. By integrating the cutting device with the robotic cell, hoses can be automatically cut at predefined positions retrieved from the extrusion line, ensuring clean and repeatable results.

The cutting system combines mechanical simplicity with programmable control. Its main components are:

- **NEMA 17 stepper motor** with lead screw (**SFK0801**) converting rotary motion into linear blade motion.

- **TB6600 driver** and **Arduino Uno** controller for motor actuation and communication with the robot.
- **Cutting cart** carrying a surgical blade, constrained by a linear sled.
- **Endstop sensor** for homing.
- **Support structure** (Fig. 1.17c) featuring interchangeable plates and resin blocks to accommodate hoses of different diameters.
- **Digital microscope** for post-cut imaging and quality analysis.

Cutting procedure

After the homing procedure, the robot inserts the hose into the support structure, possibly using the spiral search described in Sec. 1.3.3. The blade, actuated by the lead screw, and controlled by a stepper motor, performs a clean linear cut putting the blade in position according to the conversion formula:

$$steps_{mm} = \frac{steps_{rot} \times driver_{microsteps}}{screw_{pitch}}, \tag{1.10}$$

where $steps_{rot}$ is the motor step count per revolution, $driver_{microsteps}$ the microstepping factor, and $screw_{pitch}$ the lead screw pitch. After cutting, the hose is positioned in front of the microscope for visual inspection.

Automatic centering and focus with camera feedback

After the hose cutting, the prepared sample is automatically analyzed through computer vision to assess the quality of the cut surface. Due to the hose's flexibility and reduced size, achieving precise centering and focus within the microscope field of view is challenging. To overcome this, a vision-guided control strategy was developed, combining image-based feature extraction with robot motion feedback to autonomously align and focus the hose before inspection.

Mask Extraction from RGB Threshold Let p be a generic pixel of the image, with color vector

$$I(p) = \begin{bmatrix} I_r(p) \\ I_g(p) \\ I_b(p) \end{bmatrix}.$$

By defining lower and upper intensity bounds I_{min} and I_{max}, the mask extraction is performed as: Here I_{min} and I_{max} are RGB vectors and inequalities are intended component-wise.

$$\forall p \in I : \begin{cases} I_{min} \leq I(p) \leq I_{max} \Rightarrow 1, \\ I(p) > I_{max}, I(p) < I_{min} \Rightarrow 0, \end{cases} \tag{1.11}$$

where foreground pixels correspond to the hose region. This segmentation step isolates the blue tones of the hose from the background illumination, enabling robust feature detection even under non-uniform lighting.

Centroid Computation and Robot Alignment The image center is defined as

$$i_c = \frac{W}{2}, \quad j_c = \frac{H}{2}, \tag{1.12}$$

Algorithm 1 Microscope Tube Centering using Mask and Centroid

Initialize pixel tolerance ϵ and compute distance D_c
while $D_c > \epsilon$ **do**
  Extract mask (Eq. (1.11))
  Compute centroid (i_b, j_b) and image center (i_c, j_c)
  Calculate offsets $\Delta i = i_b - i_c$, $\Delta j = j_b - j_c$
  Move robot proportionally to $(\Delta i, \Delta j)$ using Eq. (1.15)
  Update $D_c = \sqrt{\Delta i^2 + \Delta j^2}$
end while

Algorithm 2 Microscope Focus using Laplacian-based Blur Evaluation

Initialize robot position (x_0, y_0, z_0) and focus metric $F_p = -\infty$
while $F_p < T_f$ **do**
  Capture image and compute F_c (Eq. (1.16))
  if $F_c > F_p$ **then**
    Move closer: $z = z + \Delta z$
  else
    Move further: $z = z - \Delta z$, reduce step $\Delta z = \alpha \Delta z$
  end if
  $F_p = F_c$
end while

where W and H are the image width and height. The centroid of the extracted binary mask is obtained from the image moments:

$$i_b = \frac{M_{1,0}}{M_{0,0}}, \quad j_b = \frac{M_{0,1}}{M_{0,0}}, \tag{1.13}$$

where $M_{m,n} = \sum_{p \in R} i^m j^n$ are the spatial moments of the binary mask.
The offset between (i_b, j_b) and (i_c, j_c) determines the correction vector

$$\boldsymbol{\Delta}_{xy} = \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} = \begin{bmatrix} i_b - i_c \\ j_b - j_c \end{bmatrix}, \tag{1.14}$$

which is converted into robot motion commands on the microscope xy-plane according to

$$\boldsymbol{\Delta r}_{xy} = k_p \, \boldsymbol{\Delta}_{xy}, \tag{1.15}$$

where k_p is a proportional gain that ensures smooth and stable convergence.

Blur Evaluation and Autofocus To achieve optimal focus, the system evaluates image sharpness using the variance of the Laplacian operator:

$$F = Var(L(x,y)), \qquad L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}, \tag{1.16}$$

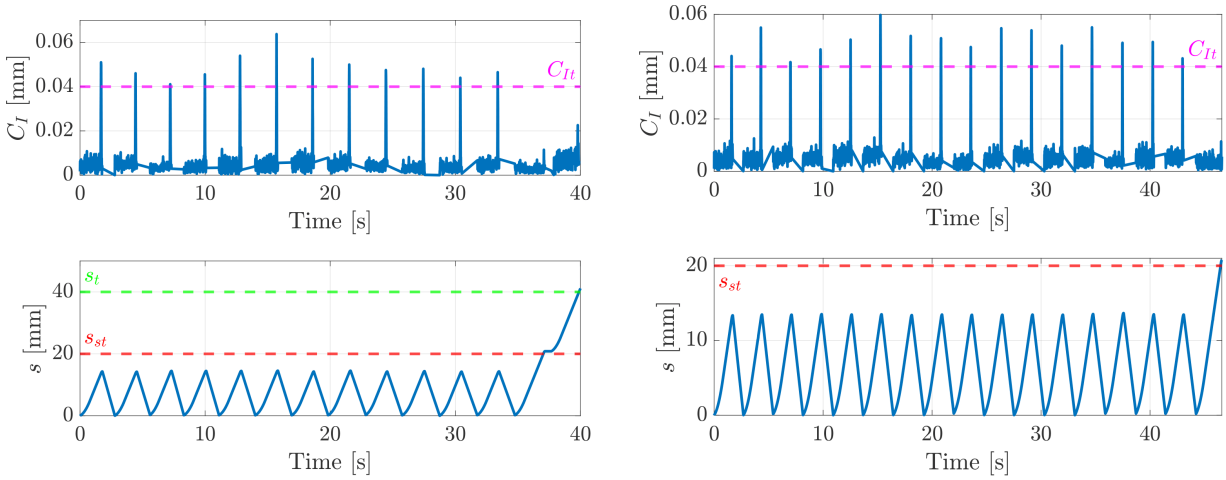
where a higher value of F indicates greater image sharpness. The robot iteratively adjusts its position along the microscope z-axis to maximize F, dynamically adapting the step size Δz to refine the focus when the metric improvement decreases.

Integrated Vision-Guided Control The two algorithms operate in closed-loop fashion: the vision module continuously provides positional and focus feedback, while the robot compensates in real time through incremental movements along the xy and z axes. This approach ensures that each hose sample is automatically centered and focused within the microscope view, minimizing operator intervention and guaranteeing repeatable, high-quality imaging for subsequent inspection and classification.

1.3.5 Results

All experiments were performed on hose samples with nominal inner diameter of 1.6 mm and various tolerances. The contact and distance thresholds were set to $C_{It} = 0.04$ mm and $s_t = 40$ mm, respectively.

Spiral Search Performance



(a) Spiral search for $D = 10$ mm hole.

(b) Spiral search for $D = 5$ mm hole.

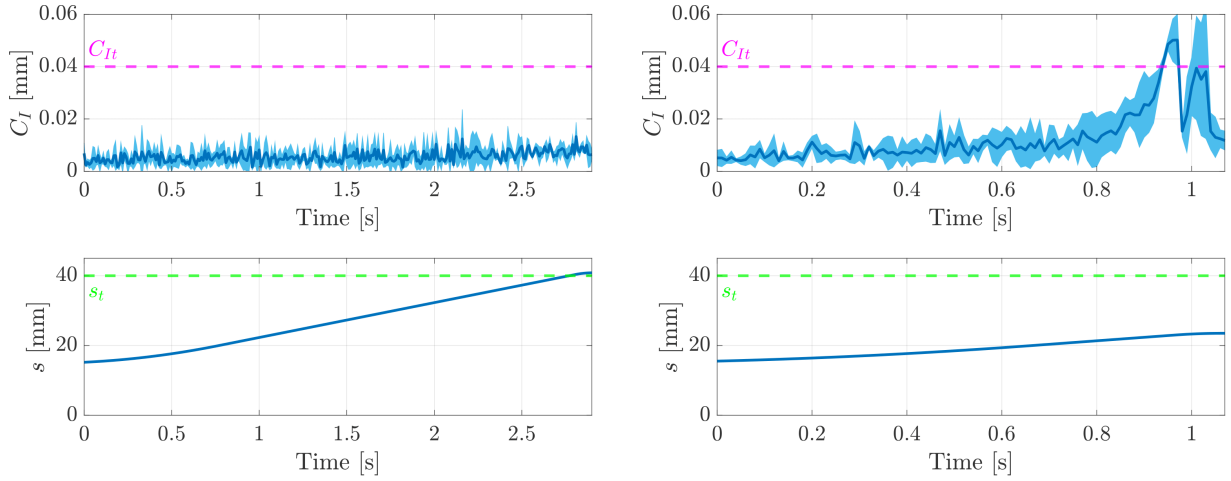
Figure 1.21. Spiral search results for different hole diameters.

To generalize the task, a spiral-based search algorithm was implemented to autonomously locate the countersunk hole before performing the *Go/No-Go* test. Experiments were conducted with two different countersinks, $D = 10$ mm and $D = 5$ mm. The impact detection threshold remained $C_{It} = 0.04$ mm, while the centering validation threshold was set to $s_{st} = 20$ mm. This parameter should not be confused with the insertion depth threshold s_t used in the *Go/No-Go* decision logic.

Figure 1.21a illustrates a successful *Go* test after 12 insertion attempts in the case $D = 10$ mm. The robot stops each time the contact threshold is reached, retracts, and moves to the next spiral point. After finding the hole, the hose is inserted successfully. For the smaller $D = 5$ mm block, the robot needed up to 16 attempts (Fig. 1.21b).

Go/No-Go Test Performance

The *Go/No-Go* experiments were first conducted under nominal conditions, i.e., assuming the hose tip was already aligned with the countersunk block. A block with $D = 10$ mm and $d = 1.6$ mm was used. Figure 1.22a shows the mean and standard deviation of the contact indicator C_I for ten passed *Go* tests, demonstrating that the chosen threshold C_{It} guarantees a sufficient safety margin. Similarly, Fig. 1.22b reports the results of ten passed *No-Go* tests, where the robot correctly detects contact before reaching the distance threshold s_t.



(a) Indicator (top) and distance (bottom) data during *Go* tests.

(b) Indicator (top) and distance (bottom) data during *No-Go* tests.

Figure 1.22. Results of *Go/No-Go* tests.

Vision-based Centering and Focus for Section Inspection

After cutting, the hose section is inspected under a digital microscope. Automatic centering and focusing are performed using the visual algorithms. Experiments were conducted on single-lumen (1.25 mm) and double-lumen (1.6 mm) hoses. The centering tolerance was set to $\epsilon = 10$ px, and the autofocus parameters to $T_f = 400$, $\alpha = 0.5$, $\Delta z = 10$ mm.

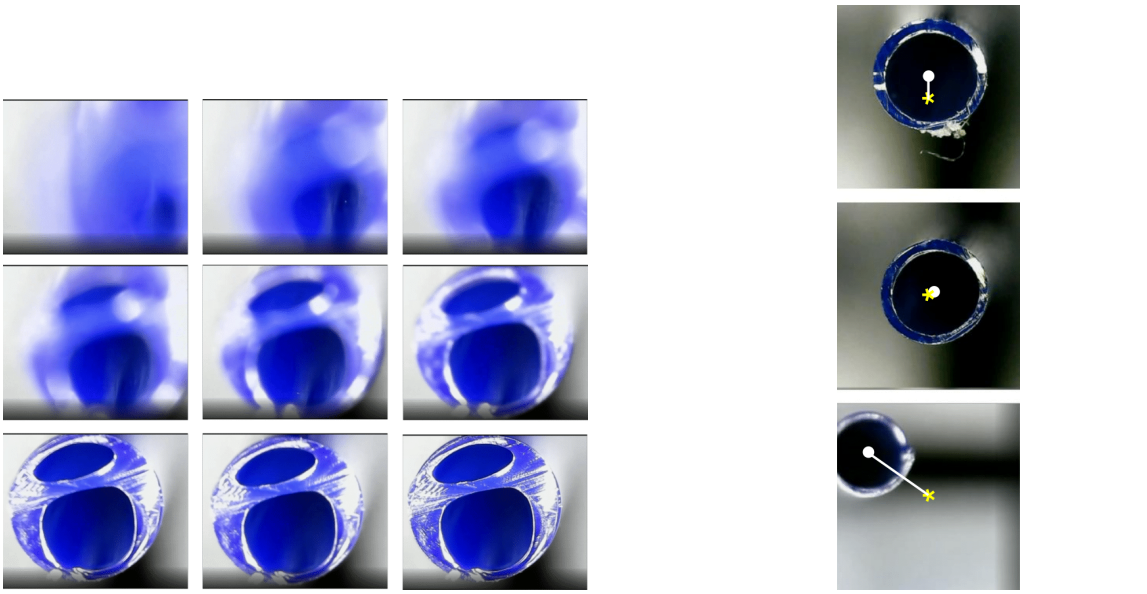
Figure 1.23a shows the centering and focusing sequence for a double-lumen hose, from the initial blurred frame to the final focused image ($F_c = 406.98$). Figure 1.23b reports typical results for single-lumen samples, including a correctly cut and focused hose and a failed case due to burrs on the edge, which distort the centroid estimation in Algorithm 1.

1.4 Tactile-Driven Cable Grasp through Hertzian Models and Bayesian Calibration

As shown in the previous sections (Sec. 1.2, 1.3) and in the literature [12, 17], mapping tactile voltages to tangential interaction forces has been explored with encouraging results. However, for tactile sensors described in Fig.1.1a and Fig.1.1b, a practical gap remains: there is still no way to map the tactile signals into an estimate of the normal contact load (or average pad pressure) that can be used online to decide how much to close/open the jaws and to regulate the clamping action on the grasped object. This section introduces a calibration and control framework that uses tactile sensing together with a Hertzian contact formulation as a model-based interpolator to learn the mapping between normal force and measured voltages for a cable/cylindrical to flat tactile contact. The resulting force–voltage relation is then used online to close the loop on grasping-force regulation in a parallel gripper. Model parameters are identified via Bayesian Optimization (BO), enabling calibration directly on the physical setup without requiring a large labeled dataset and providing a consistent voltage-to-force mapping suitable for control.

Knowing that the proposed framework can be reduced to a relation between the normal forces acting on the tactile pads, we aim to establish a link between these normal forces and the mean pressure measured on the tactile sensors. This allows us to leverage a contact model to estimate the mean pressure as well, and in future work to extend the approach toward estimating tangential forces.

We also acknowledge that our analysis may violate some of the strong assumptions underlying the contact model; nevertheless, we show that the Hertz-based formulation can still



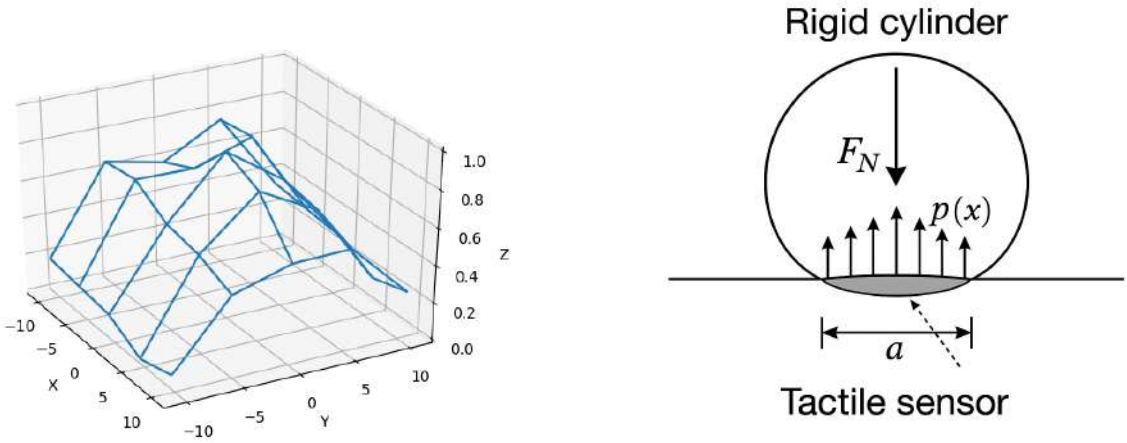
(a) Centering and focus sequence (double-lumen).

(b) Single-lumen cases of misalignments.

Figure 1.23. Results of automatic centering and focusing for hose section inspection.

serve as an effective interpolator for force estimation, even when those assumptions are only approximately satisfied.

1.4.1 Mathematical Hertz Theory for Contact Force/Pressure Estimation



(a) Tactile sensor voltage map during cable contact.

(b) Theoretical pressure distribution from Hertzian contact model.

Figure 1.24. Pressure distributions: experimental tactile readings and Hertzian theoretical model.

In this section, we present the mathematical framework employed to estimate the contact forces and parameters arising from the interaction between a cylindrical object (e.g., an electrical cable) and a tactile sensor.

In the continuous case, the total normal force can be described using Eq.(1.1) where in this case x denotes the coordinate along the direction perpendicular to the cylinder's axis, y represents the coordinate along the length of the cylinder. Given the tactile sensor described in Sec.1.1, Fig.1.1a, the integration bounds $-\frac{L_x}{2}$ and $\frac{L_x}{2}$ define the extent of the contact region along the x-axis, within which the pressure distribution $p(x)$ is nonzero.

We denote by $p(x_i, y_j)$ the pressure measured by the taxel located at the spatial coordinate (x_i, y_j), where $i = 1, 2, \ldots, N_x$ and $j = 1, 2, \ldots, N_y$ are the taxel indices along the x and y directions, respectively. If the taxels are uniformly distributed over the sensor area, the Eq.(1.2) can be used where Δx and Δy represent the uniform spacing between adjacent taxels along the x and y directions, respectively. Given the forces measured by the tactile sensor, we can estimate the contact mean pressure of the tactile using Eq.(1.5), which provides an average pressure value over the active taxels.

To relate the mean pressure to the contact mechanics, we employ Hertzian contact theory. For a cylinder in contact with a flat surface, the mean pressure p_{mean} can be expressed as:

$$F_N = \frac{1}{2}\pi a L p_{\max} \tag{1.17}$$

where F_N is the normal force, a is the half-width of the contact area, and L is the length of the cylinder in contact with the sensor. The half-width a can be derived from Hertzian contact mechanics as:

$$a = \sqrt{\frac{2F_N\left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}\right)}{\pi L\left(\frac{1}{R} + \frac{1}{R_s}\right)}} \tag{1.18}$$

where E_1, E_2 are the Young's moduli, ν_1, ν_2 are the Poisson's ratios, and R and R_s are the radii of curvature of the cylinder and the sensor surface, respectively. In our case, the silicone pad is modeled as a flat surface, hence $R_s = \infty$, which simplifies the denominator term to $\frac{1}{R}$. Assuming a Hertzian (semi-elliptic) pressure distribution (Fig. 1.24b), the relationship between maximum and mean pressure is $p_{\max} = \frac{4}{\pi} p_{\text{mean}}$. Combining this with Eq. (1.17), the mean pressure is related to the applied force and contact geometry by:

$$p_{\text{mean}}^{\text{Hertz}} = \frac{F_N}{2aL} \tag{1.19}$$

By combining Eqs.(1.2), (1.5), (1.19), and (1.18), we can establish a relationship between the tactile sensor readings and the contact parameters.

1.4.2 Bayesian Optimization for Parameter Identification

After defining the Hertzian contact model, the next step is to identify the unknown parameters (e.g., effective material properties and calibration factors) directly from tactile data. In our setting, only a limited amount of calibration data can be collected by repeatedly closing and opening the gripper on the cable, and the resulting measurements are noisy and only indirectly related to the underlying contact variables. Moreover, the number of parameters to estimate is small, and the resulting objective is generally non-convex and not readily differentiable. For these reasons, we adopt Bayesian Optimization (BO): it is particularly effective in low-dimensional parameter spaces, when function evaluations are expensive, data are scarce, and noise is present.

The first step is to place the tactile readings and the Hertzian model predictions into the same domain. To this end, we establish a relation between Eq. (1.2) and Eq. (1.17), so that tactile voltage readings can be mapped to an equivalent normal-load/pressure estimate. Using datasheet information and simple boundary conditions derived from the gripper behavior, we initialize this mapping with a linear guess.

Once tactile-derived pressures and Hertzian predictions are expressed in a consistent domain, we estimate the unknown parameters through optimization. We define $\theta = [E_1, E_2, \nu_1, \nu_2, k]$, where E and ν denote Young's modulus and Poisson's ratio of the two contacting materials, and k is a calibration factor accounting for the scale mismatch between Hertzian pressure and tactile outputs. The objective minimizes the discrepancy between the mean pressure estimated from

tactile data $p^{\text{Tact}}_{\text{mean}}$ in Eq.(1.5) and the Hertzian prediction $p^{\text{BO}}_{\text{mean}}$ using the parameters θ and the model in Eq.(1.19):

$$J(\theta) = \sum_{i=1}^{n} \left(p^{\text{BO}}_{i_{\text{mean}}} - p^{\text{Tact}}_{i_{\text{mean}}} \right)^2 + \max\left(0, E_{\text{sili}} - E_{\text{cable}}\right). \tag{1.20}$$

The penalty term enforces physical consistency, ensuring that the compliant silicone pad remains softer than the cable material. The scaling factor k is included as an additional decision variable with a bounded search range chosen to keep predicted and measured pressures on comparable scales.

BO searches for θ by modeling the unknown objective $J(\theta)$ with a probabilistic surrogate, typically a Gaussian Process (GP), which provides both a mean prediction and an uncertainty estimate. At each iteration, an acquisition function (e.g., Expected Improvement or Upper Confidence Bound) selects the next candidate θ by trading off:

- **Exploration:** sampling regions with high uncertainty to improve the surrogate model;
- **Exploitation:** sampling regions predicted to yield low cost to refine the optimum.

The cost is evaluated at the selected point, and the GP is updated. Iterating this procedure enables efficient identification of the parameters with far fewer trials than grid search, which is critical given the limited dataset and the need to keep calibration time short.

1.4.3 Mapping Forces to Gripper Commands

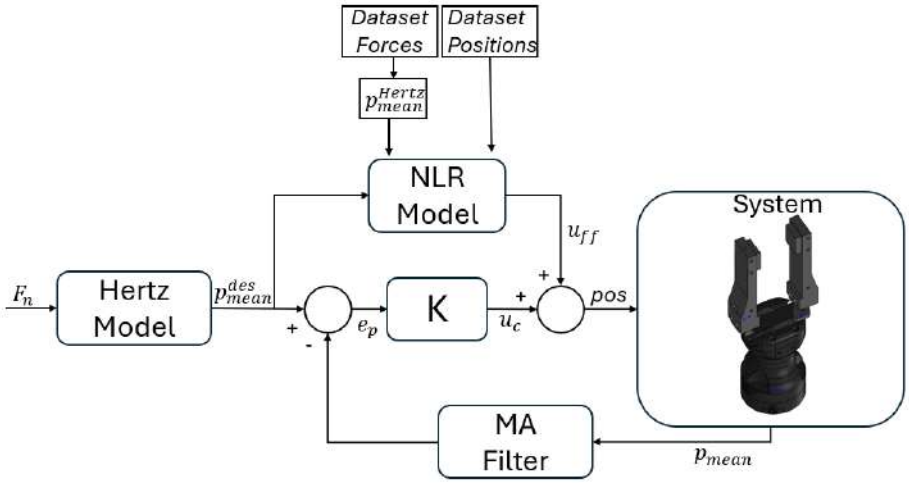


Figure 1.25. Block diagram of the closed-loop force control system using the Hertzian contact model for force estimation.

The estimation procedure described in Sec. 1.4.2 can be affected by material inconsistencies arising from 3D-printing imperfections and sensor noise, all of which contribute to irregular pressure distributions as shown in Fig. 1.24a. To mitigate these effects, the proposed control architecture incorporates a feedback component that compensates for deviations between the desired and the measured average pressures.

An initial estimate of the gripper closure required to achieve a target contact force is obtained by mapping pressure signals to gripper positions using the sampled dataset. Once the tactile forces are aligned in the same domain as the measured pressure distribution, the relation can be established. As illustrated in Fig. 1.26, this relation is strongly nonlinear and cannot be reliably captured with a linear model. For this reason, a sixth-order polynomial regressor is employed

Symbol	Description (Unit)	Parameter Range
E_{sili}	Silicone Young's modulus (Pa)	$[1 \times 10^5, 1 \times 10^8]$
ν_{sili}	Silicone Poisson's ratio (—)	[0.45, 0.49]
E_{copper}	Copper Young's modulus (Pa)	$[5 \times 10^7, 1.3 \times 10^{11}]$
ν_{copper}	Copper Poisson's ratio (—)	[0.31, 0.36]
k	Scaling factor for pressures (—)	$[1, p_{\text{max-derived}}]$

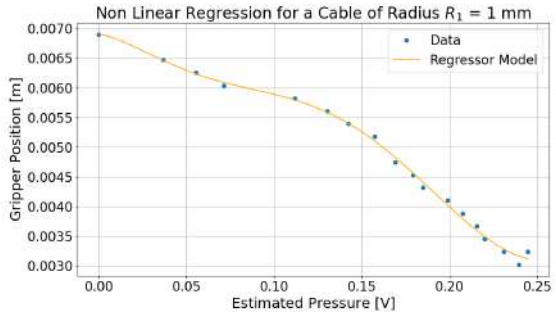
Table 1.1. Parameters used in Bayesian optimization.

Radius [mm]	k	ν_{sili}	ν_{cable}	E_{sili} (Pa)	E_{cable} (Pa)
1	9414.13	0.476	0.32	4.5×10^6	5.33×10^{10}
2	9961.99	0.482	0.348	2.33×10^7	4.44×10^{10}
3	4349.26	0.456	0.352	1.92×10^6	6.82×10^9

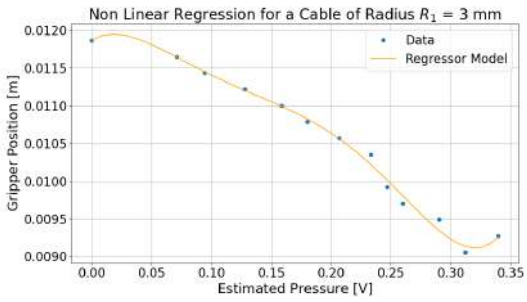
Table 1.2. Optimized parameters for different cable radii.

to map the desired pressure to the corresponding gripper position. This feedforward model is then complemented by a feedback loop, ensuring robust tracking of the target contact force.

The resulting control architecture (Fig. 1.25) combines the feedforward regressor, which maps the desired pressure predicted by the Hertzian model to a nominal gripper position u_{ff}, with a proportional feedback loop. The latter compensates for modeling inaccuracies by correcting the pressure error e_p.



(a) Pressure-to-position mapping for a 1 mm diameter cable.



(b) Pressure-to-position mapping for a 3 mm diameter cable.

Figure 1.26. Nonlinear regression models mapping desired pressure to gripper position for different cable diameters.

1.4.4 Experimental Validation

This section is organized into three parts to validate the proposed framework.

First, we describe the BO procedure used to identify the optimal parameters of the Hertzian contact model from tactile data.

Next, we present two experiments inspired by a bimanual regrasping scenario. The first assesses repeatability, requiring the robot to align the cable while closing the gripper to a reference force.

The second task is a grasp-and-lift experiment, in which the robot must apply the minimum force required to lift an object without slippage, thereby evaluating the controller's ability to regulate contact forces for stable manipulation. Since a cylindrical force sensor is not available for direct contact measurement, the assessment relies on the underlying physical model.

Bayesian Optimization Results

To validate the model across different scenarios, we benchmark electrical cables of radii R of 1 mm, 2 mm, and 3 mm, at a fixed gripper-interface contact length L that is the contact length between the cable and the gripper finger, depending on the pad used.

Starting from the search space defined in Tab. 1.1, we apply BO to identify the optimal parameters for each cable configuration. The optimization was executed over 200 iterations on

a 12th-generation Intel Core i7-12700 CPU. The full optimization procedure takes approximately 5 minutes to complete.

Once the cost function has been minimized, typically reaching residuals on the order of 10^{-2} as shown in Fig.1.27, the resulting parameters are stored in a lookup table (Tab.1.2) keyed by cable radius, so they can be reused without re-optimization and used online; the trend of the estimated pressure is then compared with the tactile-measured one in Fig.1.28. As shown in Tab. 1.2, BO addresses a nonlinear problem in a five-dimensional parameter space, comprising the Young's modulus and Poisson's ratio of both materials and a pressure scaling factor. In such an underconstrained setting, multiple parameter combinations can achieve comparable cost values, reflecting the continuous nature of the solution space. Nonetheless, the identified solutions remain physically consistent and respect the expected mechanical hierarchy (i.e., the cable material is stiffer than the compliant gripper pad) thus confirming the coherence of the cost function and the optimization procedure.

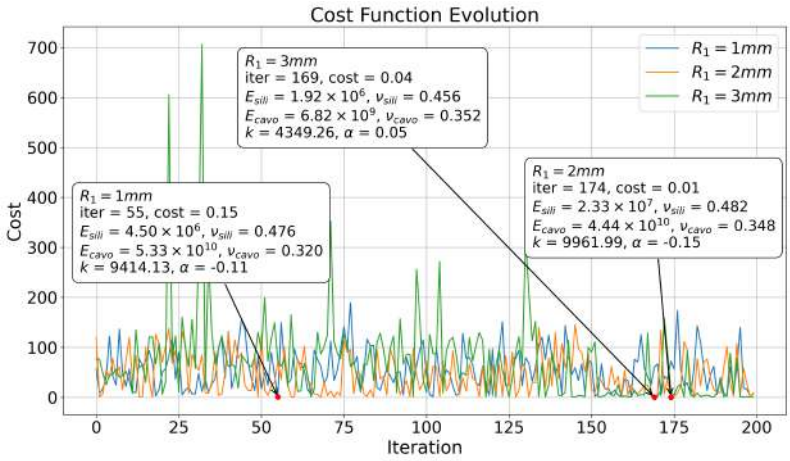


Figure 1.27. Evolution of the cost function showing convergence trends and optimal parameters for different cable radii.

Cable Alignment Experiment

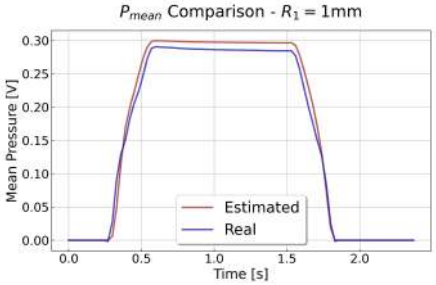
The first experiment evaluates the repeatability of cable alignment using the proposed force controller, evaluating the performance of the nonlinear regression model employed to predict the gripper closure required to achieve a specified contact pressure and presented in Fig.1.26. The evaluation is designed to address two main aspects:

- Accuracy — whether the gripper closure predicted by the regressor leads to a contact pressure consistent with the desired target force.
- Repeatability — whether the system can consistently achieve the target pressure across multiple trials, despite potential variations in initial cable positioning.

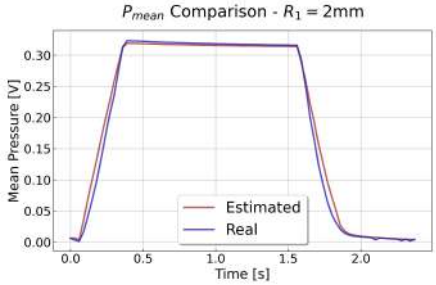
The aim of this study is therefore to assess the regressor's feedforward accuracy while simultaneously leveraging closed-loop tactile correction.

The experimental procedure is as follows. For a given desired clamping force, the corresponding contact pressure is first estimated using the Hertzian contact model. This target pressure is then provided as input to the nonlinear regressor, which produces an initial estimate of the gripper aperture. The gripper executes this closure, after which tactile feedback is incorporated in a closed-loop manner to refine the prediction

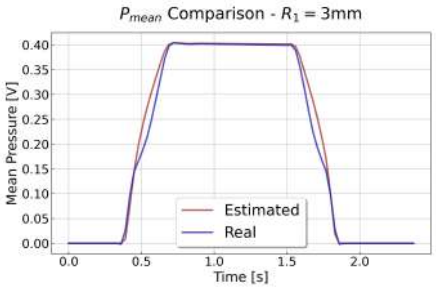
A representative sequence of the task is shown in Fig. 1.30, illustrating a dual-arm manipulation scenario where the robot must regrasp a cable after correcting its pose. Once the regrasp



(a) $R_1 = 1$ mm

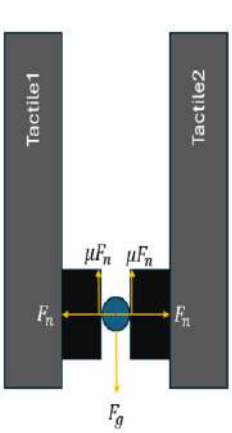


(b) $R_2 = 2$ mm



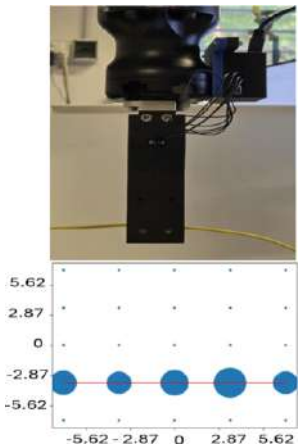
(c) $R_3 = 3$ mm

Figure 1.28. Comparison between Hertz-estimated and ground-truth tactile-measured mean pressure signals across three cable radii (R_1, R_2, R_3). The estimated curves are obtained using the optimized parameters from Table 1.2.

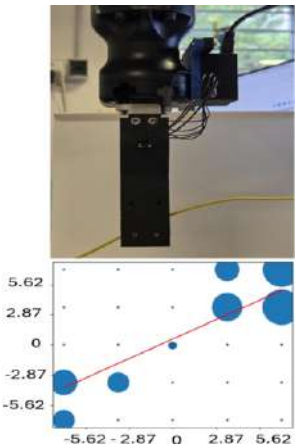


(a) Tactile clamp force distribution.

(b) Clamping of a cable.



(c) Cable aligned.



(d) Cable not aligned.

Figure 1.29. Tactile-based cable clamping: theoretical and experimental perspectives, and visual estimation of cable-in-hand alignment.

point is identified, the gripper is positioned and commanded to close according to the regressor's prediction, securing the cable.

After closure, tactile sensing is used to assess cable alignment within the gripper. The tactile map is approximated by a weighted linear regression, where each taxel is weighted by its voltage to emphasize high-pressure regions. From the regression, the slope provides the cable's angular deviation around the gripper normal (Fig. 1.29d), while the intercept indicates its lateral offset along the pad (Fig. 1.29c). Based on these estimates, corrective motions are computed in the gripper's local frame, consisting of a rotation about the normal axis and a translation along the pad surface to re-center the cable. The process is repeated until both deviations fall below predefined thresholds, indicating successful alignment.

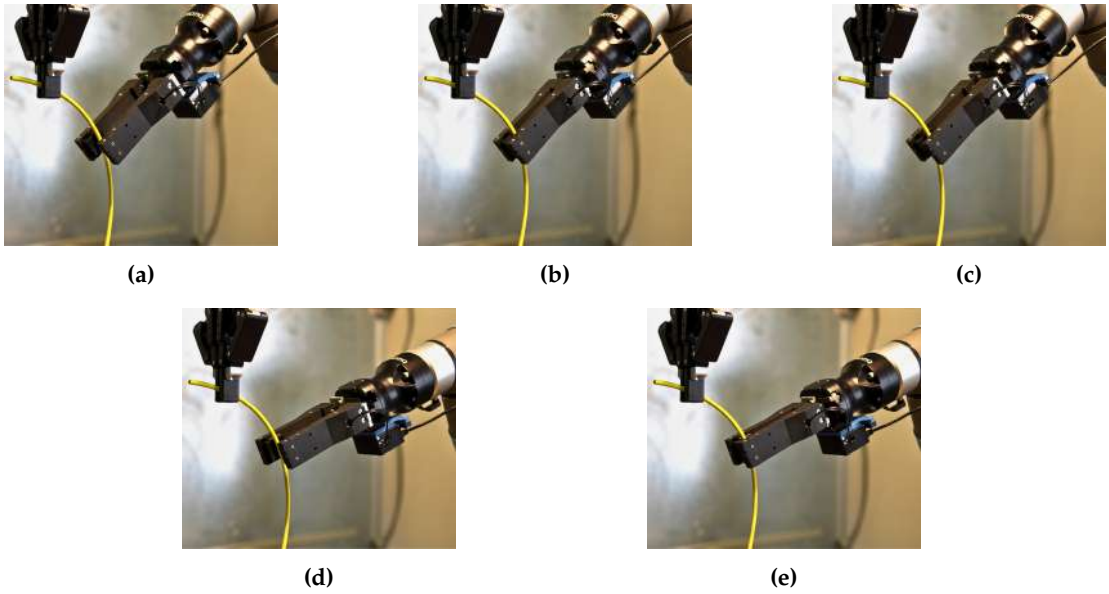


Figure 1.30. Sequence showing the gripper aligning and grasping the cable using tactile feedback. In case of misalignment, the gripper rotates until a symmetric pressure distribution is detected.

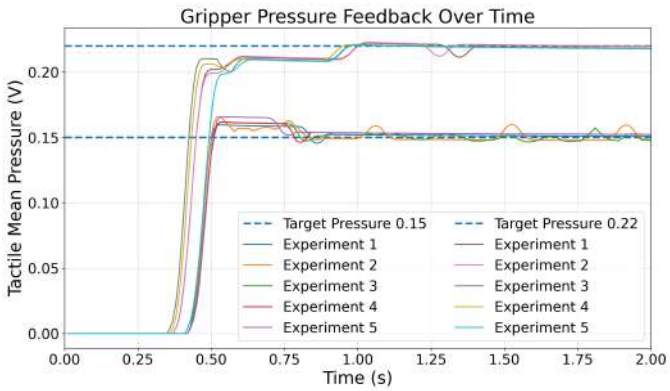
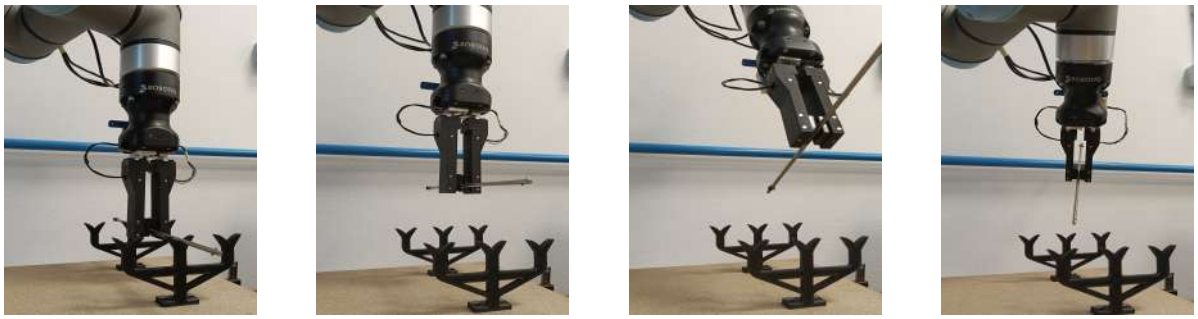


Figure 1.31. Measured pressure during the cable alignment task for different target pressures. The system consistently achieves the desired pressure with minimal error.



(a) Grasp **(b)** Rise **(c)** First Rotation **(d)** Second Rotation

Figure 1.32. Sequence showing the gripper aligning and grasping the cable using tactile feedback. In case of misalignment, the gripper rotates until a symmetric pressure distribution is detected.

Table 1.3. Success rate (%) for different clamping force levels across cylindrical objects of varying radii and lengths.

Object	Radius	Length Range	P_{mean}	P_{mean} + 10%
Cable 1	1.0 mm	0.50–0.90 m	80%	100%
Cable 2	2.0 mm	0.70–1.10 m	76%	100%
Cable 3	3.0 mm	0.60–0.90 m	95%	100%
Al. Tube	2.5–3 mm	0.30–0.35 m	60%	100%

The effectiveness of the combined feedforward–feedback strategy is shown in Fig. 1.31, where measured pressures closely match the desired profiles. Despite some uncertainty from the feedforward regressor, the mean error remains within 0.01–0.02 V, confirming both accuracy and repeatability.

Tests with cables of radii $R = 2$ mm and $R = 3$ mm show successful alignment within five corrective steps, demonstrating robustness to initial misalignment and sensor variability. Residual discrepancies with the Hertzian model arise mainly from uncertainties in the contact length L, which even in small variations significantly affect the pressure distribution.

Cable Grasp-and-Lift Experiment

Since a cylindrical force sensor was not available for this experiment, we validated the effectiveness of our tactile-informed clamping strategy through a series of grasping trials designed to resist gravitational slip while using the minimum required clamping force. In the absence of a cylindrical force sensor, we validated our tactile-informed clamping strategy by performing grasping experiments that quantify the minimum normal force needed to prevent gravity-induced slip. Given the object's weight F_g, and assuming Coulomb friction at both contact interfaces, the total frictional force is proportional to the normal force applied by each finger and the effective coefficient of friction μ between the silicone skin and the object surface.
To prevent the object from slipping under gravity, the frictional force must be greater than or equal to the gravitational load. This leads to the stability condition:

$$F_n \geq \frac{F_g}{2\mu}, \tag{1.21}$$

where F_n is the normal force exerted by a single finger. This inequality defines the theoretical minimum clamping force required to prevent slippage under idealized assumptions. To evaluate repeatability and robustness under uncertainty and sensor noise, each trial re-executed the full pipeline: the tactile dataset was re-acquired, and both the Hertzian contact model and

the nonlinear regressor were re-optimized. The robot then grasped the object with the nominal clamping force predicted by the tactile-based model, corresponding to the minimum force required to prevent slippage under gravity. After grasping, the object was lifted vertically and subjected to a short arm motion sequence (Fig. 1.32), performed at low speed to isolate gravitational effects and minimize dynamic contributions.

A trial was considered successful if the object remained secured throughout the sequence, and a failure if slippage occurred. Results are reported in Tab. 1.3. To further test robustness, the experiment was repeated with a 10% increase over the maximum clamping force observed in the dataset; in this case, no object slipped, confirming that the tactile-informed estimate was close to the true clamping requirement.

1.5 Cable Warehouse

In electrical cabinet assembly, the wiring phase remains one of the least automated stages, despite the extensive automation achieved in cable production. Modern machines such as the Komax ZETA 640 (Fig. 1.33) can fully automate cutting, stripping, and labeling, yet cable collection and organization are still performed manually, introducing variability and inefficiencies that limit full process integration.

To address this gap, this section proposes a robotic solution with future aim to connect cable production with automated wiring. The entire process is illustrated in Fig. 1.34, where blocks show the steps from cable fabrication to robotic wiring (in red since they are not part of this work). A dedicated gripper retrieves each cable directly from the production machine and deposits it into a structured warehouse composed of modular clips optimized through finite element analysis. This approach enables deterministic storage and retrieval without vision-based identification, improving reliability, repeatability, and throughput while keeping the system cost-effective and scalable. Overall, the proposed application demonstrates how robotic design

(a) Komax ZETA 640 cable production machine.

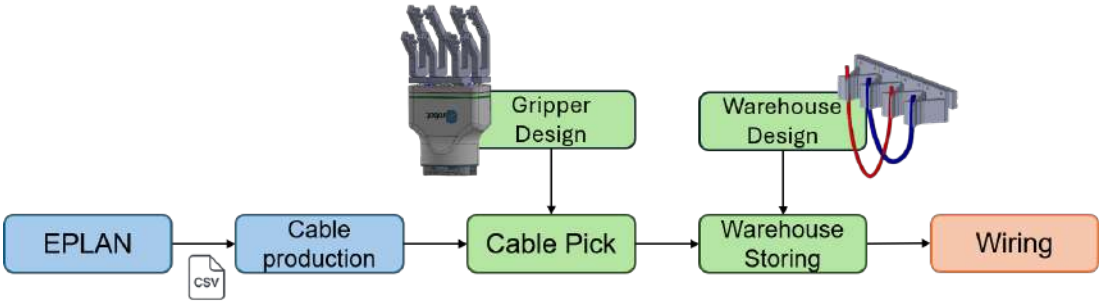
(b) End-line KomaxZ640, its gripper with the final cable product.

Figure 1.33. Komax ZETA 640 cable production machine with the robotic warehouse integrated.

can bring structure to unstructured tasks, transforming a chaotic post-production stage into a predictable and traceable process. By integrating mechanical design, control logic, and computational modeling, the system lays the foundation for end-to-end robotic automation in cable assembly and flexible manufacturing environments.

1.5.1 System Overview

The automation pipeline for cable handling, illustrated in Fig. 1.34, spans from the digital design of the wiring layout to the physical organization of cables within a robotic warehouse. The process begins with EPLAN, which generates detailed electrical schematics and exports a comprehensive CSV (see Fig. 1.34b) containing all relevant cable information, including origin



(a) Process pipeline.

	ORIGINE	Marking END 1	DESTINAZIONE		Marking END 2		Section (mm²)	Color			Lenght	
N. Pagina	Zona	Componente:pin	N. Pagina	Zona	Componente:pin	Numero filo	Sezione	Colore	Tipo pot.	Nome pot.	L filo (mm)	Traccia collegamenti
=FF100+04	PUL1	-XC0422.00:PE	=FF100+80	PUL1	-X50:100:b1	0422.01PE	1,5	BU	PE	PE	367	
=FF100+03	PUL1	-SB0300.00:14	=FF100+03	PUL1	-X50:31.2:b2	0300.01	1	WH	+	24VDC	999	=-167;=-168;=-174;=-2
=FF100+03	PUL1	-SB0300.00:24	=FF100+03	PUL1	-X50:31.1:b1	0300.02	1	TQ	+	24VDC	973	=-167;=-168;=-174;=-2
=FF100+03	PUL1	-X50:21.1:b1	=FF100+03	PUL1	-SE0305.00:11	0305.01	1	VT	+	24VDC	1012	=-CN502;=-CN501;=-2
=FF100+03	PUL1	-X50:21.2:b2	=FF100+03	PUL1	-SE0305.00:21	0305.02	1	WH	+	24VDC	1059	=-CN502;=-CN501;=-2
=FF100+03	PUL1	-SE0305.00:32	=FF100+03	PUL1	-X50:22.1:b1	0305.03	1	TQ	+	24VDC	1015	=-167;=-168;=-174;=-2
=FF100+80	PUL1	-X50:10:a2	=FF100+03	PUL1	-SE0305.00:31	12L+	1	RD	+	24VDC	874	=-CN503;=-CN501;=-2
=FF100+03	PUL1	-SE0305.00:31	=FF100+04	PUL1	-SB0421.00:13	12L+	1	BK	+	24VDC	286	=-173
=FF100+04	PUL1	-SB0421.00:13	=FF100+04	PUL1	-SB0421.01:11	12L+	1	RD	+	24VDC	167	=-173
=FF100+03	PUL1	-SB0300.00:13	=FF100+04	PUL1	-SB0421.01:11	12L+	1	BK	+	24VDC	129	=-173
=FF100+03	PUL1	-SB0300.00:13	=FF100+03	PUL1	-SB0300.00:23	12L+	1	RD	+	24VDC	98	=-173

(b) Example of cable production layout in Excel.

Figure 1.34. Overview of the cable warehouse system, from cable production to robotic wiring.

and destination points, numbering, cross-section, color, and length. This dataset defines the exact production sequence and serves as a direct interface between digital design and automated manufacturing.

Cable production is performed by the Komax ZETA 640, which automatically cuts, strips, crimps, and labels each cable according to the CSV specifications. At the end of this phase, the cables must be collected and organized in the same order in which they are produced, ensuring traceability for subsequent wiring operations. To accomplish this, a robotic system equipped with a custom-designed gripper (described in Sec. 1.5.2) transfers each cable directly from the Komax output to a structured warehouse. The warehouse itself, composed of modular clips detailed in Sec. 1.5.3, allows deterministic storage and retrieval, eliminating the need for visual identification and ensuring consistency across the production workflow.

This integrated process, from digital data to robotic storage, transforms an unstructured post-production phase into a fully traceable and automated pipeline, paving the way toward seamless robotic wiring operations.

1.5.2 Gripper Design

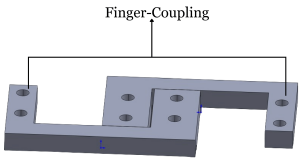
The robotic gripper has been specifically designed to automate the cable transfer from the Komax ZETA 640 machine to the structured warehouse. Its configuration ensures reliable grasping of cables with varying diameters while maintaining modularity, mechanical robustness, and compatibility with standard industrial grippers. The complete setup is shown in Fig. 1.35.

Adapter Mechanism

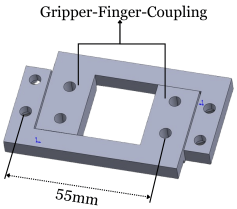
The gripper employs a modular adapter that allows the use of custom fingers with different commercial parallel grippers. In this application, the adapter has been tailored for the OnRobot 2FG7 model. Each adapter is composed of two mirrored parts, each supporting a pair of

(a) Robotic gripper for cable handling.

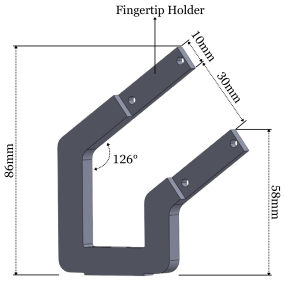
(b) Cad of the robotic gripper



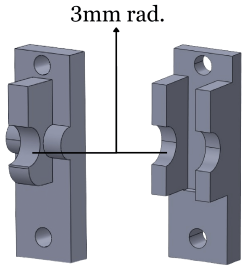
(c) Finger slider in its open position.



(d) Finger slider in its closed position.



(e) Holder for the fingers.



(f) Fingers for cable grasping.

Figure 1.35. Detailed views of the robotic gripper designed for cable handling in the warehouse system.

fingers. A primary design requirement is the synchronized motion of all four fingers (two active (*Gripper-Fingers*) directly actuated by the gripper sliders and two passive (*Fingers*) moving through mechanical coupling.

The adapter design produces an inverted kinematic behavior compared to a standard parallel gripper: when the jaws close, the fingers open, and vice versa. This motion ensures the correct spacing for grasping the cables as they are released from the Komax system. The fixed 55 mm distance between the mounting holes corresponds to the spacing of the Komax release gripper, guaranteeing alignment and avoiding collisions throughout the full stroke of the jaws.

Fingers

The fingers connect the adapter to the fingertips and are angled at 126° to optimize alignment with the cable output of the Komax ZETA 640. This geometry facilitates insertion into the workspace without interference and maintains sufficient cable rigidity during the transfer phase. The spacing between the two fingertip holders is calibrated to allow the robot to position itself between the Komax release jaws, ensuring precise and stable grasping.

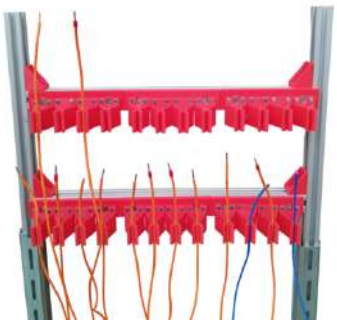
The finger modules are interchangeable, enabling quick adaptation to different cable geometries or gripper configurations. This modularity increases system flexibility while preserving mechanical integrity and alignment accuracy.

Fingertips

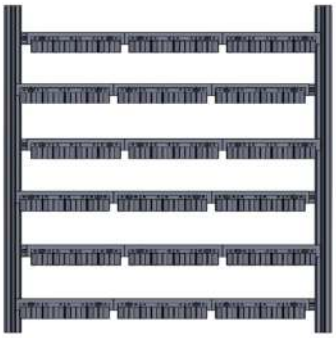
The fingertips are designed to handle cables of various diameters (0.5–6 mm) without damaging or deforming them. Each pair of fingertips forms an adaptive cylindrical interface that evenly distributes grasping forces, maintaining local rigidity while minimizing slip. The symmetrical design allows the tips to intersect slightly during closure, providing a stable and self-centering

grip on cables of different sizes. This configuration ensures reliable cable transfer while maintaining compliance necessary for deformable object manipulation.

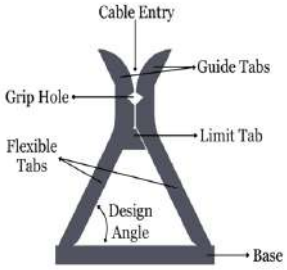
1.5.3 Warehouse Design



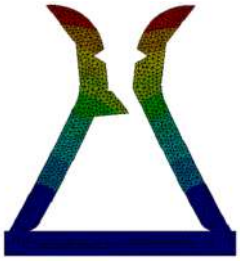
(a) Modular warehouse structure real implementation.



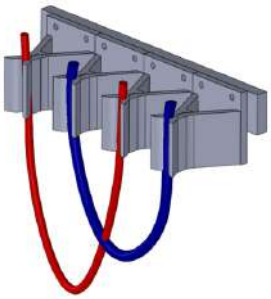
(b) CAD of the modular warehouse structure.



(c) Front view of a single clip.



(d) FEA analysis of a clip under 3 N load.



(e) Example of cable arrangement in the warehouse.

Figure 1.36. Detailed views of the modular warehouse structure for cable storage and organization.

The warehouse system was developed to provide a structured, modular, and cost-effective solution for cable storage. As shown in Fig. 1.36, the setup consists of multiple rows of clips designed to hold cables in an organized and easily accessible layout. The system ensures deterministic cable placement and retrieval while remaining adaptable to different workspace configurations.

Storage Concept

Each storage unit, referred to as a couple of *clips* (Fig.1.36e), is designed to hold one cable end. Since cables are stored by both ends and the robotic gripper fingers are spaced at a fixed distance of 55 mm, a pair of clips is required to secure each cable. To maximize density, cables are stored in pairs using four clips combined into a single modular block (see Fig. 1.36e). This configuration guarantees both compactness and mechanical stability during cable insertion and retrieval.

The modular nature of the warehouse allows flexible scaling according to production needs, while its low manufacturing cost enables rapid prototyping and reconfiguration.

Clip Design

The clip (Fig. 1.36c) was engineered to enable smooth robotic insertion and removal of cables while ensuring secure retention. It features two flexible tabs whose geometry determines the compliance during deformation. Guide tabs assist alignment during insertion, while a central

Clip	Mean [N]		Min [N]		Max [N]	
	2 mm	4 mm	2 mm	4 mm	2 mm	4 mm
48°	36.5	37.1	34.6	33.8	41.2	40.5
65°	21.0	21.2	19.5	19.3	22.0	22.5
77.7°	6.3	6.7	5.3	5.7	8.4	8.0

Table 1.4. Comparison of clip configurations for 2 mm and 4 mm cable diameters.

N° Tests	Clip	Success [%]	Misalign. [%]
40	48°	20	0
40	65°	100	7.5
40	77.7°	80	30

Table 1.5. Success rate and alignment performance.

grip hole increases friction for thinner cables. A limit tab prevents excessive cable displacement, maintaining precise positioning within the warehouse.

1.5.4 Experimental Validation

Finite Element Analysis for design choices

Given the fixed spacing between the gripper fingers, the base length of the clip cannot be modified; thus, the key design variable is the angle between the clip base and its flexible tabs. This parameter directly affects compliance and insertion force. Finite Element Analysis (FEA) was performed on ABS 3D-printed prototypes to determine the optimal geometry (Fig. 1.37). Simulations assessed tab deformation under varying angles (48°, 65°, 77.7°) and applied forces, complemented by experimental tests using a force-torque–equipped robot.

As summarized in Tab. 1.4 and Tab. 1.5, the clip with a 65° angle achieved the best trade-off between insertion force and retention stability, reaching a 100% success rate with minimal misalignment. This configuration was therefore selected for the final warehouse assembly.

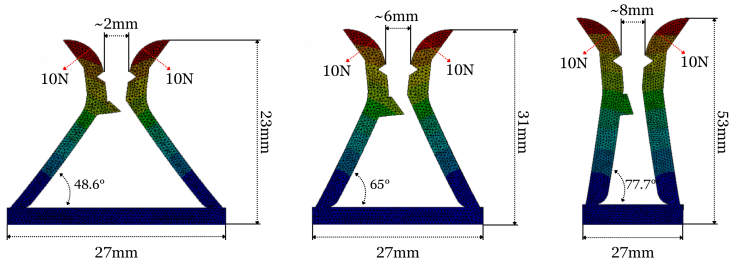
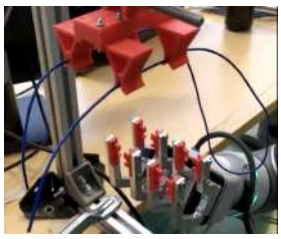


Figure 1.37. Finite Element Analysis (FEA) of the clip design under load, used to optimize the angle between the base and flexible tabs for optimal performance.

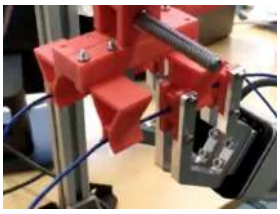
Robotic Integration

The experimental campaign was designed to validate the functionality and reliability of the automated warehouse system for cable storage, using the clip design at 65° as described in Sec. 1.5.3. The warehouse filling process is based on a matrix structure that defines the position of each cable within the storage layout. Following a Last-In, First-Out (LIFO) strategy, the robot iteratively populates this matrix by placing cables in pairs: two cables are positioned one on top of the other before proceeding to the next storage block. This arrangement prevents interference between cables, optimizes space utilization, and facilitates organized retrieval during the unloading phase (see Fig. 1.36e).

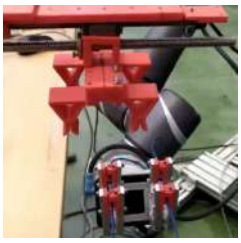
Once the reference frame and the initial placement point are established, the robot sequentially fills the warehouse, adjusting its position according to the predefined matrix offsets until all clips are occupied. This method ensures consistent cable placement, minimizes potential collisions, and maintains precise traceability of stored elements.



(a) Robot approaches the fake Komax cart to pick a cable.



(b) Cable picking from the fake Komax cart.



(c) Robot leaves the fake Komax cart after picking.

(d) Robot approaches the warehouse.

(e) Cable placement inside the warehouse.

(f) Robot leaves the warehouse after placement.

Figure 1.38. Complete robotic workflow for cable manipulation: (a–c) cable picking from the fake Komax cart, and (d–f) cable placement into the warehouse.

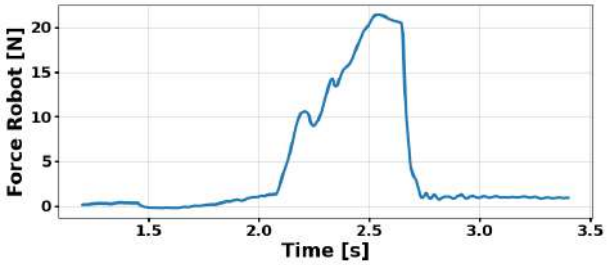


Figure 1.39. Force profile during cable insertion into the clips.

As shown in Figs. 1.38a, 1.38b 1.38c, the experimental setup aimed to replicate realistic working conditions. An Omron TM900 collaborative robot, equipped with an integrated force sensor, was controlled via an Omron PLC. Since direct communication with the Komax control system was not available, a dedicated prototype was developed to emulate the Komax cable cart. This prototype includes a stepper-driven linear guide that reproduces the original system's dimensions and operational sequence, enabling realistic pick-and-place trials.

The results demonstrated that the robotic gripper can consistently grasp and manipulate cables with diameters ranging from 0.5 mm to 6 mm, adapting to different stiffness and surface properties.

During the insertion phase shown in Figs. 1.38d, 1.38e, 1.38f, force measurements revealed that when the applied force exceeds the threshold values defined in Tab. 1.4, the system correctly flags a failed insertion, prompting operator verification. For instance, during tests with a 2 mm cable, the insertion force increased sharply upon initial contact, peaking at approximately 22.1 N (see Fig. 1.39). This peak corresponds to the maximum resistance encountered during insertion and confirms the clip's ability to hold cables securely while allowing compliant deformation.

Overall, the experimental results confirm that the proposed warehouse system provides a reliable, repeatable, and scalable solution for automated cable storage, effectively bridging the gap between cable production and robotic wiring operations.

Chapter 2

Design of a Climbing Robot for Continuous WAAM 3D Printing

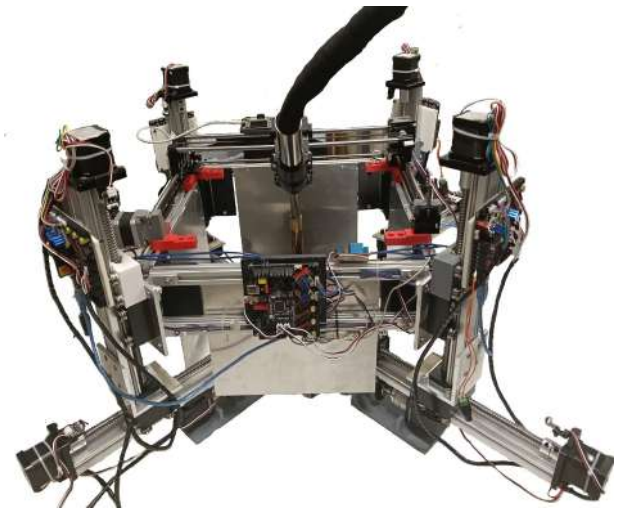


Figure 2.1. Climbing Mobile 3D Printer prototype operating on a WAAM-fabricated structure.

Wire Arc Additive Manufacturing (WAAM) is a Directed Energy Deposition (DED) process that employs an electric arc to melt a continuous metallic wire feedstock, depositing molten material layer by layer to form near-net-shape components [87, 42]. Compared to powder-based AM techniques, WAAM offers significantly higher deposition rates, often reaching several kilograms per hour, making it particularly suitable for large-scale metal fabrication. Its scalability, material efficiency, and capability to produce or repair large structural parts have led to applications across aerospace, shipbuilding, energy, and construction industries [37].

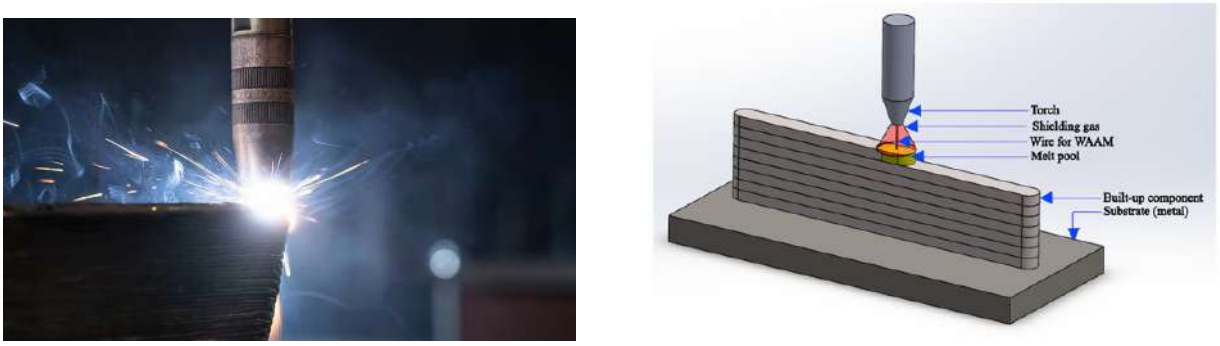
Despite its maturity as a manufacturing technology, integrating WAAM within mobile and autonomous robotic systems remains challenging. The process inherently introduces geometric irregularities, such as uneven layer thickness and surface roughness, which compromise the mechanical interface between the printed structure and the robot that must navigate or anchor onto it. These uncertainties limit the feasibility of continuous climbing and on-site manufacturing systems, particularly when precise positioning and stable clamping are required. Previous mobile 3D printers, often based on gear–rack mechanisms or smooth adhesion surfaces, struggle to maintain accuracy and reliability when interacting with WAAM-produced geometries, where small deviations can cause misalignment or slippage.

Addressing these limitations demands robotic solutions capable of adapting in real time to the geometric variability of WAAM structures, ensuring secure anchoring and consistent motion along irregular surfaces. Conventional climbing strategies, such as wheeled traction systems or gear–rack mechanisms [45], [44], [73], including our previous prototype based on

toothed wheels, are not suitable for WAAM-produced components. The inherent surface irregularities and layer discontinuities prevent reliable gear engagement, leading to slippage, loss of alignment, and ultimately unstable locomotion along the printed element. To overcome these drawbacks, the mechanism developed in this work adopts a fundamentally different approach: the printed structure itself is designed with integrated anchoring holes that act as discrete and repeatable gripping points. The robot employs a clamp-based anchoring mechanism capable of inserting and locking into these holes, allowing secure attachment and controlled climbing even in the presence of local geometric imperfections. This strategy decouples locomotion from the surface topology, ensuring stable mobility over irregular WAAM geometries and enabling truly autonomous Mobile Additive Manufacturing.

This work tackles this open problem by introducing a novel Climbing Mobile 3D Printer (CM3DP) designed to operate directly on WAAM-fabricated structures, integrating motor current perception and online optimization to achieve robust mobility and process continuity in uncertain environments.

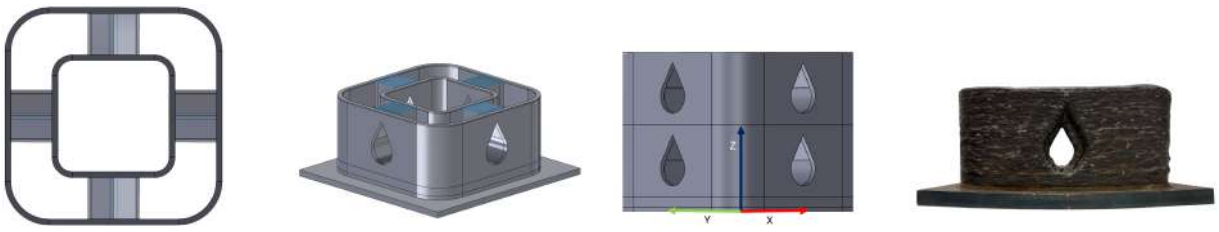
This work is under submission in IEEE Transactions on Mechatronics.



(a) WAAM process **(b)** Deposited layers

Figure 2.2. Overview of Wire Arc Additive Manufacturing : process, deposited layers, and torch operation.

2.1 WAAM-produced self supporting structure



(a) Cross-section. **(b)** 3D Iso view. **(c)** Two-level anchor design. **(d)** Real printed element.

Figure 2.3. WAAM-produced basement for the robot prototype, showing cross-section, 3D view, frontal anchor design, and the real printed element.

To validate the proposed climbing mechanism, a self-supporting structure was designed and fabricated using WAAM technology. The structure incorporates a series of evenly spaced anchoring holes that serve as secure attachment points for the climbing robot. The design accounted for the geometric irregularities typically observed in WAAM processes, ensuring reliable anchoring and traversal performance.

Table 2.1. Anchor Point height statistics.

Parameter	1st	2nd
Nominal height [mm]	35.0	35.0
Maximum height [mm]	36.0	38.0
Minimum height [mm]	33.0	24.5
Average height [mm]	34.5	29.7
Std. deviation [mm]	1.2	4.8

Table 2.2. Anchor Point diameter statistics.

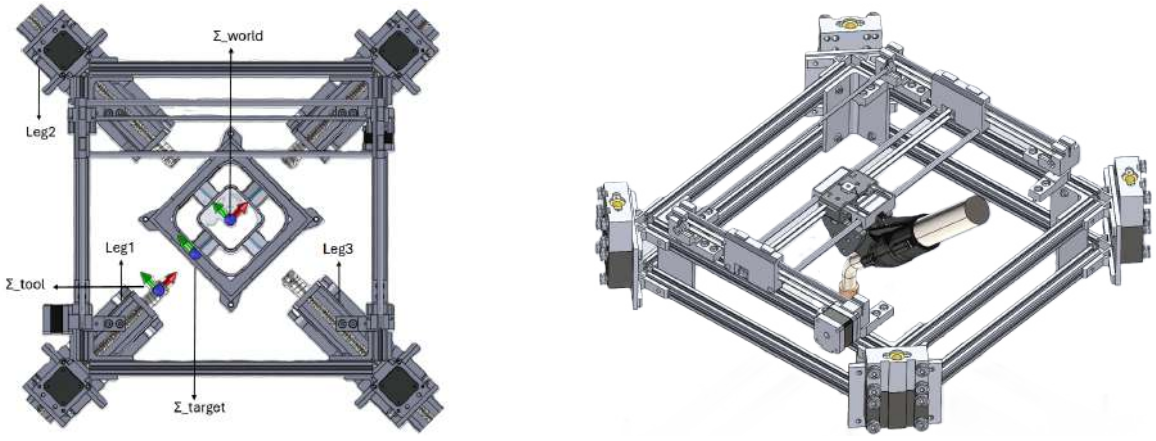
Parameter	1st	2nd
Nominal diam. [mm]	21.0	21.0
Maximum diam. [mm]	21.5	22.0
Minimum diam. [mm]	20.0	20.0
Average diam. [mm]	21.2	21.7
Std. deviation [mm]	0.7	0.8

The proposed CM3DP printer system enables the fabrication of self-supporting, longitudinal WAAM structures without external supports, such as those shown in Fig. 2.3. These elements integrate discrete anchoring holes that allow the robot to grip and climb along the printed element. Each anchor features a drop-shaped geometry composed of a circular base and a tapered upper region inclined at 60°, which facilitates gradual bridging of molten metal, minimizes sagging, and preserves structural continuity in unsupported areas. The anchor points measure approximately 35 mm in height and 21 mm in diameter, dimensions optimized for manufacturability and compatibility with the robot's climbing system described in the following sections.

A WAAM basement prototype was fabricated to validate the robot's mobility along the printed structure (Fig. 2.3). It consists of inner and outer walls forming a rounded-square cross-section (72 mm and 120 mm edges, respectively), with a wall thickness of 3.2 mm and anchoring holes 28 mm deep (Fig. 2.3a).

Despite minor irregularities typical of WAAM processes, the anchor geometry exhibits good repeatability. As summarized in Tabs. 2.2 and 2.1, the diameter variation is limited, while the height variation mainly affects the upper tapered region without compromising plug insertion. Overall, the adopted design ensures both manufacturability and functional robustness of the anchoring system.

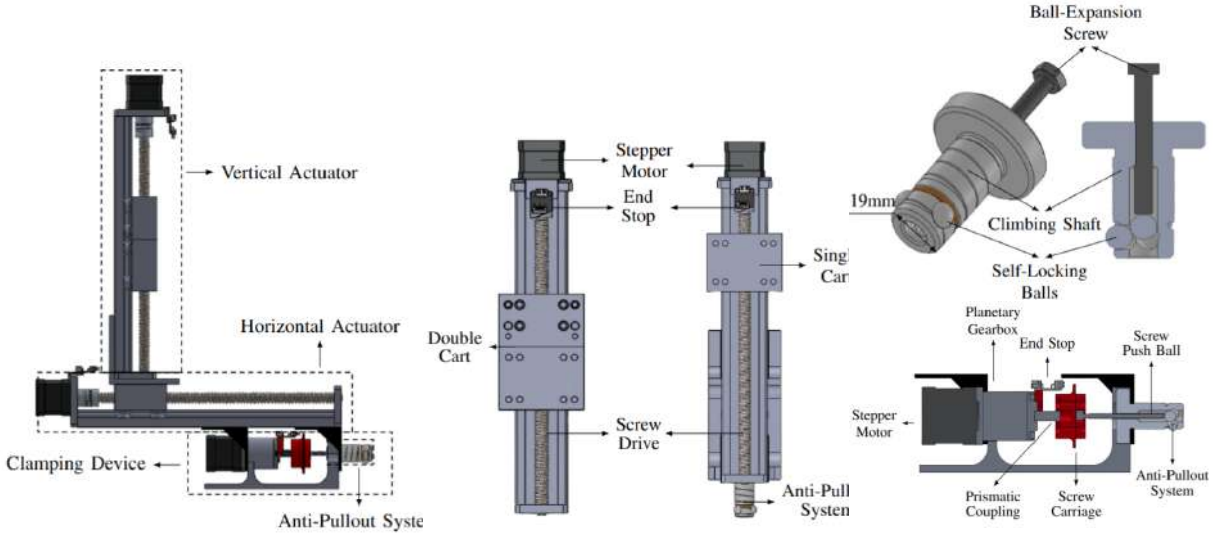
2.2 Robot Mechanical Design



(a) Coordinate frames of the climbing system showing tool, target, and world references.

(b) CAD model of the WAAM printer structure with torch and extrusion axes.

Figure 2.4. Architecture of the WAAM climbing printer: (a) reference frames definition and (b) mechanical frame assembly.



(a) Vertical and horizontal actuation unit.

(b) Climbing leg with clamping and anti-pullout systems.

(c) Locking system in (top) closed and (bottom) open states.

Figure 2.5. Overview of the climbing module components: (a) vertical–horizontal actuation unit; (b) climbing leg with clamping and anti-pullout systems; (c) locking mechanism in closed and open configurations.

2.3 CM3DP Mechanical Design

The concept behind this mechanism is to insert one leg into the hole of the WAAM-printed structure. The goal is to align the frame of the plug tip, denoted as Σ_{tool}, with the target frame Σ_{target} located inside the hole, as illustrated in Fig. 2.4a. To achieve this, the tip coordinates are controlled through the actuation of the X and Z legs, while the remaining two legs provide motion along the Y direction.

2.3.1 Robot Frame

The Robot Frame, shown in Fig. 2.4b, forms the main structural unit of the robot, that holds the WAAM torch and the X–Y motion system for deposition. It consists of two rigid rectangular modules connected by coupling plates that are built to be connected with the vertical actuators of the legs. In this way the 4 legs form the base of the printer able to move the frame in the Z direction. Mean while, the X-Y stepper axis to ensure the WAAM torch's precise movement during the printing process, are mounted on the top module. The frame, with overall dimensions of 390 mm (Y-axis) by 270 mm (X-axis), defines the workspace available for deposition. Two stepper motors, one per axis, drive the X–Y motion through toothed-belt transmissions, allowing controlled movement of the torch across the printing area. Fig. 2.4a illustrates the coordinate frames Σ_{world} used for kinematics and control.

2.3.2 Motion Legs

The Motion Legs subsystem provides motion along the Z-axis and enables the robot to climb along the printed WAAM-produced element. Each leg (Fig. 2.5) integrates: a Vertical Actuator, a Horizontal Actuator, and an Anti-Pullout System. The actuators employ NEMA17 stepper motors coupled with ball-screw transmissions to ensure precision and load-bearing capability up to 200 N per axis, resulting in a total vertical load capacity of 800 N. Endstops define reference positions, ensuring repeatable motion and safe operation.

An additional secondary carriage reinforces the vertical actuator by increasing stiffness and reducing deformation under load. It includes brass inserts, bushings, and bearings to guarantee low-friction sliding and rigid coupling with the frame. This configuration ensures accurate leg positioning and improved stability during both printing and climbing phases.

Clamping Device and Anti-Pullout System

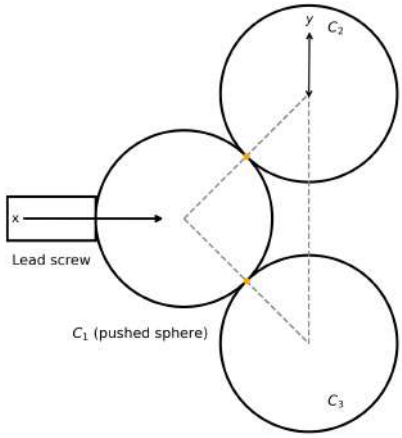


Figure 2.6. Geometry of the Anti-Pullout System showing the lead screw and the three locking spheres.

The Clamping Device, shown in Fig. 2.5c, couples a NEMA17 motor with a planetary gearbox (ratio 1:5.3) and a prismatic joint that transmits torque to a threaded shaft inside the Anti-Pullout System. The mechanism converts rotational motion into screw linear motion to actuate the locking balls in such a way: when the screw advances it pushes the ball mechanism that expand radially to press against the inner surface of the WAAM-produced anchor holes, generating a clamping force up to 3.3 kN. Reversing the motor retracts the screw, releasing the grip and restoring the balls to their seats. An end stop defines the home position, ensuring consistent engagement cycles.

As depicted in Fig. 2.4a, the Anti-Pullout tip is denoted as the control frame Σ_{tool} that must be moved to the desired insertion point Σ_{target} through coordinated legs motion. The vertical actuator controls the insertion depth along the Z-axis, the horizontal actuator handles lateral X-axis adjustments, and the Y-axis translation is achieved by the synchronized motion of Legs 2 and 3. This coordination enables accurate plug insertion and secure anchoring during climbing operations.

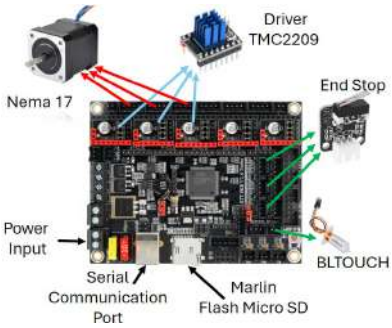
Consider the configuration shown in Fig. 2.6, where the lead screw pushes the left sphere horizontally, resulting in a vertical displacement of the upper one. Let the three spheres have radius R and be initially arranged with center-to-center distance $2R$ and angular spacing θ (e.g., $\theta = 45^\circ$ in the current design).

Kinematic constraint. Before motion, the centers of the contacting spheres are

$$C_1 = (0, 0), \qquad C_2 = (2R\cos\theta,\ 2R\sin\theta).$$

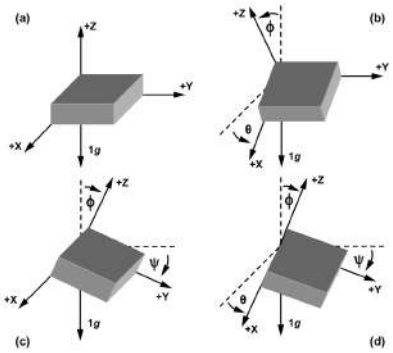
After the lead screw pushes sphere C_1 by a horizontal displacement x, the new positions become

$$C_1' = (x, 0), \qquad C_2' = (2R\cos\theta,\ 2R\sin\theta + y),$$



(a) BigTreeTech SKR v3.0 motherboards used for the CM3DP subsystems.

(b) STEVAL-MKI210V2K IMU kit and STEVAL-MKI109V3 interface board.



(c) Reference frame and IMU tilt angles (ϕ, θ, ψ).

Figure 2.7. Control and sensing hardware of the CM3DP system. (a) SKR∼v3.0 control boards for distributed actuation; (b) STEVAL IMU kit and interface board for attitude sensing; (c) IMU reference frames and tilt angles for self-balancing control.

where y is the corresponding vertical displacement of the upper sphere. Since the spheres remain in tangential contact, their center-to-center distance is preserved:

$$(2R\cos\theta - x)^2 + (2R\sin\theta + y)^2 = (2R)^2.$$

Solving for y yields:

$$y(x) = \sqrt{4R^2 - (2R\cos\theta - x)^2} - 2R\sin\theta. \tag{2.1}$$

2.4 Control Architecture

2.5 CM3DP Control Architecture

The control architecture of the CM3DP system is divided into one *Frame subgroup* and four *Climbing subgroups*, each composed of a motion leg and its associated clamping device (see Fig. 2.5). Each subgroup is managed by a BigTreeTech SKR v3.0 motherboard equipped with a 32-bit ARM Cortex-M7 STM32H743VIT6 processor (480 MHz). The firmware is based on the open-source *Marlin* platform [50], which interprets G-code commands for multi-axis coordination and stepper motor control. Boards communicate via USB with a central PC for command streaming and monitoring.

2.5.1 Leg Subgroup Control

Each device is driven by NEMA17 stepper motors with 1.8° resolution (200 steps/rev) and TMC2209 drivers operating at 1/16 microstepping. The number of microsteps (N_{LEG}) per millimeter (mm) required for precise motion is

Table 2.3. Custom G-code commands for CM3DP.

Command	Description
G1 X⟨pos⟩ F⟨vel⟩	Linear motion along X (horizontal actuator).
G1 Z⟨pos⟩ F⟨vel⟩	Linear motion along Z (vertical actuator).
G28 X Y Z	Homing all or selected axes.
G90 / G91	Absolute or incremental positioning.
M17 X Y Z	Enable stepper torque hold.
M410	Stop all steppers instantly.
L1 X Y Z	Start clamping procedure for Leg 1.

$$N_{\text{LEG}} = \frac{N_{\text{steps}} N_{\text{microsteps}}}{P_{\text{screw}}} \tag{2.2}$$

where $N_{\text{steps}} = 200$, $N_{\text{microsteps}} = 16$, and P_{screw} is the lead-screw pitch.

The microstepping resolution for the clamping motor, considering a gearbox ratio n, and the step of the lead screw, is given by

$$N_{\text{CLAMP}} = \frac{N_{\text{steps}} N_{\text{microsteps}}}{P_{\text{screw}} n} \tag{2.3}$$

which allows accurate control of the Anti-Pullout mechanism via custom G-code commands.

Finally, the number of microsteps per millimeter for the X-Y frame motors is

$$N_{\text{FRAME}} = \frac{N_{\text{steps}} N_{\text{microsteps}}}{2\pi R_{\text{pulley}}} \tag{2.4}$$

where R_{pulley} is the pulley radius driving the toothed-belt transmission.

2.5.2 Frame Control and IMU Compensation

To maintain the frame's horizontal alignment during anchoring and printing, a STEVAL-MKI210V2K IMU measures gravitational components (A_X, A_Y, A_Z) along the three axes. From these, the frame orientation is computed as:

$$\theta = \arctan\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right), \quad \psi = \arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right), \quad \phi = \arctan\left(\frac{\sqrt{A_X^2 + A_Y^2}}{A_Z}\right) \tag{2.5}$$

Considering the frame as a parallelepiped with sides l_X and l_Y, the corrective displacements along the Z-axis required to restore planarity are:

$$\Delta_{XZ} = \pm\frac{l_Y}{2}\sin(\psi), \qquad \Delta_{YZ} = \pm\frac{l_X}{2}\sin(\theta) \tag{2.6}$$

The sign depends on each leg's position relative to the measured tilt direction.

This integrated control strategy ensures precise motion synchronization, autonomous leveling, and stable operation of the CM3DP system during both printing and climbing.

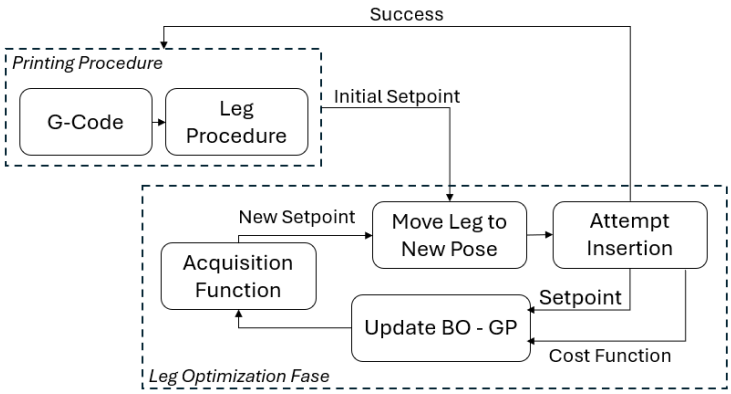


Figure 2.8. Bayesian Optimization-based clamping procedure for anchor insertion.

2.6 Bayesian Optimization for Anchor Insertion

During the climbing and anchoring phase, inserting the Clamping Device into the Anchor Points is challenging due to WAAM-induced irregularities such as burrs and geometric deviations (see Tabs. 2.1–2.2). These imperfections generate uncertainty in the true anchor position, making insertion prone to misalignment and stepper slip. To address this, a Bayesian Optimization (BO) framework (Fig. 2.8) is employed to refine the insertion trajectory online, compensating for both geometric and actuation uncertainties.

BO is well-suited to this task, as it efficiently optimizes black-box objectives with few variables and costly evaluations [25]. A Gaussian Process (GP) models the objective function over the search space, while the Expected Improvement (EI) Acquisition Function selects new sampling points by balancing exploration and exploitation. This iterative process refines the estimated anchor position until a successful insertion is achieved, after which the learned model is reused across subsequent layers to accelerate convergence.

The optimization cost is defined as:

$$J_{\text{BO}} = -\left(G_x e_x + G_{yz} e_{yz} + G_f e_f\right) \tag{2.7}$$

where $e_x = x_d - x$ is the axial insertion error, $e_{yz} = \sqrt{(y_d - y)^2 + (z_d - z)^2}$ is the radial misalignment, and $e_f = f - f_d$ (if $f > f_d$, else $e_f = 0$) penalizes excessive force, with G_x, G_{yz}, and G_f as weighting coefficients. If f exceeds a critical limit, the attempt is discarded and penalized.

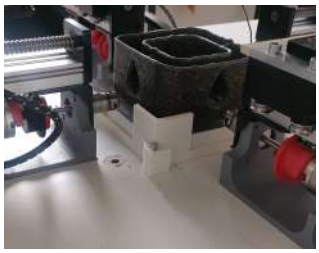
A simple algorithm outlines the procedure: after each WAAM layer, the leg moves to the nominal anchor pose $\Sigma_{\text{target}} = (x_d, y_d, z_d)$; the insertion proceeds while monitoring the motor current. When force exceeds the threshold, motion halts, the cost J_{BO} is updated, and a new candidate pose is proposed by the EI policy. The process repeats until a stable insertion is detected. The approach generalizes across all legs by applying appropriate coordinate transformations.

2.7 Results

This section presents the experimental validation of the CM3DP system, focusing on the climbing and anchoring performance on pre-printed WAAM elements. The current experiments validate the gripping precision and robustness of the climbing strategy on WAAM elements with teardrop-shaped holes, which inherently exhibit geometric deviations requiring active compensation.

Two representative WAAM-produced structures were used: a single-level element with one plane of anchoring holes and a two-level element with two sets of holes at different heights.

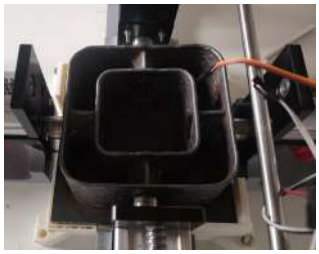
The latter was sufficient to validate the entire climbing and reanchoring cycle, as the procedure remains consistent for additional levels.



(a) Starting climbing position.

(b) Insert and lock left leg.

(c) Insert and lock right leg.



(d) First level completed.

(e) Leg released from first and inserted into second level.

(f) All legs anchored to the second level.

Figure 2.9. CM3DP climbing sequence: complete transition from the first to the second level. (a)–(c) show the anchoring of the left and right legs; (d)–(f) illustrate the reconfiguration and final positioning on the second level.

2.7.1 Climbing Procedure

The basic climbing sequence (Fig. 2.9) begins with the vertical actuator lifting and aligning the Anti-Pullout shaft with the next anchoring hole. The horizontal actuator then drives the shaft into the hole, and the Clamping Device locks the robot in place. This sequence is repeated for all four legs, enabling progressive climbing. However, due to WAAM-induced irregularities (Tabs. 2.1–2.2), holes may deviate in position or diameter, leading to either collision during insertion or insufficient locking torque. These issues can cause slight frame tilts, affecting subsequent insertions.

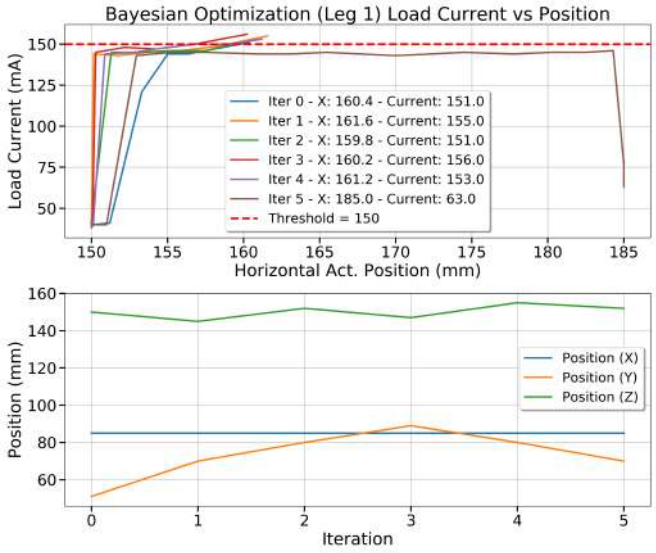
To ensure reliable engagement, a feedback-based control strategy integrated with Bayesian Optimization refines each insertion trajectory online. In parallel, a self-balancing routine (Sec. 2.5.2) maintains the frame horizontal alignment before resuming printing.

Figure 2.10 shows the BO control applied to the horizontal actuator of Leg1. A 150 mA threshold distinguishes normal insertions ($\sim$140 mA) from overloads. When exceeded, the cost function penalizes the configuration, steering the optimizer toward new poses. Successful insertions (brown curve) remain below the threshold, confirming proper alignment. The search space was limited to $\pm$10 mm in depth (Z) and $\pm$30 mm radially (Y), discretized at 0.5 mm to balance speed and robustness.

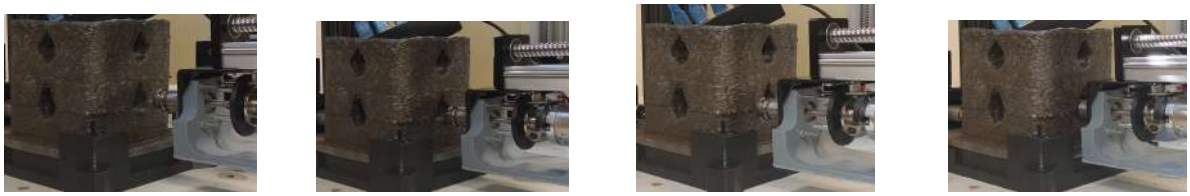
Each leg pair requires up to 25 exploration steps for the first insertion and up to 10 for the second, achieving average insertion times of 6 min and 2 min per level, respectively. Failures only occur when opposing anchors are misaligned, a rare event during tests.

2.7.2 Frame Balancing

After all legs are anchored, the robot automatically levels the frame before resuming printing in the parallel to the printing plane direction. Figure 2.11 shows a typical correction sequence:



(a) Current vs. position of the horizontal actuator (top), and position setpoint provided by Bayesian Optimization (bottom). The insertion trajectory is interrupted when the current exceeds the threshold.



(b) Initial position.

(c) First iteration failed due to wall contact.

(d) Second iteration failed due to wall contact.

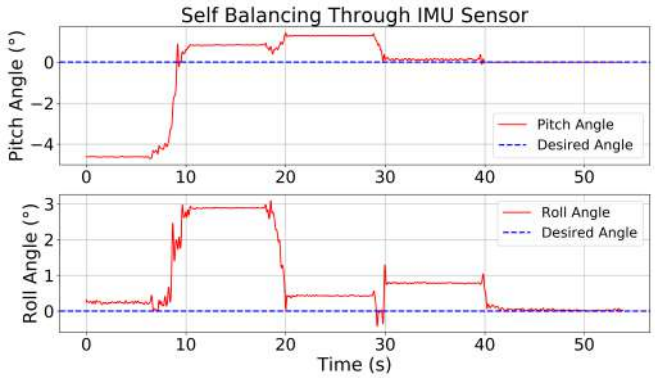
(e) Fifth iteration successful insertion.

Figure 2.10. Bayesian Optimization–guided insertion sequence. (a) shows the actuator current and position evolution; (b–e) illustrate the iterative learning process leading to a successful anchor engagement.

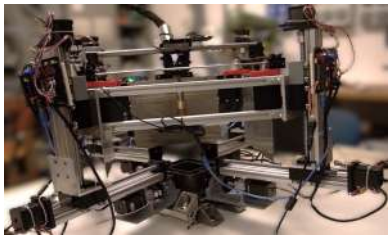
the IMU measures tilt angles, and leg displacements compensate for the misalignment. The red line represents measured angles, while the dashed blue line denotes the horizontal reference.

2.7.3 Torch Integration and Motion Testing

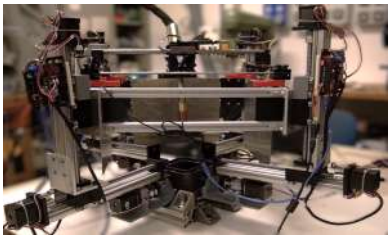
Fig. 2.12 reports preliminary trials aimed at identifying feasible parameters both at the power-source level and on the robot side. The welding current was varied from 75 A to 150 A. A representative setting at 130 A, with a printer TCP speed of 1000 mm/min and wire feed speed of 3.8 mm/min, provided a suitable quality thickness tradeoff. Under these conditions, the deposited bead exhibited approximately 3 mm width and a quasi uniform 1.5 mm height.
The selected parameters were then used to print a multi-layer square path representative of the baseline WAAM geometry, evaluating dimensional uniformity over three deposited layers. The G-code was generated from the CAD model by requesting a nominal toolpath speed of 1000 mm/min in the slicer. After the first layer, measurements along straight segments were consistent with the single-bead test (approximately 3 mm width and 1.4 mm height). However, at the rounded corners the deposition became non-uniform, reaching a maximum local height of 1.8 mm. As three additional layers were printed, corner over-deposition accumulated: the straight segments remained close to the expected value, while the corner region reached approximately 5.5 mm total height and not uniform. By manually adjusting the tool speed at the corner regions, the corner error was reduced and deposition quality augmented. These results validate the basic printing functionality of the integrated system, while indicating the need for dedicated G-code validation and corner-aware speed scheduling in future developments.



(a) Angular correction over time during the self-balancing process.

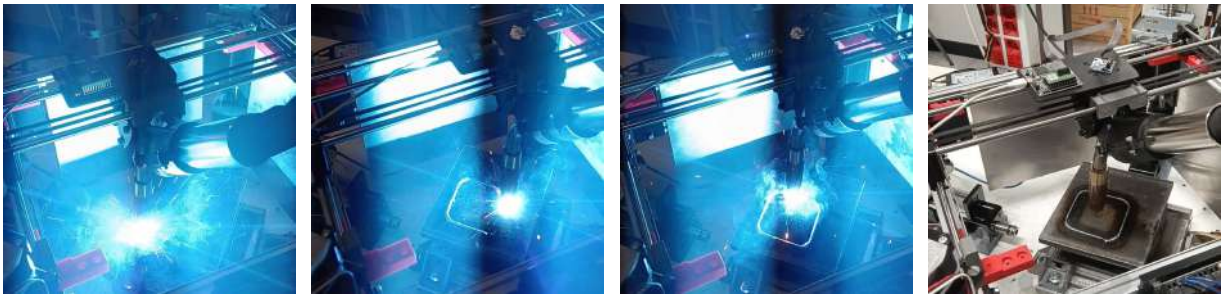


(b) Initial misalignment before compensation.



(c) Aligned configuration after correction.

Figure 2.11. Self-balancing procedure during the first anchoring phase. (a) Angular correction trend estimated by the IMU. (b–c) Visual comparison between initial misalignment and final aligned configuration.



(a) **(b)** **(c)** **(d)**

Figure 2.12. WAAM torch integration and motion validation: (a) modified frame design and (b) torch motion test along the element boundaries.

Chapter 3

From Computational Modeling to Real-World Transfer

Modern robotics increasingly relies on computational representations as the main interface through which reality is modeled, simulated, and predicted. Advances in computer graphics, physics-based simulation, and numerical modeling have enabled the design and validation of complex robotic behaviors entirely within the digital domain, often with remarkable precision and repeatability. Through meshes, Signed and Robotic Distance Functions (SDFs–RDFs), and dynamic models, robots can acquire a detailed geometric and physical understanding of their environment before any real interaction takes place.

Yet, despite the impressive realism achieved in simulation, the transition from computation to reality remains one of the central challenges in robotics. Digital models are necessarily idealized: they rely on perfect geometry, noiseless sensors, and stable dynamics. In contrast, the real world exhibits frictional asymmetries, mechanical backlash, material variability, and sensor uncertainty that are difficult to capture or predict. Even small discrepancies in these aspects can lead to significant deviations in shape reconstruction, dynamic response, or task execution.

This chapter explores the delicate relationship between computational modeling and physical realization.

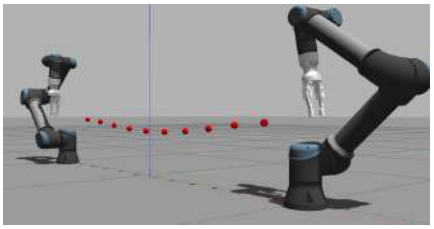
Building upon the real-world applications discussed in the previous chapters, Sec. 3.1 extends the analysis to the simulation of DLOs, focusing on cable dynamics and parameter sensitivity. The goal is to support reinforcement-learning (RL) policies for robust cable deposition tasks in electrical cabinets; part of this work was presented at ERF [34].

Subsequently, Sec. 3.2 examines the complementary problem of *sim-to-real transfer*, focusing on how discrepancies between simulated and real dynamics (the so-called *reality gap*) affect robotic control performance and policy deployment. The results of this work are planned for submission to AIM 2026. Sec. 3.3 then moves to the geometric domain, exploring how mesh-based representations can be transformed into continuous spatial fields such as Signed Distance Functions (SDFs) and Robotic Distance Functions (RDFs), which enable analytical computation and gradient-based analysis.

Finally, Sec. 3.4 introduces capability and reachability maps as higher-level tools to describe the feasible configurations and dexterous regions of robotic systems, linking geometric representations with task-level decision-making. From this work we built a library that is planned for submission to IEEE Transactions on Industrial Informatics (TII).

3.1 Simulation and Modeling of Deformable Linear Objects for Reinforcement Learning Scenario

In the context of unstructured robotic applications, the modeling and simulation of deformable linear objects (DLOs) such as cables and wires [43, 60] are essential for developing adaptive and learning-oriented control strategies. Our work presented at ERF, [34] focuses on the analysis



(a) Gazebo simulation environment for DLO manipulation.

(b) NVIDIA Isaac Sim environment for DLO manipulation.

Figure 3.1. Simulation environments used for evaluating DLO dynamics and manipulation strategies.

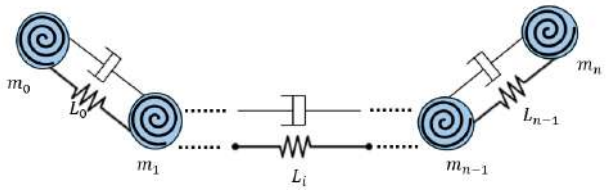


Figure 3.2. Mass-Spring-Damper (MSD) model for simulating DLO dynamics.

of a Mass–Spring–Damper (MSD) model to simulate DLO dynamics within robotic environments, comparing the performance achieved using NVIDIA Isaac Sim and Gazebo, extending the framework introduced in [59] to develop a reinforcement-learning environment for cable manipulation tasks. The study examines how variations in cable model parameters (specifically Young's modulus, damping, and discretization) affect the stability and realism of the simulated behavior. This preliminary investigation provides a systematic understanding of parameter sensitivity, establishing a basis for the future adoption of domain randomization (DR) techniques. The results align with the overall objective of this thesis: integrating computational modeling and control design to enhance robotic performance in unstructured scenarios, with particular attention to the relationship between physical fidelity, parameter optimization, and learning robustness.

3.1.1 Mass-Spring-Damper Model for DLO Simulation

As inspired by [59], the DLO can be modeled as a discrete sequence of masses connected by virtual spring–damper elements. Each mass interacts with its neighboring elements through elastic and damping forces, which collectively describe the cable's deformation under external loads.

As shown in Fig. 3.2, the DLO in simulation is represented as a chain of n masses, each linked to adjacent masses through virtual springs. The motion of the i-th mass is governed by Newton's second law:

$$m_i \frac{\partial^2 x_i}{\partial t^2} + k_d \frac{\partial x_i}{\partial t} = -\frac{\partial E_i}{\partial x_i} + F_i^e \tag{3.1}$$

where m_i denotes the mass of the i-th element, x_i its position, k_d the damping coefficient, E_i the total elastic energy stored in the springs connected to the mass, and F_i^e represents external forces such as gravity or contact interactions from the environment. The left-hand side of Eq. 3.1 expresses the dynamic equilibrium between inertial and damping effects, while the right-hand side represents the gradient of the potential energy and the contribution of external forces.

In this work, following the standard mass–spring–damper (MSD) formalism, two main elastic components are considered: *linear springs* and *bending springs*, representing stretching and bending effects, respectively. *Twisting springs* are not included in the model, as the considered

robotic task of cabling electrical cables takes into account just cable deposition and manipulation. This task does not involve torsional deformations. This simplification improves computational performance without compromising the physical fidelity of the simulated cable behavior.

Linear and Bending Springs in the Simulation

The energy stored in a linear spring (E_i^s), consequently forces, connecting two adjacent masses are defined as:

$$\begin{aligned} E_i^s &= \frac{1}{2} k_s (l_i - l_{0i})^2, \\ F_i^s &= -k_s (l_i - l_{0i}) \mathbf{u}, \end{aligned} \tag{3.2}$$

where l_i is the instantaneous distance between two mass points, l_{0i} is the rest length of the segment, and $\mathbf{u}$ is a unit vector in the direction of the spring's force. The stiffness k_s is determined by the material and geometric properties of the cable as:

$$k_s = \frac{EA}{l_{0i}}, \tag{3.3}$$

with E being Young's modulus and A the cross-sectional area of the wire. This formulation models axial elasticity, ensuring that the cable resists stretching in proportion to its material stiffness.

For bending behavior, the potential energy (E_i^b), consequently forces, are defined as:

$$\begin{aligned} E_i^b &= \frac{1}{2} k_b \beta_i^2, \\ F_i^b &= -\left(\frac{\partial E_{i-1}^b}{\partial x_i} + \frac{\partial E_i^b}{\partial x_i} + \frac{\partial E_{i+1}^b}{\partial x_i} \right), \end{aligned} \tag{3.4}$$

where β_i is the local bending angle between consecutive segments, computed as:

$$\beta_i = \arctan \left(\frac{|\mathbf{l}_{i+1} \times \mathbf{l}_i|}{\mathbf{l}_{i+1} \cdot \mathbf{l}_i} \right), \tag{3.5}$$

with $\mathbf{l}_i$ representing the segment vector between two consecutive masses, and $\mathbf{l}_{i-1}$ and $\mathbf{l}_{i+1}$ the adjacent segments. The bending stiffness k_b is calculated as:

$$k_b = \frac{EI}{l_{0i}}, \tag{3.6}$$

where I is the second moment of inertia of the cable's cross-section. This formulation enables the simulation of smooth curvature and realistic bending resistance. At each simulation step, the forces computed from both linear and bending contributions are integrated to update the velocities and positions of all masses.

3.1.2 Domain Randomization for Learning-Based Cable Manipulation

As highlighted by the previous results, the dynamics of a DLO are highly sensitive to the physical parameters used in simulation. Small variations in quantities such as mass, stiffness, or damping can significantly alter the system's stability and overall behavior. This sensitivity becomes even more critical when the goal is to generate training data for learning-based control algorithms, where the model's response directly influences the quality and consistency of the policy learned.

In real-world robotic manipulation, the physical properties of deformable cables are rarely known with precision. Factors such as material heterogeneity, manufacturing tolerances, or

Table 3.1. Parameters used in the simulation experiments for DLO stability analysis.

No.	E [MPa]	Discretization (n masses)	Sampling period Δt [s]	Stability	Sec. to instability
1	12.6	$n = 10$	0.0000050	Stable	∞
2	526.0	$n = 10$	0.0000050	Stable	∞
3	1002.0	$n = 6$	0.0000050	Stable	∞
4	1002.6	$n = 10$	0.0000050	Unstable	0.4
5	1002.6	$n = 10$	0.0000001	Stable	∞

environmental conditions can cause variations in mass distribution, elasticity, and damping. To cope with these uncertainties, *Domain Randomization* (DR) provides a powerful framework that enables reinforcement learning (RL) agents to acquire policies robust to parameter variations. Instead of training the policy on a single nominal model, DR exposes the agent to a wide range of randomized physical conditions, encouraging it to learn invariant behaviors that generalize across different system configurations.

In the context of DLO manipulation, this approach can be realized by randomizing the key parameters that define the cable's dynamics:

- the discretization level n (number of masses), which affects inertia distribution and the resulting oscillatory properties of the system;
- the Young's modulus E, governing both linear and bending stiffness as defined in Eqs. 3.3 and 3.6;
- the cross-sectional area A, which influences the stiffness and therefore the stored elastic energy in each segment;
- and, optionally, the damping coefficient k_d, which determines the system's energy dissipation and impacts numerical stability.

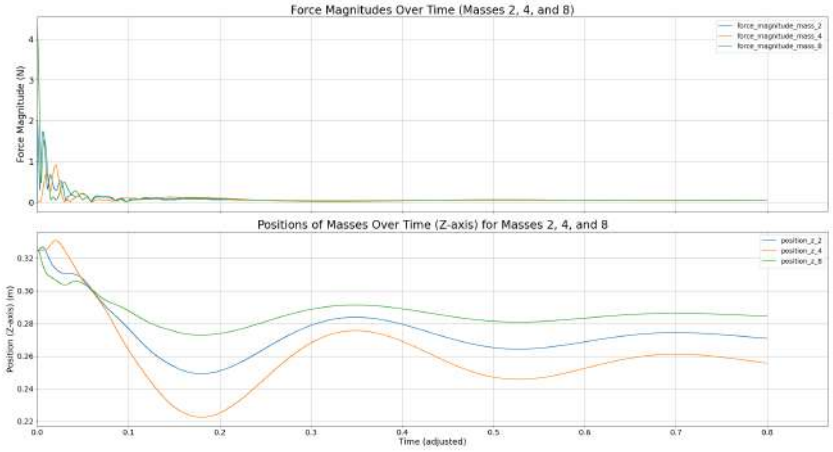
As shown in the previous section, variations in physical parameters significantly affect the DLO's energy and deformation behavior. This observation underpins the use of DR, where parameter randomization introduces controlled variability in the cable's dynamics, enriching the simulation data used for learning.

In this study, *we focus on randomizing the Young's modulus and the discretization level*, as they have the strongest impact on stability and global behavior. Extending this approach to other parameters would similarly diversify the simulated dynamics, ranging from soft to stiff cables.

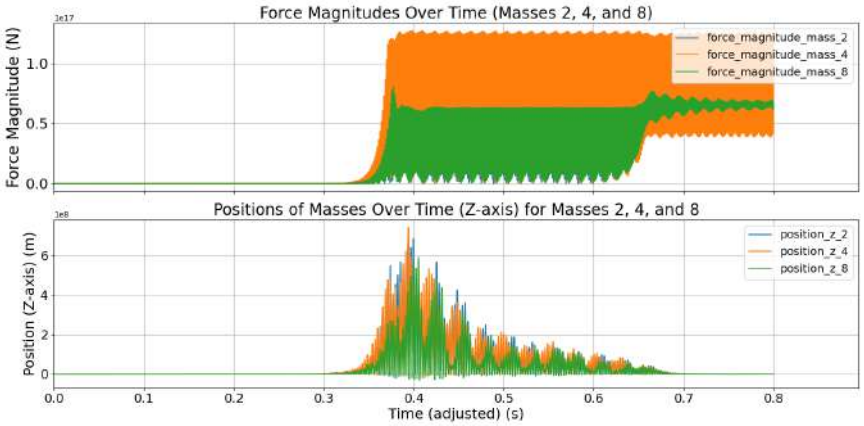
Ultimately, this framework aims to be integrated into a RL pipeline, where training under varying DLO properties allows the policy to generalize to real-world conditions, effectively reducing the *sim-to-real gap* and enabling robust manipulation of deformable objects.

3.1.3 Results

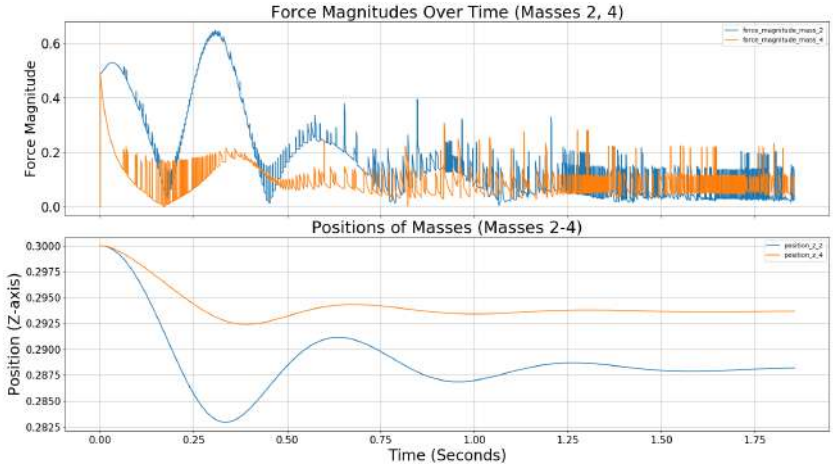
The results summarized can be compared in Gazebo and Isaac Sim as well. Table 3.1 and Fig. 3.3 clearly show that uncontrolled parameter randomization can lead to qualitatively different dynamic behaviors. As depicted in Fig. 3.3a, certain combinations of Young's modulus, mass discretization, and sampling time result in stable and physically consistent simulations. Conversely, excessive stiffness or inappropriate time discretization can lead to divergence and instability, as shown in Fig. 3.3b. Between these two extremes, marginally stable configurations may occur (Fig. 3.3c), where small oscillations persist over time, resembling a limit-cycle behavior caused by imperfect numerical damping.



(a) Stable cable simulation in NVIDIA Isaac Sim.



(b) Unstable cable simulation in NVIDIA Isaac Sim.



(c) Marginally stable cable simulation in NVIDIA Isaac Sim.

Figure 3.3. Examples of stable, unstable, and marginally stable cable simulations in NVIDIA Isaac Sim.

Using the PPO reinforcement learning algorithm, and after appropriate empirical tuning of its parameters, the agent successfully trains but exhibits strong oscillations during inference, indicating that it has poorly encoded the cable dynamics. These outcomes emphasize that, while parameter randomization is a powerful concept for improving learning robustness through DR, its application to deformable object modeling requires strict control of numerical stability and energy conservation. Randomly varying parameters such as mass or stiffness without adjusting the damping coefficient (k_d) or the integration time step (Δt) may produce simulations that remain apparently stable but exhibit unrealistic force oscillations and non-physical energy accumulation. When such corrupted data are used for RL, the agent may inadvertently learn inaccurate or unstable manipulation strategies.

Therefore, we conclude that the current DLO simulation framework, although suitable for static and well-tuned parameter studies, is not yet fully prepared for large-scale DR integration. Before introducing DR within RL pipelines, it is necessary to implement stabilization mechanisms that preserve total energy consistency and dynamically adapt the integration step to prevent divergence. Future work will focus on developing these stabilization strategies and evaluating more advanced simulation platforms or solvers to ensure reliable data generation for learning-based control.

3.2 From Simulation to Reality: Bridging the Gap in Robotic Control Learning

Simulation has become a cornerstone of modern robotics, providing a safe and computationally efficient medium for developing and validating control and learning algorithms[105, 81]. High-fidelity physics engines such as MuJoCo, Gazebo, and Isaac Sim allow robots to be tested under controlled yet realistic conditions, enabling massive data generation and the training of complex models without physical wear or risk. In particular, simulators have proven essential for reinforcement learning (RL)[81, 83], where thousands of rollouts can be performed in parallel to optimize control policies that would be otherwise infeasible to train directly on real hardware.

Despite these advantages, a persistent challenge remains: the behavior of a robot in simulation rarely matches its physical counterpart perfectly. Even small inaccuracies in dynamic parameters, such as link inertias, joint friction, or actuator models, can lead to substantial discrepancies in velocity and torque profiles once the policy is deployed on the real robot [5, 2]. This phenomenon, often referred to as the dynamical gap, represents one of the main obstacles to reliable sim-to-real transfer.

To mitigate this mismatch, the first part of this section focuses on a Real2Sim2Real pipeline developed in MuJoCo for torque-controlled manipulators. The approach begins with the acquisition of real joint torque trajectories, which are used to adapt the simulation parameters, such as friction and inertia, via trajectory matching and genetic optimization [41, 83]. Once the parameters are aligned with the physical system, RL is performed within a domain-randomized version of the calibrated simulator [40, 88, 76]. This process allows the trained policy to generalize beyond the exact nominal model, achieving smoother and more robust transfer when deployed on the real robot.

The second part of this section extends the same concept to velocity-controlled systems through a digital twin framework implemented in Isaac Lab. Here, the simulator is continuously synchronized with the physical robot, updating its dynamic parameters online through real data streams. Unlike the previous approach—where identification occurs offline before training—this method treats system identification as an ongoing learning process. A RL agent is used to tune the damping and friction parameters of the simulator in real time, ensuring that the simulated joint velocities remain consistent with those observed on the actual robot.

Both the approaches are planned to be published in the next months.

3.2.1 A simple reinforcement learning overview of algorithms used in this work

The RL algorithms adopted in this work for Real2Sim transfer are grounded in the actor–critic framework, which combines the estimation of a value function with a parameterized policy optimized through gradient methods. These approaches are designed for continuous state and action spaces, as required in robotic control where torque or velocity commands vary smoothly. Their common goal is to approximate the solution of the Bellman optimality equation through iterative policy improvement while maintaining numerical stability and efficient exploration.

Formally, an RL problem can be described as a Markov Decision Process (MDP) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where the agent learns a policy $\pi(a|s)$ that maximizes the expected discounted return:

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{3.7}$$

In actor–critic methods, two neural networks are jointly trained: the *critic*, which estimates the state–action value function $Q_\phi(s,a)$, and the *actor*, which parameterizes the policy $\pi_\theta(a|s)$. The critic minimizes the Bellman residual:

$$L_Q(\phi) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})}\Big[\big(Q_\phi(s_t,a_t) - y_t\big)^2\Big], \tag{3.8}$$

with the target $y_t = r_t + \gamma Q_{\bar{\phi}}(s_{t+1}, \pi_\theta(s_{t+1}))$, while the actor is updated via the deterministic policy gradient:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s_t\sim\mathcal{D}}\Big[\nabla_\theta \pi_\theta(s_t) \nabla_a Q_\phi(s_t,a)\Big|_{a=\pi_\theta(s_t)}\Big]. \tag{3.9}$$

DDPG and TD3. The Deep Deterministic Policy Gradient (DDPG) [56] algorithm represents the foundation of this family. It enables direct learning of deterministic continuous policies using experience replay and target networks to stabilize training. However, DDPG is sensitive to noise and prone to overestimation bias in the Q-function. The Twin Delayed Deep Deterministic Policy Gradient (TD3) [26] mitigates these issues through the use of twin critics,

$$y_t = r_t + \gamma \min_{i=1,2} Q_{\bar{\phi}_i}(s_{t+1}, \pi_\theta(s_{t+1})), \tag{3.10}$$

and delayed policy updates, improving convergence stability and robustness, especially for torque control.

SAC and TQC. The Soft Actor–Critic (SAC) [38] algorithm extends the classical formulation by introducing an entropy term into the objective, encouraging stochastic exploration:

$$J_{\text{SAC}}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \big(r_t + \alpha \mathcal{H}(\pi_\theta(\cdot|s_t))\big)\right], \tag{3.11}$$

where α controls the trade-off between reward maximization and entropy. This formulation yields policies that are both smooth and robust to modeling inaccuracies. The Truncated Quantile Critics (TQC) [48], algorithm builds upon SAC by representing $Q_\phi(s,a)$ as a quantile distribution and truncating the highest quantiles to suppress overestimation, resulting in improved stability in torque-based control.

PPO. Proximal Policy Optimization (PPO) [82] represents a complementary approach based on stochastic on-policy learning. It introduces a clipped surrogate objective to constrain the

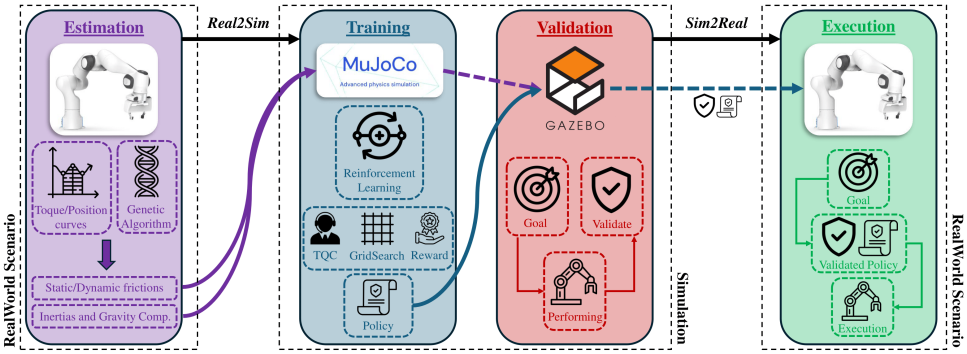


Figure 3.4. Diagram illustrating the workflow for validating and executing a reinforcement learning policy in both simulation and real-world scenarios. The process includes parameter estimation (e.g., torque/position curves, friction, inertias, and gravity compensation), training via RL (TQC in figure) policy validation, and execution for Real2Sim and Sim2Real applications

policy update within a trust region:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\big)\Big], \tag{3.12}$$

where $r_t(\theta)$ is the probability ratio between new and old policies, and A_t the advantage function. This mechanism ensures monotonic policy improvement and avoids destabilizing updates.

3.2.2 Real2Sim2Real for Torque-Controlled Manipulators

This section outlines the framework developed to achieve robust torque-level control in robotic manipulators through a *Real2Sim2Real* pipeline. The method extends the approach in [5] by integrating advanced identification techniques [29] to improve the dynamic fidelity of the simulated model.

Physical data are first used to calibrate the robot's dynamics (*Real2Sim*), enabling a realistic simulation environment for policy training with domain randomization. The trained RL policy is then validated in both MuJoCo and Gazebo before being deployed on the real Franka Emika Panda robot, ensuring generalization across simulation domains and robustness under real-world torque control *Sim2Real*. The goal of this is to train a RL-agent to drive the robot from a random initial joint configuration to a target pose using only torque commands.

Reward Function and Policy Update

Following the calibration and validation stages described above, the next step focuses on defining the control objective that guides the learning process. This objective is expressed through a reward function that quantifies the quality of the robot's behavior in simulation, forming the basis for policy optimization. To achieve a desired goal the robot-agent seeks to maximize the expected cumulative reward, defined as:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{3.13}$$

where R_t is the expected return at time t, r_{t+k} is the instantaneous reward at step $t+k$, and $\gamma \in [0, 1)$ is the discount factor weighting the contribution of future rewards.

The reward function r is designed to encourage accurate trajectory tracking while penalizing excessive joint velocities and accelerations that could induce instability or high energy

consumption. It is defined as:

$$r = -k_1 d - k_2 \sum_{i=1}^{7} \tanh\left(|\dot{q}_i|\right) - \frac{k_3 \sum_{i=1}^{7} \tanh\left(|\ddot{q}_i|\right)}{\left|k_4 - \min(1, d)\right|} - \frac{k_5 \sum_{i=1}^{7} \tanh\left(|\ddot{q}_i|\right)}{k_6 + d}. \tag{3.14}$$

where d represents a normalized distance-to-goal term (defined below), $\dot{q}_i$ and $\ddot{q}_i$ denote the joint velocities and accelerations, and $k_1, \ldots, k_6$ are weighting coefficients selected empirically to balance convergence speed and motion smoothness. The use of hyperbolic tangent functions ensures bounded penalization and smooth gradients during policy optimization.

Policy learning is performed using a continuous control RL algorithm (e.g., TQC, SAC, or TD3), which iteratively updates the policy parameters θ to maximize the expected return $J(\pi_\theta)$ (see Eq.3.9). The specific terms in the reward function are defined as follows:

- k_1: A positive value that affects how much the agent is penalized based on the distance between the end effector and the goal position.
- k_2: A positive constant that controls the contribution of the joint velocity $\dot{q}_i$ and joint acceleration $\ddot{q}_i$ terms in the reward function. It adjusts the penalty for high velocities and accelerations of the joints.
- k_3: A positive constant that determines the relative importance of the joint acceleration term $\ddot{q}_i$ in the reward function. It scales the penalty based on joint accelerations.
- k_4: A positive constant that replaces the value 1.15 in the original equation. It scales the penalty term for angular acceleration and modifies the impact of the term $|k_4 - \min(1, d)|$ in the reward.
- k_5: A positive constant that replaces the value 0.15 in the original equation. It influences the denominator of the second angular acceleration term, modifying how the distance affects the angular acceleration penalty.
- d: The normalized distance between the end effector and the goal position, defined as:

  $$d = \frac{\|\mathbf{x}_{\text{obs}}\|}{d_{\text{goal}}},$$

  where $\|\mathbf{x}_{\text{obs}}\|$ is the Euclidean distance from the current position of the end effector to the goal position, and d_{goal} is the distance to the goal.
- $\dot{q}_i$: The velocity of joint i. It represents how quickly the joint configuration is changing over time.
- $\ddot{q}_i$: The acceleration of joint i. It indicates how quickly the joint velocity is changing over time.
- The terms $\ddot{q}_i$ and $\dot{q}_i$ are used to penalize the agent for rapid movements and excessive accelerations, encouraging smoother motion. The tanh function is used to bound the values of these terms, ensuring the penalties are non-linear and prevent excessively large values from dominating the reward.

The reward function is designed to balance accuracy in reaching the goal position with smooth and energy-efficient joint movements. The first term encourages proximity to the goal, while the velocity and acceleration penalties (via $\dot{q}_i$ and $\ddot{q}_i$) discourage abrupt or unstable motion. The use of tanh ensures numerical stability and limits the influence of extreme values.

Through iterative training, the agent learns to associate specific joint torques with successful task completion, gradually improving its policy π to minimize errors and enhance task efficiency. The policy update can be expressed as:

$$\theta_{t+1} = \theta_t - \eta(t)\nabla_\theta L, \tag{3.15}$$

where θ_t represents the parameters of the policy at time t, $\eta(t)$ is the learning rate, $L(\cdot)$ is the loss function and $\nabla_\theta L$ denotes the gradient of the loss. See [95] for more details.

As claimed before, the performance of RL in robotic control strongly depends on the accuracy of the robot's dynamic model. In this work, we focus on estimating key dynamic parameters of the Franka Emika Panda, to enhance the fidelity of simulation and improve sim-to-real transfer.

The robot dynamics are described by the standard equation of motion for an n-DOF manipulator:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q), \tag{3.16}$$

where τ denotes the vector of joint torques, $M(q)$ the inertia matrix, $C(q,\dot{q})$ the Coriolis and centrifugal terms, and $G(q)$ the gravity vector. This is the equation we want to match between simulation and reality by identifying the parameters involved.

Preliminary experiments [5], shown in Figs.3.7a, 3.7c, revealed discrepancies between simulated and real behaviors, mainly due to unmodeled friction and inaccurate gravity compensation.

To address these issues, we identified a set of parameters p for each joint, including the friction coefficients and the six independent elements of the joint inertia matrix. The identification problem was formulated as an optimization task minimizing the trajectory tracking error between simulated and real joint trajectories:

$$E(p) = \sum_{t=0}^{T} \left\| y_{\text{sim}}(t,p) - y_{\text{real}}(t) \right\|, \tag{3.17}$$

where $y_{\text{sim}}(t,p)$ and $y_{\text{real}}(t)$ represent the simulated and measured torque trajectories, respectively.

During the experiments, we observed an asymmetry in kinetic friction: the torque required to move a joint differed depending on its direction, leading to trajectory mismatches between simulation and reality. Indeed in simulations as MuJoCo, Isaac Sim and Gazebo friction is typically modeled as two variable called 'friction' and 'damping', strongly symmetric that represent the static and dynamic friction components, respectively. For this reason we also noticed that modifying the friction parameters provided by mujoco simulation was not sufficient to reduce the reality gap. While static friction is computed by experimental results, computing the torque necessary to move a single joint, dynamic joint friction was modeled using a smooth sigmoid-based function already defined in [29]:

$$\tau_{f,j} = \frac{\varphi_{1,j}}{1 + e^{-\varphi_{2,j}(\dot{q}_j + \varphi_{3,j})}} - \frac{\varphi_{1,j}}{1 + e^{-\varphi_{2,j}\varphi_{3,j}}}, \quad j \in [1,\ldots,7], \tag{3.18}$$

where $\varphi_{1,j}$, $\varphi_{2,j}$, and $\varphi_{3,j}$ are friction-related parameters specific to joint j.

In order to avoid local minima of classical method, parameter identification was performed using a stochastic search through an iterative optimization process inspired by genetic algorithms [75]. At each iteration, a set of candidate parameter vectors was evaluated by simulating the corresponding trajectories and computing the tracking error using Eq. 3.17. The best-performing candidates were then used to generate new parameter sets via mutation and selection, progressively refining the estimate until convergence.

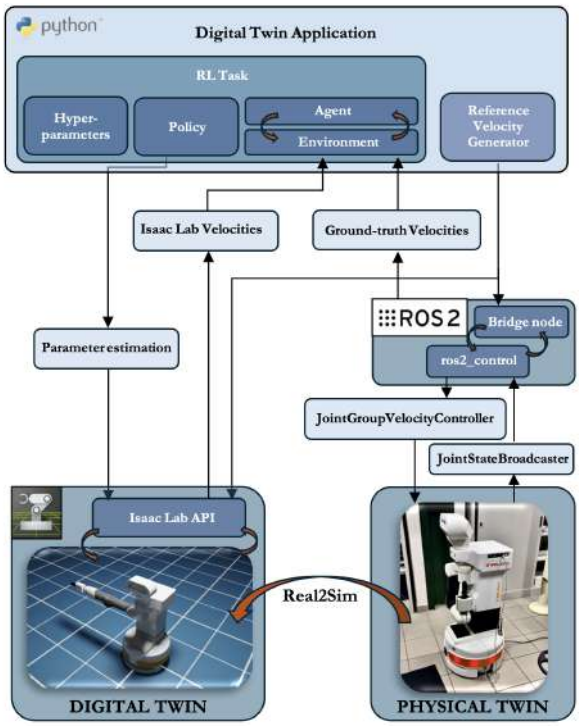


Figure 3.5. Sim2Real digital architecture.

3.2.3 Digital Twin for Velocity-Controlled Manipulators

Since RL models trained in simulation often underperform in real-world control tasks due to the well-known *reality gap*, the proposed approach repurposes such models for online parameter estimation rather than direct policy deployment. While in previous section the identification was performed offline before training, here the focus is on continuously adapting the simulation parameters in real time to match the physical robot's behavior. Using real-world data collected from the TIAGo robotic manipulator, the goal is to develop a high-fidelity Digital Twin within a novel, real data-driven framework. First we used a torque controller, now since we want to bring the Task Priority controller that will be presented in Chapter 4 in simulation, we focus on velocity control. The target in this section is to identify the simulation parameters that minimize the velocity error between the real robot and its digital twin, since a mismatch in velocity profiles can lead to significant performance degradation in simulation and redundant robot will not behave as expected. A reference velocity profile is taken into account. The core idea is: the Digital Twin continuously updates its dynamic parameters to reproduce the real robot's joint velocities in real time, ensuring synchronized behavior between the simulated and physical system.

As illustrated in Fig. 3.5, the experimental architecture includes a reference signal generator that periodically emits joint-velocity commands transmitted simultaneously to both the physical robot and its Digital Twin. The real system operates in an open-loop configuration, with no feedback influencing the command generation. Meanwhile, the real robot's state feedback is continuously streamed into the simulation environment, allowing the computation of the real-to-sim discrepancy. This discrepancy, expressed as the velocity difference between the two systems, is integrated into the loss function used for online training of simulation parameters. The RL agent thus minimizes the velocity error over time, dynamically adjusting the simulator's internal parameters to achieve a high-fidelity digital replica of the real robot.

The trajectory selected for these experiments is a high-frequency sinusoidal motion applied at the joint level. As shown in Fig. 3.6, the trajectory is accurately tracked by the Isaac Lab simulator, while the real-world robot exhibits notable discrepancies. To enhance the dynamic fidelity

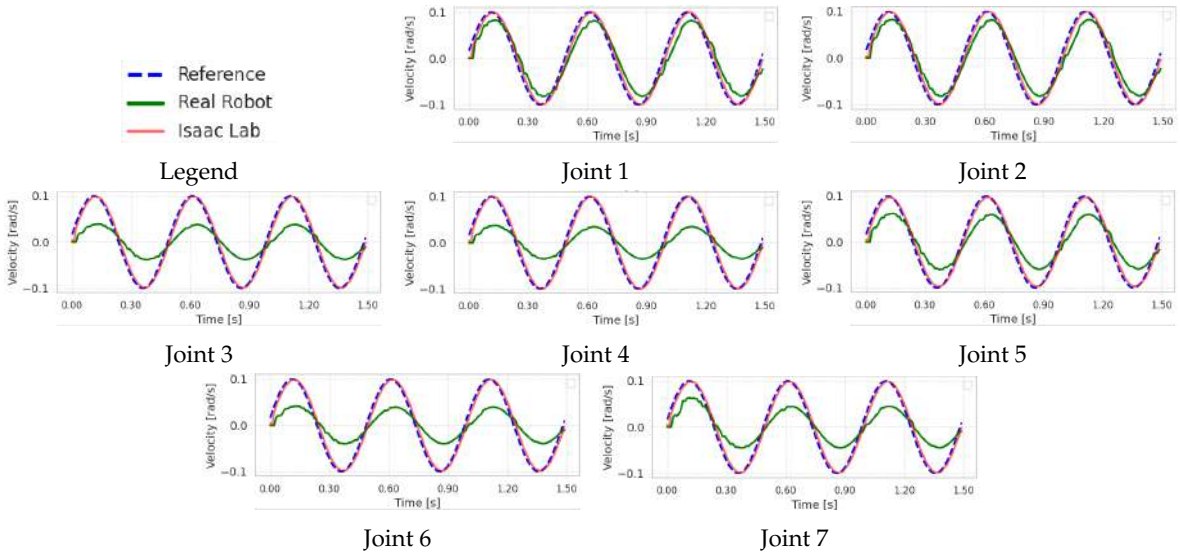


Figure 3.6. Comparison of joint velocities between Reference (blue dashed), Gazebo (green), and Isaac Lab (red) across all joints during the pre-training phase.

of the digital twin, the problem of system identification is formulated as a RL task focused on estimating the simulation parameters that most influence velocity control behavior. Thus, we racall that in this case the main objective is not to perform high-precision reference tracking, but to replicate the velocity response of the physical robot inside the simulated environment.

A preliminary analysis identified the parameters exerting the greatest influence on velocity-controlled dynamics. Among the various actuator models available in PhysX, the *implicit actuator model*, based on an internal spring-damper (PD-like) control structure, demonstrated the most predictable and tunable behavior. In particular, variations in the **damping** and **friction** coefficients produced systematic changes in the simulated velocity response across joints. Conversely, the stiffness parameter was excluded from optimization, as it mainly affects position regulation rather than velocity dynamics. In short, we aim to identify the function that best approximates the friction behavior within the simulator.

Due to the nonlinearities introduced by PhysX's implicit solver, analytical identification is not straightforward. Hence, the estimation process is reformulated as a black-box optimization problem solved using RL methods. Two algorithms, Soft Actor–Critic (SAC) and Proximal Policy Optimization (PPO) are employed. The RL agent iteratively adjusts the damping and friction parameters for each of the robot's seven joints to minimize the discrepancy between simulated and real velocity responses.

The top-performing trial for each algorithm is selected based on the highest achieved final episode reward. The corresponding hyperparameter sweep ranges adopted for SAC and PPO are summarized in Tab. 3.3.

Curriculum Learning for Multi-Joint Optimization

To address the sample inefficiency of RL in high-dimensional robotic systems, a **Curriculum Learning (CL)** strategy is introduced to accelerate training. CL structures the learning process through a sequence of progressively more complex tasks, allowing the agent to build competence incrementally. This principle is beneficial when training on complex robotic tasks is impractical from scratch.

In this context, the RL training process is divided into sequential stages:

1. *Source task*: the agent starts by optimizing damping and friction parameters for a single joint;

2. *Intermediate tasks*: additional joints are progressively activated, as defined by the variable *active_joints*;

3. *Target task*: all seven joints are optimized simultaneously.

This staged progression reduces the initial search-space dimensionality, mitigating the risk of convergence to poor local minima. Advancement through the curriculum is regulated by a **signal-correlation criterion**. Specifically, the *Pearson correlation coefficient* between the simulated and real joint velocity profiles is computed as:

$$C(\dot{q}_{\text{sim}}, \dot{q}_{\text{real}}) = \frac{\sum_{t=1}^{T} \dot{q}_{\text{sim},t} \dot{q}_{\text{real},t}}{\sqrt{\sum_{t=1}^{T} \dot{q}_{\text{sim},t}^2} \sqrt{\sum_{t=1}^{T} \dot{q}_{\text{real},t}^2}}, \tag{3.19}$$

where T is the number of signal samples. A new joint is introduced into the optimization process once the median correlation across all joints exceeds a predefined threshold ρ. The key parameters governing this progression are summarized in Tab. 3.2.

Population-Based Hyperparameter Optimization

RL performance is highly sensitive to hyperparameters such as learning rate, discount factor, and entropy coefficient. To automate their tuning, a **Population-Based Training (PBT)** approach is employed through the Ray Tune–Optuna framework. Ray Tune orchestrates distributed training, while Optuna applies the Tree-structured Parzen Estimator (TPE) for efficient search across hyperparameter spaces. Hydra manages configuration hierarchies, enabling structured sweeps via YAML files and command-line overrides, with real-time logging handled by TensorBoard.

SAC and PPO differ fundamentally in their optimization principles and hyperparameter sensitivities. The explored hyperparameter ranges are listed in Tab. 3.3, while the best configurations and corresponding rewards are reported in Tab. 3.5. There are three shared standard hyperparameters plus two parameters each algorithm typical of on-policy and off-policy algorithm respectively.

Reward Function and Policy Update

The designed reward function acts as a loss function, encouraging the agent to minimize velocity errors between simulation and reality. At each timestep, it penalizes the Mean Absolute Error (MAE) between the joint velocities of the Isaac Lab simulation and those recorded in the real or Gazebo-based robot:

$$R = -\frac{1}{active\_joints} \sum_{i=1}^{active\_joints} |\dot{q}_{i,real} - \dot{q}_{i,sim}| \tag{3.20}$$

This formulation guides the agent to iteratively refine joint dynamics, increasing fidelity across all degrees of freedom as training progresses.

Table 3.2. Correlation parameters for curriculum progression.

Parameter	Symbol	Value
Number of signal samples	T	200 steps
Correlation threshold	ρ	0.90

Table 3.3. Hyperparameter sweep ranges for SAC and PPO.

Hyperparameter	SAC	PPO	Meaning
`learning_rate`	$[10^{-5}, 10^{-2}]$	$[10^{-5}, 3\times10^{-4}]$	Policy update step size.
`batch_size`	[64,512]	[64,512]	Samples per gradient update.
`gamma`	$[0.9, 0.999]$	$[0.9, 0.999]$	Reward discount factor.
`ent_coef`	$[10^{-3}, 10^{-1}]$	$[0.0, 0.01]$	Entropy regularization weight.
`tau`	$[10^{-3}, 0.1]$	–	Target network soft update (SAC only).
`buffer_size`	{50k,100k,500k}	–	Replay buffer size (SAC only).
`n_steps`	–	$[256, 2048]$	Env. steps per policy update (PPO only).
`n_epochs`	–	$[3, 20]$	Batch passes per update (PPO only).

Action Space and Scaling The agent optimizes joint damping (ζ) and friction (μ) via a continuous action space defined as a *gym.Box* space with 28 elements per environment, four control parameters per joint for each of the seven arm DOFs:

$$\mathbf{a} = \begin{bmatrix} a_{\mu\_\text{scale}} & a_{\zeta\_\text{scale}} & a_{\mu} & a_{\zeta} \end{bmatrix} \in [-1, 1]^{m \times 4n}, \tag{3.21}$$

where $n = 7$ is the number of arm joints, and m is the number of parallel environments.

Given that relevant damping and friction values span several orders of magnitude (empirically observed in the range [1, 1000]), a logarithmic scaling strategy is adopted. Raw action values are first scaled to logarithmic exponents:

$$\mu_{\text{scale}} = \frac{a_{\mu\_\text{scale}} + 1}{2} \cdot 3, \quad \zeta_{\text{scale}} = \frac{a_{\zeta\_\text{scale}} + 1}{2} \cdot 3 \tag{3.22}$$

These values define the upper bounds of the estimated parameters:

$$\mu_{\max} = 10^{\mu_{\text{scale}}}, \quad \zeta_{\max} = 10^{\zeta_{\text{scale}}} \tag{3.23}$$

Final parameter values are obtained through linear scaling of the remaining actions:

$$\mu = \frac{a_{\mu} + 1}{2} \cdot \mu_{\max}, \quad \zeta = \frac{a_{\zeta} + 1}{2} \cdot \zeta_{\max} \tag{3.24}$$

This adaptive rescaling allows the agent to explore both low and high parameter ranges with appropriate resolution.

Action Masking To align with the curriculum learning approach, the agent controls only a subset of joints at each stage of training. The number of controlled joints is determined by the variable *active_joints*. Actions corresponding to inactive joints are masked out and excluded from both policy learning and simulation updates. This ensures progressive learning while maintaining consistent action dimensionality.

The masking is applied by selecting only the first *active_joints* elements of the friction and damping vectors.

Observation Space The observation space, implemented as a *gym.Box*, includes 49 elements per environment instance, seven features per joint. It's defined as:

$$\mathbf{o} = [\dot{q}_{ref}, \dot{q}_{real}, \dot{q}_{sim}, m] \in \mathbb{R}^{m \times 7n}, \tag{3.25}$$

where:

- $\dot{q}_{ref} \in \mathbb{R}^{m \times n}$: reference joint velocities,
- $\dot{q}_{real} \in \mathbb{R}^{m \times n}$: ground-truth joint velocities,

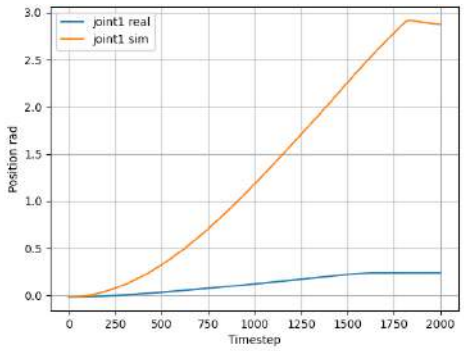
- $\dot{q}_{sim} \in \mathbb{R}^{m \times n}$: simulated joint velocities,
- $m \in \{0,1\}^{m \times n}$: active joint mask.

Inactive joints are zero-padded in the observation vector. The mask m ensures that policy updates are guided only by the currently active DOFs, preserving learning efficiency and stability as more joints are introduced.

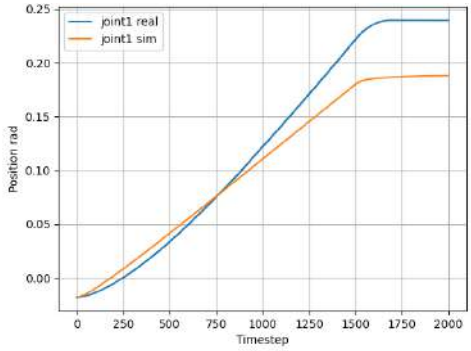
3.2.4 Results

Reinforcement Learning Results for Torque Control

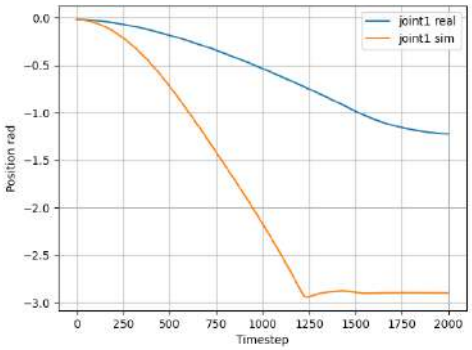
Dynamic Parameter Identification for Sim-to-Real Transfer



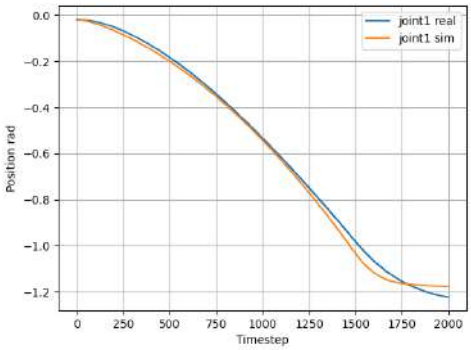
(a) Example of reality gap for first Franka joint before parameter estimation.



(b) Example of working simulation after parameter estimation.



(c) Example of reality gap for first Franka joint before parameter estimation.



(d) Example of working simulation after parameter estimation.

Figure 3.7. Comparison of torque profiles for the first joint of the Franka Emika Panda robot before and after dynamic parameter estimation. The left column shows significant discrepancies between simulated and real torques, while the right column demonstrates improved alignment post-estimation, highlighting the effectiveness of the calibration process.

We compared several RL algorithms (DDPG, SAC, TD3, and TQC) to determine the most effective approach for our setup. After tuning each algorithm via grid search, TQC consistently outperformed the others, achieving higher and more stable rewards, as shown in Fig. 3.8. Its robustness to parameter variations is particularly valuable for sim-to-real transfer, where physical uncertainties and modeling errors are unavoidable. Unlike SAC and TD3, which exhibited sensitivity to parameter changes, TQC maintained stable behavior across a broad range of conditions, making it well suited for real-world deployment.

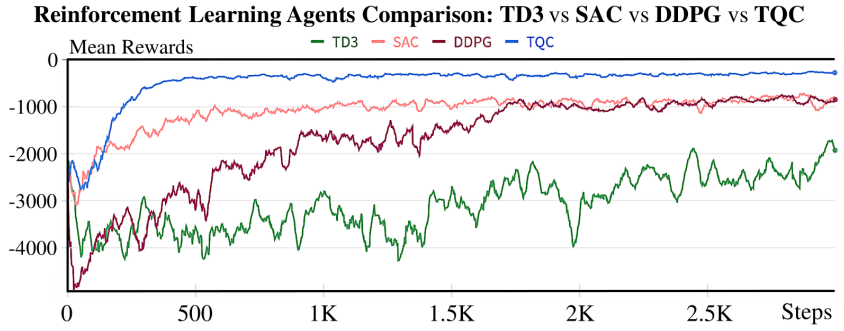


Figure 3.8. Episode mean reward comparison across DDPG, SAC, TD3, and TQC algorithms. TQC demonstrates consistently higher rewards, underscoring its superior performance and robustness in our environment.

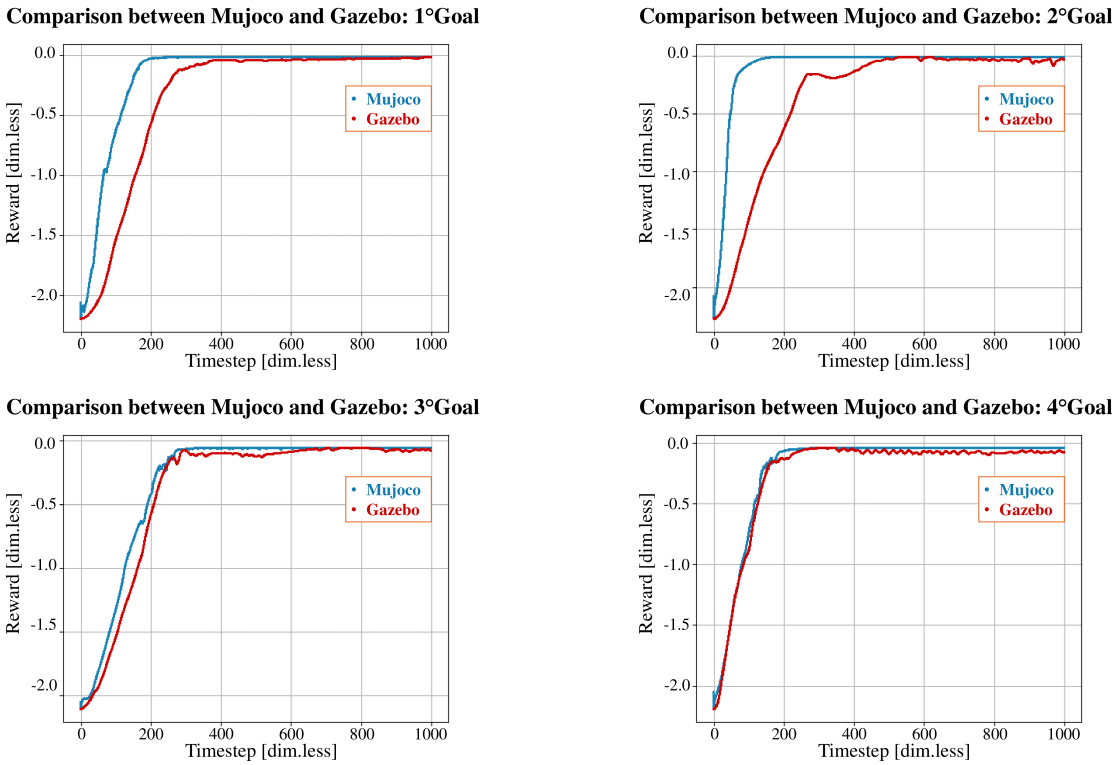


Figure 3.9. Comparison of the TQC agent's performance in the Mujoco and Gazebo simulators. The plots display the rewards obtained across four random goal scenarios, demonstrating consistent behavior and effective sim-to-sim transfer.

Parameter Estimation To correct gravity compensation errors, small torque offsets were manually introduced for each joint, ensuring symmetry between positive and negative motions. Static friction was characterized by gradually increasing torque until movement occurred, defining the minimum torque needed to overcome stiction in both directions. These thresholds were then used in simulation to replicate static friction behavior. Finally, the genetic optimization refined the friction and inertia parameters, significantly improving simulation fidelity. The updated friction coefficients are reported in Table 3.4, showing substantial adjustments from the initial estimates.

The TQC agent was trained in MuJoCo and later validated in Gazebo to progressively increase simulation realism. The agent learned to reach random goals within the robot workspace, ensuring adaptability to diverse target configurations. Domain randomization was applied by

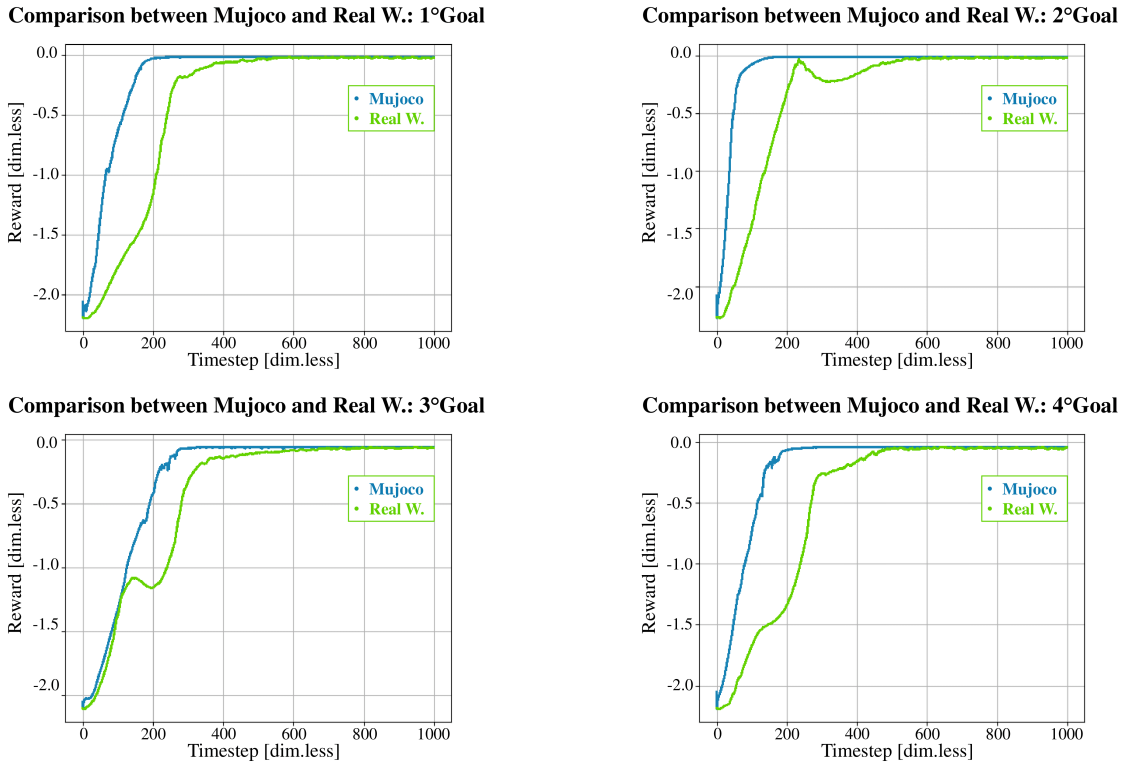


Figure 3.10. Comparison between Mujoco and Gazebo for four random goals. Mujoco shows smoother transitions while Gazebo introduces disturbances mimicking physical inconsistencies.

perturbing all friction parameters φ within $\pm 10\%$ and all inertia terms within $\pm 5\%$ of their identified values. This two-stage Sim2Sim phase allowed robust policy evaluation and fine-tuning under controlled yet diverse conditions before real-world deployment.

The final deployment on the real Franka Emika Panda validated the trained policy's. Thanks to domain randomization during training, the agent generalized effectively to physical conditions with minimal tuning. As shown in Fig. 3.10, real-world trajectories closely matched those obtained in simulation, with only minor deviations near target positions. Overall, the TQC agent maintained stable and reliable performance, confirming successful transfer from simulation to reality and demonstrating the feasibility of the proposed Sim2Real pipeline.

Reinforcement Learning Results for Velocity Control

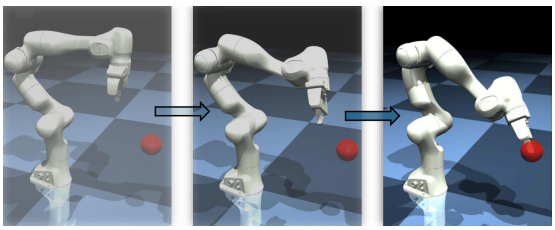
To further enhance the robustness of the learned policies and mitigate overfitting to a fixed velocity command, the reference velocity signals are randomized during training. Specifically, their amplitudes and frequencies are varied within predefined constrained ranges carefully selected to ensure that both joint position and velocity limits remain satisfied. This randomization strategy prevents the policy from specializing to a single motion pattern, thereby promoting better generalization and improving transferability to real-world scenarios, where the robot may encounter diverse motion profiles.

The complete experimental pipeline includes a systematic comparison between PPO and SAC algorithms, evaluated under both *concurrent* and *curriculum-based* training setups. Following this comparative phase, hyperparameter optimization and final policy deployment are performed to identify the most effective configuration for accurate real-world replication.

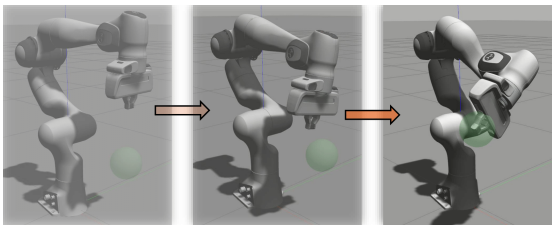
Both PPO and SAC agents undergo an extensive hyperparameter tuning stage, comprising 30 independent trials per algorithm. Although not exhaustive, this number of trials provides a balanced trade-off between parameter space exploration and available computational resources.

Table 3.4. Friction parameters $\varphi_{1,j}$, $\varphi_{2,j}$, and $\varphi_{3,j}$ for the seven joints of the Franka Emika Panda robot before and after dynamic optimization. The first block reports the analytically estimated (old) parameters, while the second lists the calibrated (new) values obtained through trajectory-based identification.

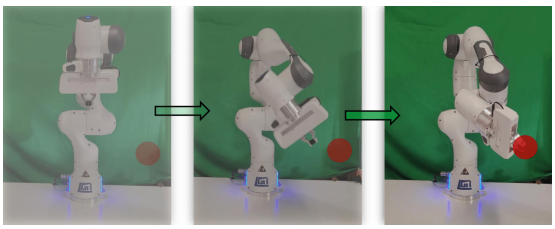
Parameter	J1	J2	J3	J4	J5	J6	J7
Old values							
$\varphi_{1,j}$	0.55	0.87	0.64	1.28	0.84	0.30	0.56
$\varphi_{2,j}$	5.12	9.07	10.14	5.59	8.35	17.13	10.34
$\varphi_{3,j}$	0.04	0.03	-0.05	0.04	0.03	-0.02	0.01
New values							
$\varphi_{1,j}$	3.40	1.98	2.51	2.44	2.60	1.19	1.55
$\varphi_{2,j}$	9.67	28.67	9.83	18.78	4.19	5.75	1.70
$\varphi_{3,j}$	0.01	0.01	-0.01	0.01	0.01	-0.01	0.01



(a) Visualization of the Panda robot reaching the goal in the Mujoco simulator.



(b) Visualization of the Panda robot reaching the goal position in the Gazebo simulator.



(c) Visualization of the Panda robot reaching the goal position in the real-world environment.

Figure 3.11. Comparison of simulated and real torque profiles for selected joints of the Franka Emika Panda after dynamic parameter estimation. Each plot illustrates the improved alignment between simulated and measured torques.

Each trial consists of 1,000,000 environment iterations, and only those configurations that successfully complete the final stage of the curriculum, representing the full seven-joint target task, are retained for evaluation.

The top-performing trial for each algorithm is selected according to the highest final episode reward, as reported in Tab.3.5. The optimized hyperparameter settings reveal that both algorithms benefit from relatively small learning rates and large batch sizes, which contribute to stable convergence under curriculum learning.

As shown in Fig. 3.13, the SAC agent trained under curriculum learning (CL) consistently outperforms both its PPO counterpart and the fine-tuned SAC agent trained in the concurrent

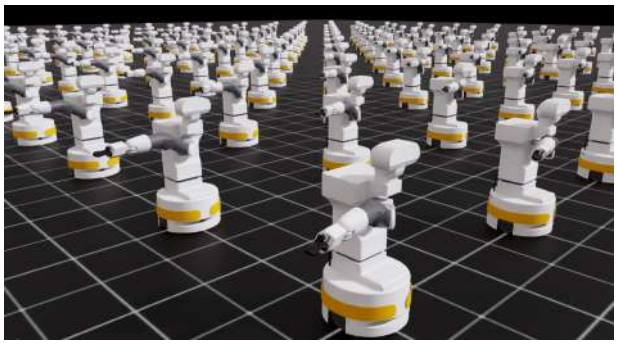
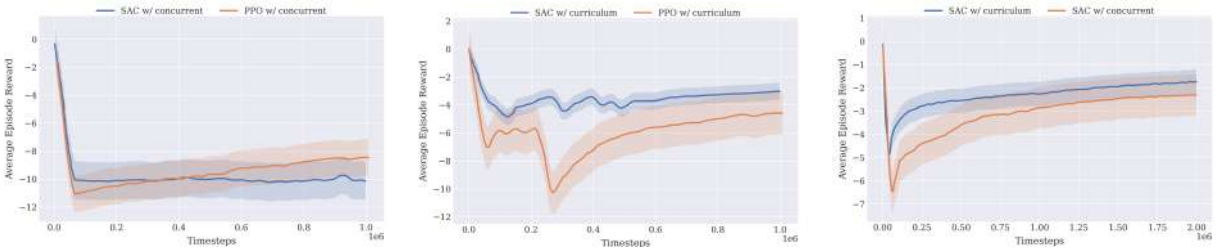


Figure 3.12. Isaac Lab environments used for training and evaluation of the Digital Twin.



(a) Concurrent training (default HP) **(b)** Curriculum training with HPO **(c)** Fine-tuning SAC

Figure 3.13. Training performance comparison: learning curves of the agents. Smooth mean reward and standard deviation are displayed.

setup. SAC demonstrates higher rewards and faster convergence across all training configurations. In particular, under the CL framework, the final reward reaches a peak of 1.71, whereas the concurrently trained SAC model plateaus at approximately -2.33. This performance gap underscores the effectiveness of the staged learning process introduced by curriculum learning, which reduces variance in gradient updates and facilitates more stable convergence.

The final SAC policy obtained through curriculum learning and fine-tuning is subsequently deployed within the Isaac Lab simulator to assess its ability to replicate real-world joint velocity responses. A benchmark sinusoidal velocity command is applied simultaneously to all seven joints. To ensure deterministic evaluation, the deployed policy operates without exploration noise. Velocity outputs are recorded within a single-instance Isaac Lab environment and compared across all joints. The resulting performance, depicted in Fig. 3.14, shows that the simulation faithfully reproduces the real robot's joint velocity dynamics, confirming that the optimized Digital Twin achieves a high level of dynamic fidelity at the velocity-controlled joint level.

3.3 Robotic Distance Functions

Within this broader perspective, this section focuses on the geometric representation of the robot and its surrounding environment through continuous distance fields. We extend the classical notion of the Signed Distance Function (SDF) to the Robotic Distance Function (RDF), originally introduced in [55], which embeds the robot's kinematic and geometric parameters directly into the distance formulation.

This extension turns the geometric model into an active component of the control and planning pipeline, enabling its use for collision checking and avoidance within robot control loops. In particular, RDFs provide differentiable computations of distances and gradients that can be exploited for motion generation, collision avoidance, and task-priority control.

Table 3.5. Best hyperparameters and final rewards under curriculum learning.

Alg.	Rew.	Env.	lr	batch	γ	ent	others
PPO	-4.58	256	1.4×10^{-5}	512	0.918	1.1×10^{-4}	steps:64, ep:9
SAC	-3.02	128	3.0×10^{-4}	512	0.905	0.0067	buf:500K, τ:0.059

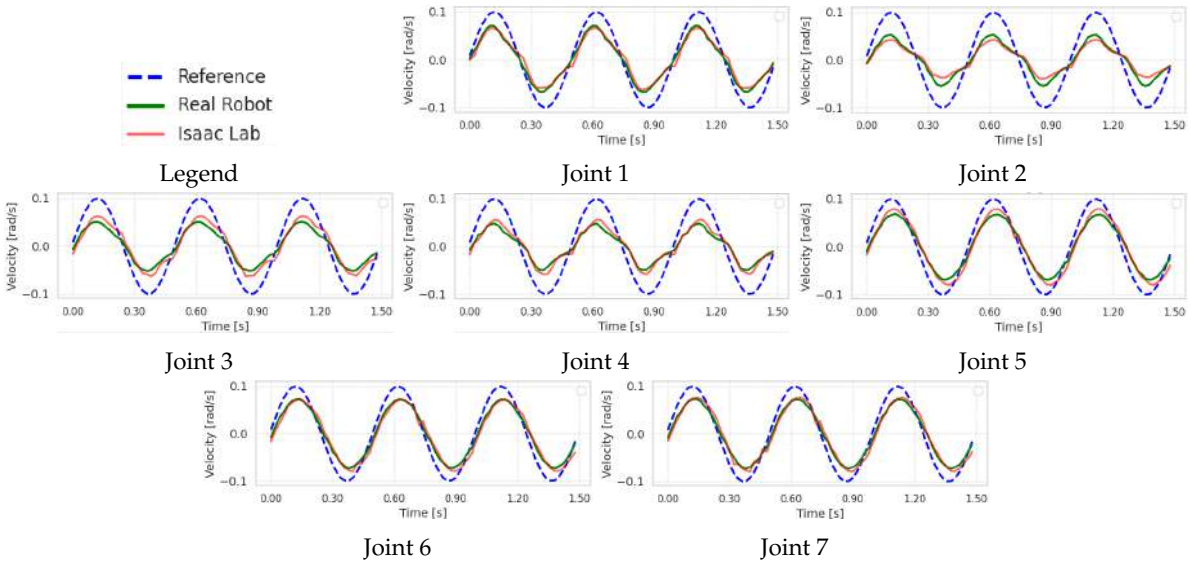


Figure 3.14. Comparison of joint velocities between Reference (blue dashed), Real Robot (green), and Isaac Lab (red) across all joints.

Emphasis is placed on computational efficiency and implementation strategies that ensure real-time performance, preserving compatibility with the temporal constraints of robotic systems. The discussion thus bridges the abstract domain of mesh-based modeling with its practical implementation, evaluating the performance of RDFs and their contribution to robust real-world robotic applications. In the following, this methodology is adopted to build a reactive controller within the Task-Priority framework and to perform collision checking, as detailed in Sec. 3.4.

Given a closed surface $\mathcal{S} \subset \mathbb{R}^3$, representing for instance the boundary of a robot link or an external obstacle, the SDF function assigns to any point $p \in \mathbb{R}^3$ its signed Euclidean distance from $\mathcal{S}$:

$$f(p) = \begin{cases} d(p,\mathcal{S}) & \text{if } p \text{ is outside of } \mathcal{S}, \\ 0 & \text{if } p \text{ is on the surface } \mathcal{S}, \\ -d(p,\mathcal{S}) & \text{if } p \text{ is inside of } \mathcal{S}, \end{cases} \tag{3.26}$$

where $d(\cdot,\mathcal{S})$ denotes the Euclidean distance.

The surface $\mathcal{S}$ can thus be recovered as the *0-level set* of the function f, that is

$$\mathcal{S} = \{p \in \mathbb{R}^3 \mid f(p) = 0\} \tag{3.27}$$

The SDF provides a continuous and compact representation of complex geometries. Negative values of $f(p)$ indicate that p lies inside the solid, positive values indicate that p is outside, while $f(p) = 0$ corresponds exactly to the boundary.

This representation is particularly well-suited for robotics because:

- it is continuous and differentiable almost everywhere,
- it provides direct access to surface normals through the gradient $\nabla f(p)$,

- it enables efficient integration into motion planning, collision avoidance, and manipulation tasks.

Despite these advantages, the exact computation of SDFs is often expensive, especially for complex surfaces or articulated robots. For this reason, practical implementations typically rely on learned or approximated representations that preserve continuity and differentiability while reducing computational cost.

3.3.1 Extending SDF to RDF

Consider a robot with joint configuration $q \in \mathbb{R}^C$, composed of K rigid links. Each link $k \in \{1, \dots, K\}$ is associated with a homogeneous transformation

$$T_b^k(q) \in SE(3), \tag{3.28}$$

which maps points from the local frame of link k to the robot base frame, while its inverse maps base coordinates into the local frame of the link.

Given the signed distance function $f_k(\cdot)$ of link k expressed in its local coordinates, the RDF is defined as

$$f_R(p_b, q) = \min_{k=1,\dots,K} f_k\Big((T_b^k(q))^{-1} p_b\Big), \tag{3.29}$$

where $p_b \in \mathbb{R}^3$ is a query point expressed in the robot base frame. Equation (3.29) encodes the global signed distance of the robot geometry as the minimum of the link-wise distances, each evaluated in its own local frame.

Equation (3.29) illustrates the core idea of the RDF: any query point in world space is first mapped into the local frame of each link through the inverse kinematic transformation, and then evaluated using the corresponding link SDF. The global robot distance field f_R is obtained by taking the minimum value across all links.

3.3.2 Approximation via Bernstein Bases

In the presented framework, the SDF of each robot link k is approximated by a weighted linear combination of basis functions:

$$f_k(p) = \Psi^\top(p)\, w_k, \tag{3.30}$$

where $\Psi(p)$ is a vector of basis functions evaluated at the query point p, and w_k is the weight vector learned for link k. Since the Bernstein basis is defined on $[0,1]$, in practice p is first normalized to $t(p)$ (defined below) and Ψ is evaluated at $t(p)$; for readability we keep the notation $\Psi(p)$. This formulation is general and allows different families of basis functions to be employed, depending on the desired trade-off between expressiveness, smoothness, and computational cost.

A particularly effective choice is given by the **Bernstein polynomials**, which provide a compact and smooth representation with guaranteed continuity. Since these basis functions are defined only on the unit interval $[0,1]$, a point $p \in [d_{\min}, d_{\max}]^3$ from the original domain must first be normalized as $t(p) = \frac{p - d_{\min}}{d_{\max} - d_{\min}} \in [0,1]^3$. The basis vector is then defined as the tensor product of univariate Bernstein polynomials:

$$\Psi(t) = \Phi(t_x) \otimes \Phi(t_y) \otimes \Phi(t_z), \tag{3.31}$$

with $\Phi(\cdot) = [\phi_0(\cdot) \ \cdots \ \phi_{N-1}(\cdot)]^\top$ and

$$\phi_n(t) = \binom{N-1}{n} t^n (1-t)^{N-1-n}, \quad t \in [0,1] \tag{3.32}$$

This construction allows for adjustable approximation resolution by varying the polynomial order N.

3.3.3 Training Via Basis Function

The training of an SDF consists of learning the weights w_k from a dataset of samples:

$$\mathcal{D} = \{(p_i, f(p_i))\}_{i=1}^{T}$$

where each $p_i \in \mathbb{R}^3$ is a point sampled in space and $f(p_i)$ is the signed distance value computed from the reference mesh.

Dataset Generation

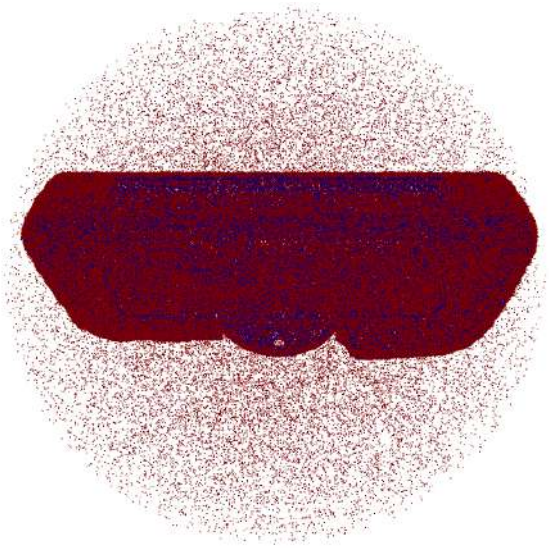


Figure 3.15. Example of a Franka Emika Panda hand link SDF dataset extracted using [74]. The blue dataset represents points inside the object, while the red one represents points outside the object.

To generate training data from a mesh, we employed the open-source library `mesh-to-sdf`, [46], [74]. This library implements a sampling procedure that produces pairs $(p_i, f(p_i))$ according to two main criteria:

- **Near-surface sampling:** points are sampled more densely near the mesh surface, in order to capture the *zero-crossing* of the SDF with higher accuracy.
- **Uniform sampling:** additional points are uniformly sampled in the surrounding space, to regularize the learning process far from the surface.

Domain Normalization

Following the DeepSDF approach [74], each mesh is scaled and translated to fit inside a unit sphere centered at the origin. This normalization step has two main advantages:

1. **Uniform domain:** data are always sampled within the same interval $[-1, 1]^3$, avoiding scaling issues across different meshes.
2. **Numerical stability:** training becomes more stable since input and output values remain within compact and controlled ranges.

Formally, let $c \in \mathbb{R}^3$ be the centroid of the mesh bounding box and $s \in \mathbb{R}^+$ the maximum radius (distance between c and the mesh vertices). Then, mesh points p are transformed as

$$\tilde{p} = \frac{p - c}{s}, \tag{3.33}$$

so that $\|\tilde{p}\|_2 \leq 1$. The SDF is computed on these normalized points, and during deployment the transformation can be inverted to map predictions back to the original space.

Weight Learning

The learning problem is formulated as a least-squares regression:

$$w^* = (\Psi^\top \Psi)^{-1} \Psi^\top f, \tag{3.34}$$

but for computational efficiency a recursive formulation is often used (e.g., Kalman filter-like updates), which allows for mini-batch processing and progressive weight updates. More detail in [55].

3.3.4 From Normalized Training to $\mathbb{R}^3$

The training strategy described above (Sec. 3.3.3), based on `mesh-to-sdf` sampling and DeepSDF-inspired normalization, ensures stable learning of SDF. However, it also introduces an intrinsic limitation: the dataset is defined only within a bounded region (typically a normalized sphere of radius one around the mesh). As a result, the learned SDF is strictly valid only inside this region, while outside it remains undefined.

Prepare data for SDF evaluation

Since the SDF is learned in the normalized space

$$\tilde{p} = \tfrac{p-c}{s}, \tag{3.35}$$

any query point must be mapped into this space before evaluation. Here, $c \in \mathbb{R}^3$ denotes the centroid of the mesh bounding box, while $s \in \mathbb{R}^+$ is scale factor that ensures the mesh fits inside the unit sphere (see Sec. 3.3.3).

At inference time, the signed distance in original units is obtained as

$$f(p) \;=\; s\,\hat{f}\left(\tfrac{p-c}{s}\right), \tag{3.36}$$

3.3.5 Domain Extension via Projections

In robotic applications, distance queries, already scaled as discussed in Sec. 3.3.4, must also be evaluated far from the normalized training domain. If left unaddressed, this mismatch between the *training domain* and the *operational domain* would cause discontinuities and unreliable distance values.

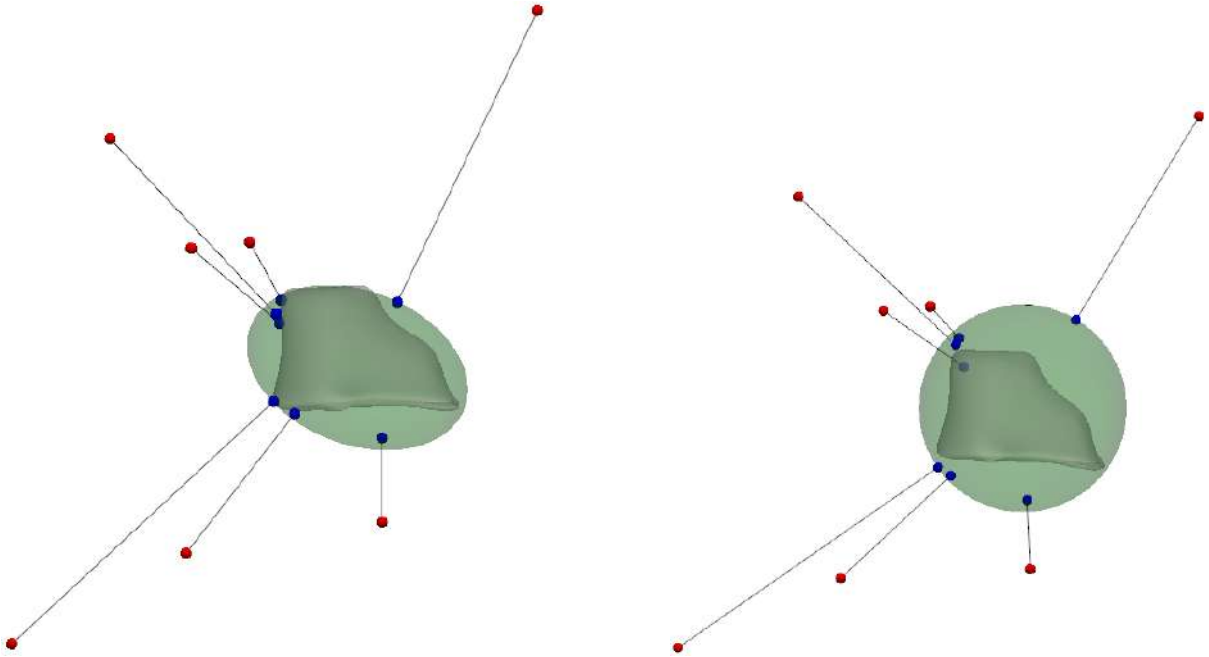
To overcome this limitation, we extend the definition of the SDF beyond the training bounds by introducing geometric *projections*. The general idea is the following: when a query point p lies outside the training domain, it is first projected back onto a bounding surface enclosing the training set, and the SDF is then defined as the sum of the SDF at the projected point plus the Euclidean distance between p and its projection.

Two alternative projection strategies are implemented:

- a **spherical projection**, which is computationally efficient and numerically stable,

- an **ellipsoidal projection**, which better adapts to the anisotropic geometry of each link, but it requires numerical iterative optimization.

Both strategies preserve continuity and differentiability of the resulting distance function, while trading off precision and computational cost. The two methods are described below.



(a) Ellipsoidal projection of a point p onto the training domain boundary (ellipsoid fitted to the training data).

(b) Spherical projection of a point p onto the training domain boundary (sphere of radius r_1).

Figure 3.16. Comparison between ellipsoidal and spherical projection strategies used to extend the SDF domain beyond the training region.

Spherical Projection

The simplest strategy assumes that the training region is a sphere of radius r_1 centered at x_0. If the dataset has been normalized as in Sec. 3.3.3, then $x_0 = 0$ and $r_1 = 1$. Given a query point $p = (x, y, z)$, we first compute its distance from the center:

$$r_2 = \|p - x_0\|. \tag{3.37}$$

If $r_2 \leq r_1$, the point lies inside the training domain and no modification is needed. Otherwise, p is projected onto the spherical surface as

$$p_{\text{sphere}} = x_0 + r_1 \frac{p - x_0}{r_2}. \tag{3.38}$$

The distance offset is given by

$$d_{\text{sphere}} = r_2 - r_1. \tag{3.39}$$

The extended SDF is then defined as

$$f(p) = f(p_{\text{sphere}}) + d_{\text{sphere}}. \tag{3.40}$$

This method is computationally inexpensive since the projection is closed-form, requiring only vector normalization. Moreover, as points move farther away from the object, the SDF

naturally tends to resemble that of a sphere, which justifies the approximation. For this reason, spherical projection is often preferable in real-time control scenarios where efficiency is critical.

Ellipsoidal Projection

Fitting To obtain a tight and anisotropic bounding region, we fit an ellipsoid to the training point cloud. We start from the subset of samples that lie inside (or on) the object surface,

$$\mathcal{P}_{\text{in}} = \{p_i \,|\, (p_i, f(p_i)) \in \mathcal{D},\ f(p_i) \leq 0\}\,,$$

so that the resulting ellipsoid remains bounded within the normalized domain.

The ellipsoid is centered at the centroid of these points,

$$\mu = \frac{1}{|\mathcal{P}_{\text{in}}|} \sum_{p_i \in \mathcal{P}_{\text{in}}} p_i, \tag{3.41}$$

and aligned along the principal directions of their covariance matrix,

$$\Sigma = \frac{1}{|\mathcal{P}_{\text{in}}|} \sum_{p_i \in \mathcal{P}_{\text{in}}} (p_i - \mu)(p_i - \mu)^\top. \tag{3.42}$$

By performing eigen-decomposition $\Sigma = \Lambda D \Lambda^\top$, the eigenvectors Λ define the ellipsoid orientation, and the square roots of the eigenvalues in $D = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ define the initial axis lengths:

$$a_0 = 2\sqrt{\lambda_1}, \quad b_0 = 2\sqrt{\lambda_2}, \quad c_0 = 2\sqrt{\lambda_3}. \tag{3.43}$$

To keep the ellipsoid bounded within the normalized domain (Sec. 3.3.3), a uniform scaling factor

$$s = \frac{1}{\max\left(\frac{a_0}{2}, \frac{b_0}{2}, \frac{c_0}{2}\right)} \tag{3.44}$$

is applied to all axes. The final semi-axes are therefore

$$a = s a_0, \quad b = s b_0, \quad c = s c_0. \tag{3.45}$$

The resulting ellipsoid provides a compact anisotropic bounding surface for projection. Note that not every point in $\mathcal{P}_{\text{in}}$ is guaranteed to lie inside the ellipsoid, but this is not a strict requirement for the projection to work effectively.

Projection A query point p is first mapped into the ellipsoid local coordinate system:

$$p_e = \Lambda^\top (p - \mu), \tag{3.46}$$

where Λ contains the eigenvectors of the covariance matrix. For convenience, let $p_e = (x_e, y_e, z_e)^\top$. The projection onto the ellipsoidal surface is obtained by solving the constrained minimization problem

$$p^* = \arg\min_u \|u - p_e\|^2 \quad \text{s.t.} \quad \frac{u_1^2}{a^2} + \frac{u_2^2}{b^2} + \frac{u_3^2}{c^2} = 1, \tag{3.47}$$

which can be solved with the Newton–Raphson method. To project points on the ellipsoid surface efficiently, the method introduced by Eberly [21, 14] is adopted, which reformulates the problem into minimizing the following function parametrized by the Lagrange multiplier λ

$$F(\lambda) = \left(\frac{a x_e}{a^2 + \lambda}\right)^2 + \left(\frac{b y_e}{b^2 + \lambda}\right)^2 + \left(\frac{c z_e}{c^2 + \lambda}\right)^2 - 1 \tag{3.48}$$

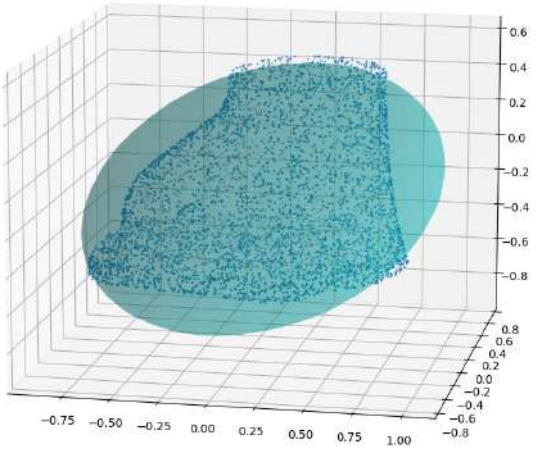


Figure 3.17. Ellipsoid fitted to the training point cloud of a robot link. The ellipsoid is centered at the centroid of the points and aligned with their principal directions, providing a tight bounding volume for projection.

The minimization problem is solved by using the Newton-Raphson method:

$$\lambda_{k+1} = \lambda_k - \frac{F(\lambda_k)}{F'(\lambda_k)}. \tag{3.49}$$

The algorithm is stopped when $|F(\lambda_k)| < \varepsilon$. The corresponding value of $\lambda_k = \lambda^*$ is used to project the point p on the ellipsoid surface. This projection, namely p^*, is expressed in global coordinates as

$$p^* = \Lambda \begin{bmatrix} \frac{a^2 x_e}{a^2+\lambda^*} \\ \frac{b^2 y_e}{b^2+\lambda^*} \\ \frac{c^2 z_e}{c^2+\lambda^*} \end{bmatrix} + \mu \tag{3.50}$$

Finally, the extended SDF is defined as

$$f(p) = f(p^*) + \|p - p^*\|. \tag{3.51}$$

The ellipsoidal projection is more computationally demanding than the spherical one, but provides a tighter and more accurate extension of the SDF domain, especially for elongated or asymmetric robot links. It is therefore the method of choice in tasks where precision outweighs efficiency, such as high-fidelity simulation or manipulation in cluttered environments.

Remark on normalization and prediction. Since the SDF is learned in the normalized space $\tilde{p} = \frac{p-c}{s}$, at inference the value in original units is obtained as

$$f(p) \;=\; s\,\hat{f}\left(\frac{p-c}{s}\right), \tag{3.52}$$

where $\hat{f}$ is the SDF learned in normalized coordinates. For spherical/ellipsoidal projections, the projection is computed in the normalized space, and only the final distance value is rescaled by s.

Remark 3.3.1. From this point onward, to generalize the projection notation, we denote by $\pi(p)$ the spherical or ellipsoidal projection of a point p.

3.3.6 Surface Analysis and Gradients

A distinctive advantage of SDFs is their differentiability. The gradient $\nabla f(p) \in \mathbb{R}^3$ provides the outward normal vector at the closest surface point. In RDF, the gradient is computed directly from the Bernstein basis representation:

$$\nabla f(p) = \nabla \Psi(p)\, w, \tag{3.53}$$

with

$$\begin{aligned} \nabla_x \Psi &= \Phi'(p_x) \otimes \Phi(p_y) \otimes \Phi(p_z), \\ \nabla_y \Psi &= \Phi(p_x) \otimes \Phi'(p_y) \otimes \Phi(p_z), \\ \nabla_z \Psi &= \Phi(p_x) \otimes \Phi(p_y) \otimes \Phi'(p_z). \end{aligned} \tag{3.54}$$

Here, accordling to Sec. 3.3.2, any query point $p \in [d_{\min}, d_{\max}]^3$ must first be normalized, and $\Phi'(t(p))$ contains the derivatives of the Bernstein polynomials. The smoothness of these functions guarantees that the gradient field is stable and suitable for control.

When computing derivatives, this change of variables must be considered. By the chain rule,

$$\frac{\partial \phi_n}{\partial p} = \frac{\partial \phi_n}{\partial t} \cdot \frac{\partial t}{\partial p} = \frac{\partial \phi_n}{\partial t} \cdot \frac{1}{d_{\max} - d_{\min}}. \tag{3.55}$$

Therefore, the gradient of the basis functions with respect to the original coordinates p includes a scaling factor given by the domain length $(d_{\max} - d_{\min})$.

In compact vector form, this relationship can be written as

$$\nabla_p \Psi(p) = \frac{1}{d_{\max} - d_{\min}} \nabla_t \Psi(t(p)), \tag{3.56}$$

where $\nabla_t \Psi$ denotes the gradient of the Bernstein basis with respect to the normalized coordinates. Inside the sphere ($\|p\| \leq r_1$), the gradient coincides with the Bernstein approximation, but outside the projection must be considered, as discussed in the next section.

3.3.7 Gradient Correction after Projection

When extending the SDF beyond its training domain, the Bernstein gradient $\nabla_B f(p)$ (simply ∇_B) is evaluated at the *projected point* $\pi(p)$, while the SDF is assigned to the *query point* p. As a result, the gradient field and the scalar field become spatially inconsistent: the direction of $\nabla_B(\pi(p))$ does not generally represent the true direction of $\nabla_B(p)$. This mismatch occurs because ∇_B is computed in a different spatial location, without accounting for the geometric distortion introduced by the projection itself.

Figure 3.18 illustrates this phenomenon in the spherical case. Each subplot shows the same signed distance field (color map) along with the query point p (red circle) and its projection $\pi(p)$ onto the training boundary (blue point). The dashed blue line indicates the projection vector connecting them. The orange, green, and purple arrows represent the Bernstein gradient ∇_B, the spherical correction ∇_S, and the final total gradient ∇_T, respectively.

Although ∇_B correctly describes the local variation of the field *at* the projected point, it fails to represent the correct direction at the query location p. This occurs because the gradient is evaluated in a different region of the field, so its orientation does not correspond to the actual direction of increasing distance at p.

To recover a consistent and continuous gradient field, a correction is applied by combining ∇_B with the radial component ∇_S:

$$\nabla_T = \nabla_B + \beta \nabla_S, \tag{3.57}$$

where $\nabla_S = \frac{p}{\|p\|}$ and β is a blending factor controlling the influence of the spherical term. This correction ensures that ∇_T points in the correct direction of growth of the signed distance while preserving the smoothness inherited from the Bernstein representation.

The same reasoning applies to the ellipsoidal projection, where ∇_S is replaced by the local outward direction $\nabla_E = \frac{p - \pi_e(p)}{\|p - \pi_e(p)\|}$, and the transformation of ∇_B is handled through the ellipsoidal Jacobian J_{π_e}. In both cases, this correction restores the coherence between the scalar and vector fields, yielding a geometrically consistent signed distance representation suitable for control and optimization.

Gradients with Spherical Projection

The Bernstein approximation introduced in Sec. 3.3.2 is valid only inside the training domain, typically bounded by a unit sphere of radius denoted r_1. For points outside the training sphere, $\|p\| > r_1$, the signed distance function is defined as

$$f(p) = f_B(\pi(p)) \; + \; d(p), \tag{3.58}$$

where $\pi(p)$ is the spherical projection of p onto the sphere surface, and $d(p)$ is the Euclidean distance of the projection.

Since f_B is evaluated at the projected point $\pi(p)$ rather than at p, the gradient is distorted by the projection. To recover the correct gradient with respect to the original coordinates, the chain rule must be applied:

$$\nabla_p \big(f_B(\pi(p))\big) = J_\pi(p)^\top \, \nabla f_B(\pi(p)), \tag{3.59}$$

where the Jacobian of the spherical projection is

$$J_\pi(p) = \frac{r_1}{\|p\|} \left(I - \frac{p p^\top}{\|p\|^2} \right). \tag{3.60}$$

The additional distance term contributes a purely radial component:

$$\nabla d(p) = \frac{p}{\|p\|}. \tag{3.61}$$

Combining the two effects, the gradient outside the sphere is

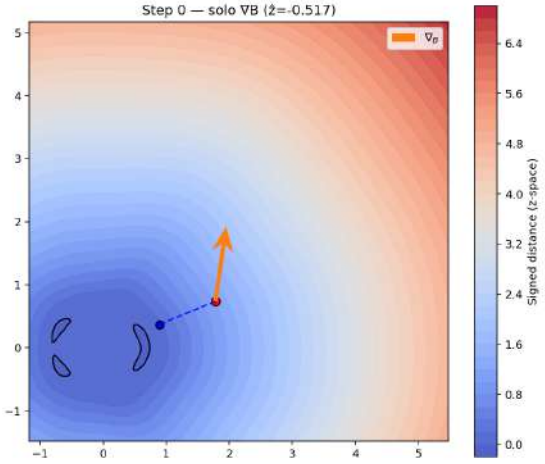
$$\nabla f(p) = J_\pi(p)^\top \, \frac{1}{d_{\max} - d_{\min}} \, \nabla_t \Psi\big(t(\pi(p))\big) \, w \; + \; \frac{p}{\|p\|}. \tag{3.62}$$

The first term is the Bernstein gradient, corrected by the Jacobian to remove the distortion due to projection, while the second term enforces the proper radial behaviour of the distance field.

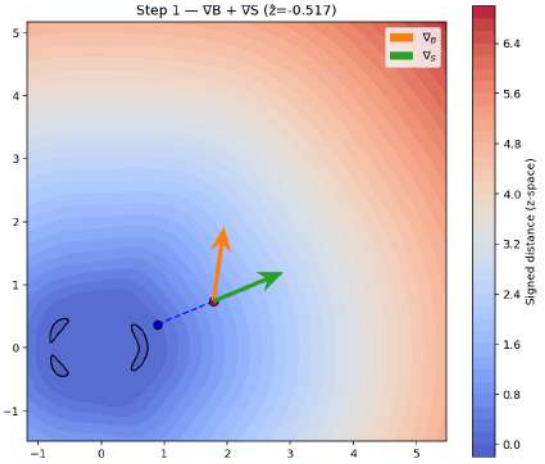
Gradients with Ellipsoidal Projection

Analogously to the spherical case discussed in Sec. 3.3.6, when a query point p lies outside the ellipsoidal training domain, the signed distance function is defined as:

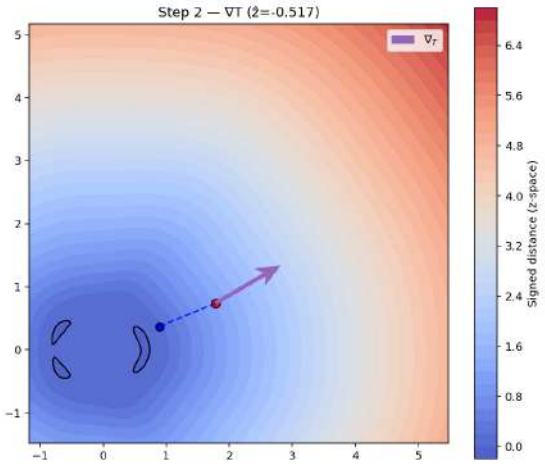
$$f(p) = f_B(\pi_e(p)) \; + \; \|p - \pi_e(p)\|, \tag{3.63}$$



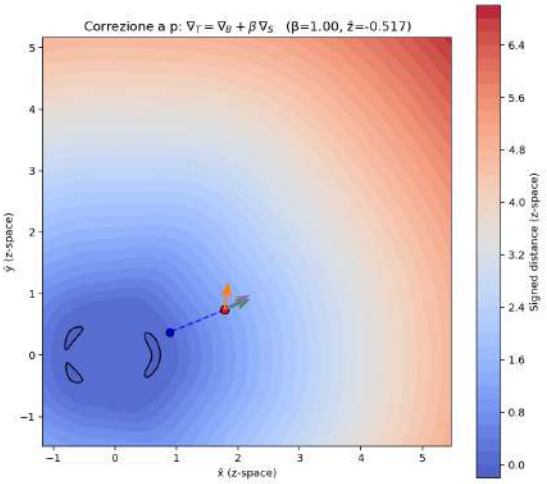
(a) Initial condition: the gradient ∇_B is evaluated at the projected point $\pi(p)$, producing a misaligned direction at the query point.



(b) Addition of the spherical term ∇_S, which introduces the correct radial component from $\pi(p)$ toward p.



(c) Combination of the two fields to obtain the total gradient $\nabla_T = \nabla_B + \beta\nabla_S$, properly oriented at p.



(d) Final corrected field: the total gradient ∇_T (purple) matches the expected direction of increasing distance at p.

Figure 3.18. Step-by-step illustration of the gradient correction process for spherical projection. The query point p (red) lies outside the training domain, and its projection $\pi(p)$ (blue) lies on the boundary. The Bernstein gradient ∇_B (orange) is evaluated at the wrong spatial location, leading to misalignment. Adding the spherical component ∇_S (green) restores the correct direction, and their combination yields the total gradient ∇_T (purple), aligned with the true variation of the signed distance. The same mechanism applies to ellipsoidal projections, where ∇_S is replaced by the local outward direction ∇_E.

where $\pi_e(p)$ is the projection of p onto the ellipsoid surface, as defined in Sec. 3.3.5. Inside the ellipsoid, the field coincides with the Bernstein approximation $f_B(p)$.

To compute the gradient of $f(p)$ with respect to p, the chain rule is again applied:

$$\nabla_p f(p) = J_{\pi_e}(p)^\top \nabla f_B(\pi_e(p)) \; + \; \nabla \|p - \pi_e(p)\|, \tag{3.64}$$

where $J_{\pi_e}(p)$ is the Jacobian of the ellipsoidal projection.

The exact analytical Jacobian of the ellipsoidal projection π_e involves the solution of a nonlinear system depending on the Lagrange multiplier λ^* (see Sec. 3.3.5), and is thus cumbersome to compute in closed form. For efficient GPU evaluation, an approximate but geometrically consistent Jacobian is adopted. Let $\Lambda \in \mathbb{R}^{3\times3}$ denote the eigenvector matrix defining the ellipsoid orientation, and let

$$p_e = \Lambda^\top (p - \mu) \tag{3.65}$$

be the query point expressed in the ellipsoid coordinate frame, with μ the ellipsoid center and (a, b, c) its semi-axes.

The local Jacobian approximation is constructed as

$$J_e(p_e) = \frac{r_1}{\|p_e\|} \left(I - \frac{p_e p_e^\top}{\|p_e\|^2} \right), \tag{3.66}$$

which projects the gradient onto the tangent plane of the ellipsoid and rescales it according to the local radial ratio $\frac{r_1}{\|p_e\|}$. This Jacobian generalizes the spherical case (Sec. 3.3.5) by replacing the isotropic radius with the anisotropic geometry of the ellipsoid.

Mapping back to the world coordinates yields

$$J_{\pi_e}(p) = \Lambda \, J_e(p_e) \, \Lambda^\top. \tag{3.67}$$

Combining the above terms, the overall gradient of the extended SDF is:

$$\nabla f(p) = J_{\pi_e}(p)^\top \frac{1}{d_{\max} - d_{\min}} \nabla_t \Psi\big(t(\pi_e(p))\big) \, w \; + \; \frac{p - \pi_e(p)}{\|p - \pi_e(p)\|}. \tag{3.68}$$

The first term accounts for the Bernstein gradient distorted by the projection geometry, while the second term represents the gradient of the added distance offset, pointing along the normal direction from $\pi_e(p)$ to p.

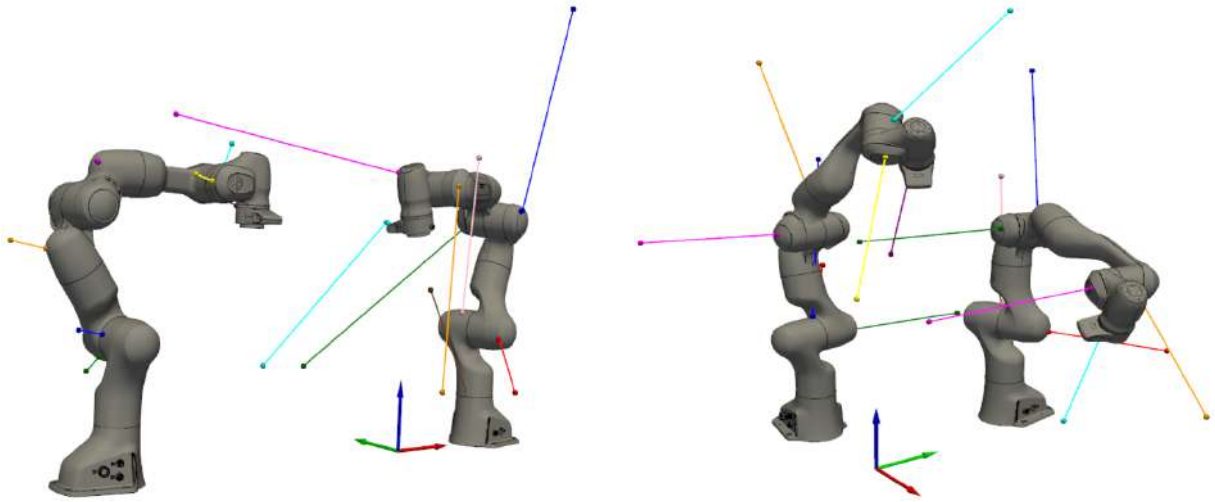
This approximation retains differentiability and geometric consistency while being efficient to compute in batched form on GPU, as implemented in Sec. 3.3.9.

3.3.8 Point Projection onto Surfaces

Another powerful property of SDFs is their ability to project any point p onto the implicit surface defined by the zero-level set $f(p) = 0$. RDF employs an iterative algorithm initialized with $p_0 = p$, and updates given by:

$$p_{n+1} = p_n - \alpha_n \frac{\nabla f(p_n)}{\|\nabla f(p_n)\|}, \tag{3.69}$$

where α_n is the step size. The iteration stops once $|f(p_n)| < \varepsilon$. This yields the projected point $\hat{p} = p_n$, which is guaranteed to be orthogonal to the surface. This operation is crucial both for geometric analysis and for defining contact points in robotics. Results of the projection are shown in Fig. 3.19, where the projection of random points onto the surface of two Panda robot links is visualized. The colored segments represent the local descent direction of the sampled points, demonstrating how the algorithm converges to the surface in just a few iterations.



(a) Dual-Panda configuration #1 (base placement and joint configuration).

(b) Dual-Panda configuration #2 (base placement and joint configuration).

Figure 3.19. Visualization of the projection step used to map sampled points to the SDF zero level set in a dual-Panda setup. For each link, points are initialized randomly in the workspace and updated via gradient descent; colored segments depict the link-local descent direction of a random sampled point. Using the SDF value as an adaptive step size is sufficient for fast convergence in 1-2 iterations.

3.3.9 Batch RDF for GPU Acceleration

Vectorized evaluation.

To apply this to *all* links and *all* points at once, we stack the data into tensors. Let $p_k \in \mathbb{R}^{M\times 3}$ be the M query points expressed in the local frame of link k. For each link, the Bernstein basis is evaluated in parallel at all M points:

$$\Phi_k = \begin{bmatrix} \Psi(p_{k,1})^\top \\ \vdots \\ \Psi(p_{k,M})^\top \end{bmatrix} \in \mathbb{R}^{M\times N}. \tag{3.70}$$

and the corresponding SDF values are obtained by $f_k = \Phi_k\, w_k \in \mathbb{R}^M$.

Stacking across all links yields the global tensors $\Phi \in \mathbb{R}^{L\times M\times N}$, $W \in \mathbb{R}^{L\times N}$, and the complete evaluation reduces to a single tensor contraction:

$$F = \langle \Phi, W \rangle_N \quad \in \quad \mathbb{R}^{L\times M} \tag{3.71}$$

where F_{ij} is the signed distance of point j with respect to link i.

Gradients.

If gradients are required, the same structure holds by replacing Φ with its derivative tensor:

$$\nabla F = \langle \nabla\Phi, W \rangle_N \quad \in \quad \mathbb{R}^{L\times M\times 3} \tag{3.72}$$

Multi configurations.

The formulation of Eq. (3.29) naturally extends to multiple robot configurations in a vectorized fashion. Let $q_1, \ldots, q_C$ denote C joint configurations and

$$T_w^i(q_c) \in SE(3), \qquad i = 1, \ldots, L, \;\; c = 1, \ldots, C,$$

Table 3.6. Computational performance comparison between spherical and ellipsoidal projection methods for different numbers of query points N generated randomly. Errors are reported as maximum and mean absolute deviations from the dataset SDF values, averaged over 1000 runs.

N_{points}	Spherical Projection			Ellipsoidal Projection			Speedup
	Error [m]		Time [ms]	Error [m]		Time [ms]	
	Max	Mean		Max	Mean		
			Bernstein basis: 8^3				
10^2	3.45×10^{-2}	1.35×10^{-2}	0.47	2.50×10^{-2}	1.13×10^{-2}	1.59	3.4×
10^3	3.74×10^{-3}	2.36×10^{-2}	0.47	2.10×10^{-2}	1.78×10^{-2}	1.54	3.3×
10^4	3.70×10^{-3}	2.32×10^{-2}	0.48	2.03×10^{-2}	2.23×10^{-2}	1.63	3.4×
10^6	5.69×10^{-3}	2.29×10^{-2}	0.49	4.01×10^{-2}	2.71×10^{-2}	33.64	69×
			Bernstein basis: 16^3				
10^2	2.45×10^{-2}	1.55×10^{-2}	0.46	2.80×10^{-2}	1.28×10^{-2}	1.52	3.3×
10^3	3.67×10^{-2}	2.18×10^{-2}	0.47	3.07×10^{-2}	1.73×10^{-2}	1.64	1.0×
10^4	5.02×10^{-2}	2.02×10^{-2}	1.66	4.60×10^{-2}	3.79×10^{-2}	2.08	1.8×
10^6	✗	✗	✗	✗	✗	✗	✗ (OOM)

be the forward kinematics of link i at configuration q_c. Given P query points $\{p_1, \ldots, p_P\}$ in world coordinates, each point is mapped into the local frame of every link and configuration:

$$p_{i,c,j} = \left(T_w^i(q_c)\right)^{-1} p_j. \tag{3.73}$$

The resulting tensor has shape $(C, L, P, 3)$, where each entry corresponds to one query point expressed in the local coordinates of a specific link at a specific configuration. This vectorized formulation allows all transformations and evaluations to be carried out in parallel on GPU, enabling efficient multi-link, multi-configuration distance queries.

The resulting tensor is reshaped into $(L, C \cdot P, 3)$, allowing all links, all configurations, and all query points to be evaluated in a single batched call to Eq. (3.71) and Eq. (3.72).

Padding

Note that each robot link's SDF can be modeled with a basis of varying resolution, allowing the complexity of the representation to be adapted to the geometry of each link. For instance, while some links of the Panda manipulator can be accurately described with eight basis functions per dimension, others require twelve to capture finer details. To make this approach practical for implementation on parallel hardware, zero-padding is adopted by extending smaller weight vectors w_i to match the dimensionality of the largest basis N^3_{highest}. Formally, each padded weight vector is defined as: $\tilde{w}_i = \begin{bmatrix} w_i \\ 0_{(N^3_{\text{highest}} - N_i^3)} \end{bmatrix} \in \mathbb{R}^{N^3_{\text{highest}}}$ where $N_{\text{highest}} = \max_i(N_i)$ is the largest number of basis functions per link.

This padding preserves the original SDF evaluation since the zero-extended components do not affect the result: $\tilde{\Psi}(p)\tilde{w}_i = \Psi(p)w_i + \psi(p)\,0 = f(p)$ where $\psi(p) \in \mathbb{R}^{N^3_{\text{highest}} - N^3}$ is the basis extension evaluated at the point of padding.

3.3.10 Results

The results in Table 3.6 emphasize the trade-off between accuracy and computational efficiency when extending SDFs via spherical or ellipsoidal projection. The **spherical projection** exhibits

higher numerical stability and significantly lower execution times, making it preferable for *real-time control* or online distance evaluation. In contrast, the **ellipsoidal projection** provides improved geometric fidelity—particularly for anisotropic or elongated shapes—since the fitted ellipsoid more closely approximates the true boundary of each link's training domain. However, this comes at a substantial computational cost that scales unfavorably with the number of query points N. For large-scale evaluations ($N \geq 10^5$), the ellipsoidal method becomes impractical due to GPU memory saturation (Out of Memory OOM), as indicated by the ✗ entries for $N = 10^6$ in Table 3.6.

From a control perspective, the projection method should therefore be selected based on the operating regime: for dense or high-frequency distance evaluations, the spherical formulation is more stable and computationally sustainable, whereas the ellipsoidal variant is better suited for offline optimization or high-precision modeling tasks.

The increase in the Bernstein basis resolution from 8^3 to 16^3 yields a modest reduction in both mean and maximum SDF errors, confirming that higher-order polynomial bases capture finer surface details. Nonetheless, the associated computational burden grows rapidly with N, due to the cubic scaling of the basis evaluation. As a result, high-resolution Bernstein models are beneficial primarily for low-dimensional queries or precomputed field maps, whereas for online applications, lower-order bases achieve a better balance between accuracy and speed.

Finally, it should be noted that the reference SDF dataset was generated using a mesh-based sampling procedure (`mesh-to-sdf`), which may not fully handle non-watertight or imperfect geometries. Such artifacts can introduce localized inconsistencies in the ground-truth distances, occasionally producing outlier errors. These are effectively attenuated by the recursive Kalman-based weight update strategy, which filters local noise and enforces smoothness in the reconstructed distance field.

Robot Distance Function vs other methods

The experiments and results are conducted on a system equipped with an Intel Core i7-12700H (12th generation, 2.7 GHz) processor and an NVIDIA GeForce RTX 4060 Ti GPU. It is worth mentioning that, to obtain reliable results, the quality of the input mesh plays a fundamental role. Notably, the dataset proposed in [74] does not robustly handle non-watertight meshes, often leading to poor sampling quality near topological defects such as holes or gaps. To address this issue, in this work all the meshes are preprocessed using the open-source library PyMeshFix [4], which automatically fills holes and repairs topological inconsistencies. To show the performance of the proposed method, we use the Franka Emika Panda robot as a benchmark, along with a table used as collision object in pick and place actions.

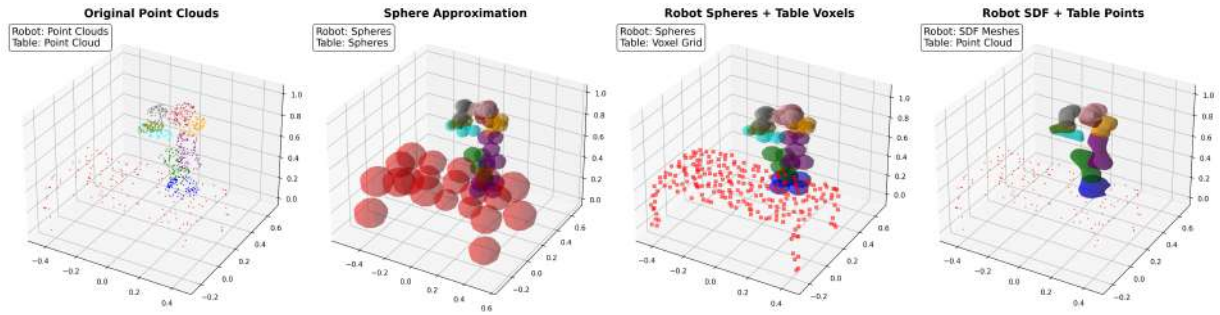


Figure 3.20. Comparison of surface approximation techniques for collision avoidance: Point-to-Point, Sphere-to-Sphere, Sphere-to-Voxel, and RDF-to-Point representations.

In Tab.3.7 this is explicitly marked as *Ref*. For completeness, we also report lower resolution point to point variants to complete the ablation study, to quantify the accuracy of the proposed method. In Tab. 3.7, the column Robot/Table surface approx reports the surface approximation

Table 3.7. Performance comparison of collision detection methods at different resolution levels (Low / Medium / High). Reported metrics: mean absolute error (MAE), standard deviation, both referenced to the point-to-point Chamfer distance baseline (first row), and average execution time.

Method	Robot/Table	MAE [mm]	Std [mm]	Time [ms]
Point-to-Point	L: 2700/500 (Pts/Pts)	23	15	0.9
	M: 9000/1000 (Pts/Pts)	11	12	1.8
	H: 22500/1500 (Pts/Pts)	Ref.	Ref.	10.9
Sphere-to-Point [3]	L: 9/500 (Sph/Pts)	47	30	0.4
	M: 45/1000 (Sph/Pts)	10	7	1.8
	H: 90/1500 (Sph/Pts)	6	5	5.2
Sphere-to-Sphere [11]	L: 9/18 (Sph/Sph)	105	32	0.1
	M: 45/36 (Sph/Sph)	34	11	1.1
	H: 90/72 (Sph/Sph)	21	1	2.9
Sphere-to-Voxel [6]	L: 9/61 (Sph/Vox)	131	39	0.3
	M: 45/2284 (Sph/Vox)	22	10	2.8
	H: 90/19355 (Sph/Vox)	19	17	32.6
Segment-to-Point [96]	L: 9/500 (Seg/Pts)	57	50	31.1
	M: 9/1000 (Seg/Pts)	52	43	32.0
	H: 9/1500 (Seg/Pts)	50	51	32.6
RDF-to-Point [55]	L: 8/500 (NFunc/Pts)	8	2	0.5
	M: 12/1000 (NFunc/Pts)	6	1	1.1
	H: 16/1500 (NFunc/Pts)	4	1	1.9

the number of surface *primitives* used to approximate the robot and the table surfaces, respectively. Depending on the method, these primitives are points, voxels, spheres, or segments, as shown in Fig. 3.20. The results highlight a clear trade off between computational cost and accuracy for discrete methods, such as point-, voxel- and sphere-based representations. These approaches require increasingly fine discretizations to maintain accuracy, which significantly raises computational load. Although accurate at high resolution, such methods scale poorly, limiting their applicability in real-time settings. By contrast, the RDF based approach leverages a continuous implicit surface model, providing stable and accurate distance estimates already at relatively low discretization, resulting in a more favorable accuracy and runtime balance.

Limitations on Multi-Configuration Evaluation

One of the principal computational bottlenecks of the Bernstein-based RDF representation arises from the three-dimensional tensor product of univariate bases. For a polynomial order N, the number of basis functions grows cubically as N^3, implying that the weight tensor $\mathcal{W}$ of each link has dimension $\mathbb{R}^{N\times N\times N}$. Both memory usage and computational cost of the SDF evaluation therefore scale as $\mathcal{O}(N^3)$, which quickly becomes prohibitive for high-resolution models or dense spatial sampling.

This issue is further amplified in the case of **multi-configuration batch evaluation**, where the same computation must be repeated for multiple robot states and query points. In this context, the total tensor size scales as

$$\mathcal{O}(C \times L \times P \times N^3),$$

where C is the number of robot configurations, L the number of links, and P the number of query points. Consequently, both the memory footprint and the computational load grow rapidly with the basis resolution N and with the number of evaluated configurations.

In practice, this leads to severe throughput and memory bottlenecks on GPU hardware. For instance, with the current setup on an NVIDIA RTX 4060 Ti, the feasible batch size is limited to 8-basis functions approximately **40 robot configurations**, each evaluated on **40 query points per link**, plus about **100 environment points** for collision checking. Beyond these limits, GPU memory saturation and kernel dispatch overheads cause significant slowdowns or execution failures, highlighting the necessity of low-rank tensor approximations such as the CP model introduced in Sec. 3.3.11.

3.3.11 Low-Rank Tensor Decomposition of RDF

Recall of the Bernstein tensor form. Starting from Eq. (3.30), the 3D Bernstein basis at point $p = (p_x, p_y, p_z)$ can be written as

$$\Psi(p) = \Phi_x(p_x) \otimes \Phi_y(p_y) \otimes \Phi_z(p_z) \in \mathbb{R}^{N^3}. \tag{3.74}$$

Therefore, the same scalar field is equivalently expressed as

$$f(p) = \Psi^\top(p)\, w = \sum_{i,j,k=1}^{N} w_{ijk}\, \phi_i(p_x)\, \phi_j(p_y)\, \phi_k(p_z). \tag{3.75}$$

Equation (3.75) makes explicit that the couplings among the three coordinates are fully encoded in the coefficient tensor w.

CP decomposition of the weight tensor. To reduce the cubic complexity of Eq. (3.75), we approximate w with a rank-R CANDECOMP/PARAFAC model:

$$w \approx w_{\mathrm{CP}} = \sum_{r=1}^{R} \lambda_r\, a_r \otimes b_r \otimes c_r, \tag{3.76}$$

where $a_r, b_r, c_r \in \mathbb{R}^N$ and $\lambda_r \in \mathbb{R}$.

Substituting Eq. (3.76) into Eq. (3.75), and using

$$(x \otimes y)^\top (u \otimes v) = (x^\top u)(y^\top v), \tag{3.77}$$

we obtain the separable CP form

$$f_{\mathrm{CP}}(p) = \sum_{r=1}^{R} \lambda_r \left(\Phi_x(p_x)^\top a_r\right) \left(\Phi_y(p_y)^\top b_r\right) \left(\Phi_z(p_z)^\top c_r\right). \tag{3.78}$$

The approximation error is bounded by

$$\|f - f_{\mathrm{CP}}\|_2 \le \|\Psi\|_2\, \|w - w_{\mathrm{CP}}\|_F. \tag{3.79}$$

Hence, if the CP approximation is accurate in Frobenius norm, the induced SDF error is also controlled.

As a result, computational complexity drops from $\mathcal{O}(N^3)$ to $\mathcal{O}(R\,N)$, while the memory footprint reduces from N^3 to $3RN$.

Matrix form over sampled points. Let

$$A = [a_1\ \dots\ a_R], \quad B = [b_1\ \dots\ b_R], \quad C = [c_1\ \dots\ c_R], \quad \lambda = (\lambda_1, \dots, \lambda_R)^\top.$$

Given P sampled points $\{(x_\ell, y_\ell, z_\ell)\}_{\ell=1}^{P}$, and basis-evaluation matrices $\Phi_x, \Phi_y, \Phi_z \in \mathbb{R}^{N\times P}$, Eq. (3.78) becomes

$$f_{\text{CP}} = \Big((\Phi_x^\top A) \odot (\Phi_y^\top B) \odot (\Phi_z^\top C)\Big)\, \lambda, \tag{3.80}$$

where $\odot$ denotes the Hadamard (element-wise) product between matrices of equal size.

The gradient of the CP-approximated SDF can be derived analytically from CP-decomposition by differentiating the separable terms along each coordinate:

$$\nabla f(p) = \begin{bmatrix} \sum_{r=1}^{R} \lambda_r \, (\Phi_x'(p_x)^\top a_r)\, (\Phi_y(p_y)^\top b_r)\, (\Phi_z(p_z)^\top c_r) \\ \sum_{r=1}^{R} \lambda_r \, (\Phi_x(p_x)^\top a_r)\, (\Phi_y'(p_y)^\top b_r)\, (\Phi_z(p_z)^\top c_r) \\ \sum_{r=1}^{R} \lambda_r \, (\Phi_x(p_x)^\top a_r)\, (\Phi_y(p_y)^\top b_r)\, (\Phi_z'(p_z)^\top c_r) \end{bmatrix} \frac{1}{d_{\max} - d_{\min}}\,. \tag{3.81}$$

This expression yields the analytical gradient of the Bernstein field under CP decomposition, maintaining full differentiability and linear computational scaling with respect to the rank R.

Intuitive interpretation. Each rank-one component r can be interpreted as a *separable mode* capturing a dominant geometric pattern of the SDF. The vectors a_r, b_r, and c_r describe how this mode varies along each coordinate axis, while the scalar λ_r quantifies its relative importance. Hence, the CP model can be seen as learning a small number of "principal geometric modes" that reconstruct the entire 3D shape as a sum of separable interactions. This intuition connects the Bernstein–CP model to the concept of modal decomposition in structural mechanics or to principal component analysis (PCA) in statistics, but extended to third-order tensors.

Numerical and practical considerations. The proposed learning scheme estimates the Bernstein coefficients directly from SDF samples through a recursive regularized least-squares (RLS) update, which can be interpreted as an online variant of the Alternating Least Squares (ALS) minimization of the squared reconstruction error. This adaptive filtering process yields a smooth and noise-suppressed weight tensor w, representing the projection of the original SDF dataset onto the Bernstein polynomial space.

Once w is obtained, the tensor decomposition can be performed directly on it:

$$\min_{\{\lambda_r, a_r, b_r, c_r\}} \|w - \sum_{r=1}^{R} \lambda_r \, a_r \otimes b_r \otimes c_r\|_F^2. \tag{3.82}$$

This formulation is equivalent, in representational power, to applying a low-rank factorization on the original dataset, since w fully encodes the mapping between spatial coordinates and field values. However, it is numerically more robust: w lies in a compact, orthogonal polynomial basis, and the decomposition acts on smooth, denoised coefficients rather than raw and potentially irregular SDF samples. Consequently, the CP factors $\{a_r, b_r, c_r\}$ can be identified more reliably, and moderate ranks $R \in [5, 64]$ already achieve reconstruction errors below 10^{-2} with a 2–3$\times$ reduction in evaluation time.

Results. The results in Table 3.8 highlight the trade-off between accuracy and computational efficiency introduced by the CP decomposition of the Bernstein weight tensor. Lower ranks ($R \leq 10$) yield significant speedups (up to 2.4$\times$) at the cost of a moderate increase in approximation error, whereas higher ranks progressively recover the original SDF precision. The evaluation time remains almost constant due to GPU parallelization overhead, indicating that the complexity reduction mainly benefits memory usage and scalability to larger batches.

Table 3.8. Comparison between the baseline SDF evaluation and the CP-decomposed version for different tensor ranks R. Reported metrics include the mean and maximum reconstruction error of the distance field, and the corresponding speedup with respect to the full Bernstein model. 2000 random query points are evaluated per link.

Rank	Mean Error	Max Error	Speedup
5	3.9616×10^{-2}	1.2964×10^{-1}	2.24
10	4.1725×10^{-2}	8.0760×10^{-2}	2.41
16	5.6570×10^{-3}	2.0381×10^{-2}	1.98
32	7.5828×10^{-5}	3.4434×10^{-4}	1.84

At this stage, no automatic criterion for selecting the *optimal* CP rank has been implemented. The rank R is manually chosen as a trade-off between reconstruction accuracy and computational cost. Future work will address adaptive rank selection based on the tensor spectral decay or cross-validation on the SDF training set.

Although the current implementation applies the CP decomposition at the single-link level, the same principle naturally generalizes to the multi-link, multi-configuration RDF formulation of Sec. 3.3.9.

3.4 Robot Maps

Understanding how a robot can act within its workspace requires a unified description of its kinematic structure, geometric constraints, and interaction capabilities. Rather than treating these elements as independent, it is increasingly beneficial to represent them within a common spatial framework — the so-called capability maps [102]. Such maps encode where and how a robot can operate, capturing reachability[102, 61], manipulability[92], and collision information [98] across the workspace. They have proven fundamental for trajectory planning, base placement, and autonomous decision-making, especially in unstructured environments where adaptability and robustness are critical [49, 103].

The integration of detailed robot models, including link geometries, joint limits, and mesh-based representations, allows these maps to go beyond purely kinematic descriptions. By combining analytical kinematics with geometric reasoning, one can construct continuous representations of reach and dexterity while explicitly accounting for physical constraints. In this work, reachability maps are generated through large-scale inverse-kinematics sampling, from which manipulability indices are derived via Jacobian analysis. Collision regions are then identified using RDFs (discussed in Sec.3.3), which provide differentiable estimates of proximity between the robot and the environment.

In this chapter, inspired by the work of [61], we improve and extend existing methods for generating capability maps. In particular, we describe in detail the process of generating such maps, focusing on the computational aspects required to obtain an accurate and usable model via parallelized algorithms on GPU.

3.4.1 Workspace Discretization

The first step consists in delimiting the operational volume of the robot and subdividing it into *voxels*. Given a bounding box centered at the robot base, a three dimensional grid with resolution r is applied:

$$V = \{v_i \mid i = 1, \dots, N\}, \quad N = r^3$$

where each voxel v_i represents a cubic cell of side Δx.

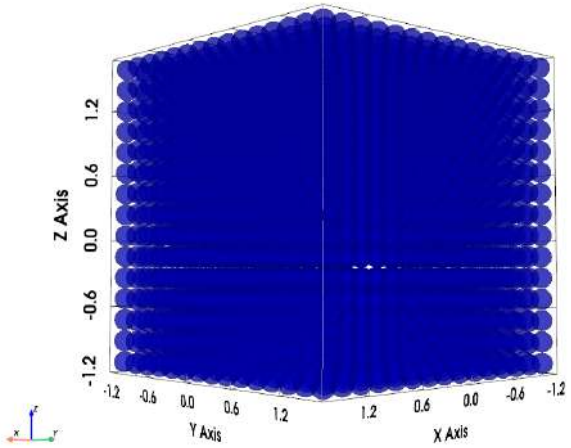


Figure 3.21. Voxelization of the workspace with internal spheres.

Orientation Sampling

As shown in Fig.3.21, for each voxel, a virtual sphere of radius equal to half the voxel size is placed at the voxel center. On the surface of this sphere, N_P points are generated in order to approximate a uniform coverage of possible orientations of the end-effector's z–axis.

A common strategy is to employ the *Fibonacci spiral*, which produces a quasi–uniform point distribution over the unit sphere, results are shown in Fig.3.22.

$$\begin{cases} \phi_k = \arccos\left(1 - \frac{2(k+0.5)}{N_P}\right), \\ \theta_k = \pi(1+\sqrt{5})\,k, \end{cases} \qquad k = 0, \ldots, N_P - 1. \tag{3.83}$$

Each pair (ϕ_k, θ_k) defines a point in spherical coordinates, which is then mapped to Cartesian coordinates (x_k, y_k, z_k). The resulting set of vectors corresponds to candidate directions of the local z–axis of the end-effector.

Since an orientation in $SO(3)$ is not uniquely determined by the z–axis direction alone, multiple in–plane rotations around this axis are applied. By discretizing the roll angle into N_R values, the method produces a set of candidate end-effector poses $p \in SE(3)$ for each voxel.

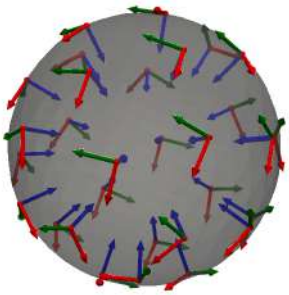


Figure 3.22. Spherical sampling of orientations using Fibonacci spiral.

3.4.2 Inverse Kinematics Problem

The orientations obtained from the sampling process, described in Sec.3.4.1, must be *shifted* to the voxel location described in Sec.3.4.1. That is, the sphere is centered at the voxel center c, and all sampled points are translated accordingly. Each voxel is thus associated with a discrete

set of candidate end-effector poses, obtained by combining the sampled orientations with the voxel center translation. Formally, the discretization yields a finite set of poses

$$\mathcal{P}_c = \{\, p_i = (R_i, t_i) \in SE(3) \mid R_i \in SO(3),\ t_i = c \,\}_{i=1}^{N_P N_R},$$

where N_P is the number of sampled z–axis directions and N_R the number of roll discretizations.

Given the set $\mathcal{P}_c$, the inverse kinematics problem consists in verifying, for each candidate pose $p_i = (R_i, t_i)$, whether there exists a joint configuration $q \in \mathbb{R}^n$ such that the forward kinematics satisfies

$$T(q) = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix}. \tag{3.84}$$

The solution of the *Inverse Kinematics* (IK) problem may yield zero, one or several valid configurations.

Batch IK Solving

Since the discretization procedure generates a large number of candidate poses, evaluating the inverse kinematics for each element of $\mathcal{P}_c$ becomes computationally demanding. To address this issue, we exploit the parallel computing capabilities of modern GPUs by processing poses in *batches*. Rather than evaluating the entire set $\mathcal{P}_c$ at once, a user–guided subset $\mathcal{P}_c^{(b)} \subseteq \mathcal{P}_c$ is selected and passed to the `inverse_batch` routine:

$$\mathcal{P}_c^{(b)} = \{\, p_{i_1}, p_{i_2}, \ldots, p_{i_B} \,\}, \quad B \ll N_P N_R,$$

where B is the batch size. Each pose p_{i_j} is mapped to its homogeneous transformation T_{i_j} and processed simultaneously on the GPU. For each pose p_i, the IK equations yield up to K candidate joint configurations $\{q_{i,k}\}_{k=1}^{K}$. The total number of candidates K depends on the manipulator's kinematic structure; for a 6–DOF arm, up to 8 solutions may exist due to multiple joint configurations leading to the same end-effector pose. The total number of batch solutions is therefore $B \cdot K$ and will be identified as $\mathcal{Q}_{\text{sols}}^{B} \in \mathbb{R}^{B \times K \times N_{\text{joints}}}$.

The inverse kinematics solver is based on the C++ code automatically generated by *IKFast*, which we translated into PyTorch to enable efficient tensor operations. Since, within a batch, not all candidate poses correspond to valid solutions, a logical mask $\mathcal{M}_i \in \{0,1\}^K$ is applied for batching computation such that

$$\mathcal{M}_{i,k} = \begin{cases} 1 & \text{if } q_{i,k} \text{ satisfies all feasibility constraints,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.85}$$

The feasibility constraints include:

$$\begin{aligned} &\text{Geometric validity: } |c_3| \le 1, \quad c_3 = \frac{\|p_{13}\|^2 - a_2^2 - a_3^2}{2a_2a_3}, \\ &\text{Numerical stability: } |f(q_{i,k})| > \varepsilon \quad \Rightarrow \quad \text{reject}, \\ &\text{Joint limits: } q_{\min} \le q_{i,k} \le q_{\max}. \end{aligned} \tag{3.86}$$

The final set of admissible configurations for pose p_i is therefore

$$\mathcal{Q}_i = \{\, q_{i,k} \mid \mathcal{M}_{i,k} = 1 \,\}. \forall i = 1, \ldots, N_P N_R.$$

A pose is said to be *reachable* if at least one valid configuration exists.

3.4.3 Maps

The discretization procedure previously described and the subsequent inverse kinematics evaluation provide, for each voxel, a set of sampled configurations $\mathcal{Q}_i$. Aggregating this information over the entire workspace allows the construction of *robot maps*, which encode structural properties of the manipulator at the joint level.

Reachability Map

For each sphere associated with a voxel, a **reachability index** D_k is defined:

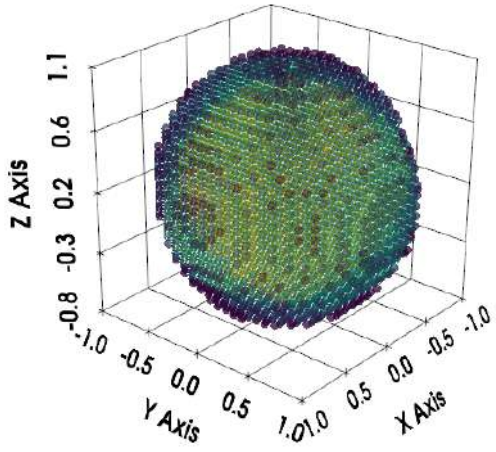
$$D_k = \frac{R_k}{N_P \cdot N_R} \cdot 100\% \tag{3.87}$$

where R_k is the number of reachable poses among those sampled, N_P is the number of poses sampled on the sphere and N_R is the number of roll discretizations. The index D_k is then stored in the map. Visualization is commonly performed using color coding, yielding a three–dimensional *heatmap* of the workspace. The **reachability map** assigns to each voxel the index D_k defined above, i.e. the ratio of reachable poses within the sampled set.

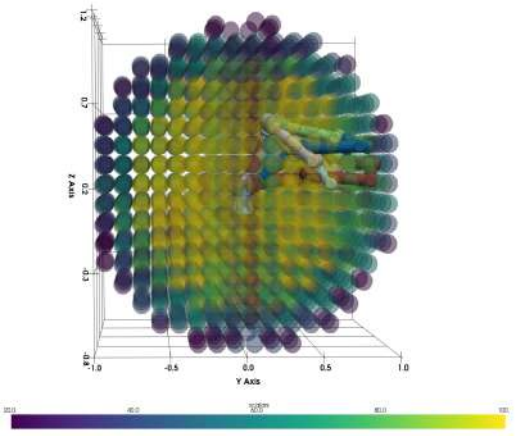
In addition to D_k, a *boolean reachability index* B_k can be defined:

$$B_k = \begin{cases} 1 & \text{if } \exists\, p_i \in \mathcal{P}_c : \rho(p_i) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.88}$$

Here, $B_k = 1$ marks voxels with at least one feasible pose, whereas $B_k = 0$ denotes unreachable voxels. Thus, D_k measures the density of reachable poses, while B_k provides a coarse but cheaper reachability descriptor.



(a) Complete reachability map of a 6–DOF manipulator.



(b) Cutted Reachability map of a 6–DOF manipulator. Here we can see more in detail the distribution of the reachability index in a local voxel.

Manipulability Map

Following the results presented in [94] and [93], we adopt the parallelized *Product of Exponentials* (POE) formulation to compute the performance index of the manipulator at each voxel. Given the screw axes $\{S_i\}_{i=1}^{n}$ and the home configuration $H \in SE(3)$ and given a batch $\mathcal{Q}_{\text{sols}}^{B}$ inside a reachable voxel rewritten in the form $\{\boldsymbol{q}^{(b)}\}_{b=1}^{B}$, the forward kinematics is computed as

$$T^{(b)} = \prod_{i=1}^{n} \exp([S_i] q_i^{(b)})\, H, \qquad b = 1, \ldots, B. \tag{3.89}$$

and the corresponding body Jacobian is

$$J_b^{(b)} = \mathrm{Ad}_{(T^{(b)})^{-1}} \left[S_1,\ \mathrm{Ad}_{e^{[S_1] q_1^{(b)}}} S_2,\ \ldots,\ \mathrm{Ad}_{\prod_{k=1}^{n-1} e^{[S_k] q_k^{(b)}}} S_n \right]. \tag{3.90}$$

For each batch element, the *Yoshikawa manipulability* index is computed as

$$m^{(b)} = \sqrt{\det\left(J_b^{(b)} (J_b^{(b)})^\top \right)}. \tag{3.91}$$

In analogy with the reachability map, composed by reachability indeces, the manipulability manipulability index is obtained by associating to each voxel v_k the average index over its feasible joint configurations $\mathcal{Q}_k$:

$$M_k = \frac{1}{|\mathcal{Q}_k|} \sum_{q^{(b)} \in \mathcal{Q}_k} m^{(b)}. \tag{3.92}$$

where $|\mathcal{Q}_k|$ is the cardinality of the valid solution in the k-th voxel.

The collection $\{M_k\}$ thus extends the reachability map into a *manipulability-map* representation of the workspace and is shown in Fig.3.24.

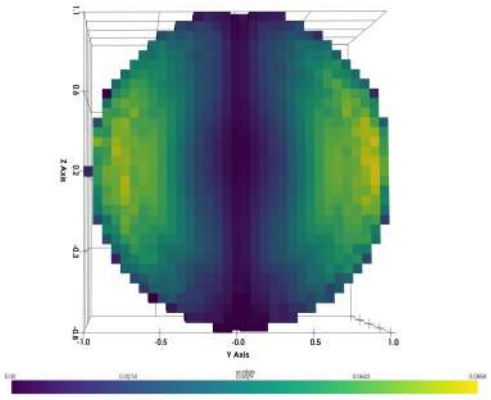
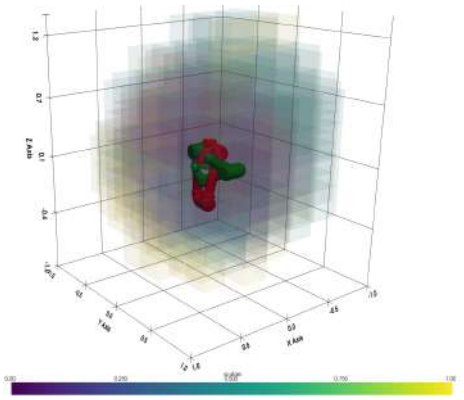


Figure 3.24. Manipulability map of a 6–DOF manipulator.

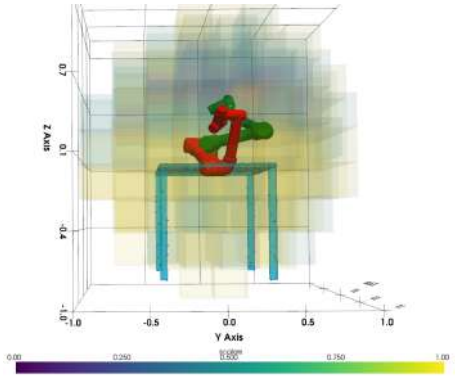
Collision Map

To account for environmental obstacles, a *collision map* is constructed by evaluating the proximity of the robot to surrounding objects using RDFs as described in Sec.3.3. For each joint configuration Q_k associated with voxel v_k, the minimum distance between the robot links and the environment is computed using the method descibed in Sec.3.3.9 that allows to evaluate multiple joint configurations in a single batch. To handle collisions, a threshold distance d_{thresh} is defined. A single configuration $\mathbf{q}_k$ is then classified as *in collision* if any query point (on the robot body or sampled in the workspace) satisfies

$$f_R(p_j, \mathbf{q}_k) < \epsilon. \tag{3.93}$$



(a) Self Collision map of a 6–DOF manipulator, it can be noticed that in a voxel some configurations are in collision while others are not.



(b) Environment Collision map of a 6–DOF manipulator in presence of an obstacle. Some configurations are in collision while others are not.

Figure 3.25. Collision maps representing the fraction of collision-free configurations per voxel.

Then, a *collision index* C_k for voxel v_k is defined as the ratio of collision-free configurations to the total number of configurations evaluated:

$$C_k = \frac{1}{|\mathcal{Q}_k|} \sum_{\mathbf{q}_k \in \mathcal{Q}_k} \begin{cases} 1 & \text{if } \forall p_j : f_R(p_j, \mathbf{q}_k) \geq \epsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{3.94}$$

This index C_k thus quantifies the safety of the robot's operation within voxel v_k, with higher values indicating a lower likelihood of collision.

As can be seen in Fig.3.25, the collision maps handles both self-collisions (Fig.3.25a) and collisions with external obstacles (Fig.3.25b), providing a representation of collision risks throughout the robot's workspace.

3.4.4 Database Architecture

The capability analysis described in the previous sections produces a large amount of structured data: for each robot configuration, thousands of sampled poses, inverse kinematics solutions, and associated performance indices are generated. To manage this information in a consistent, queryable, and reusable way, a dedicated relational database has been designed. Its structure, illustrated in Fig. 3.26, organizes the data hierarchically, reflecting the geometric and kinematic decomposition of the robot workspace.

At the highest level, the **Robot** entity defines a specific manipulator, characterized by its degrees of freedom, kinematic parameters, and workspace boundaries. Each robot is associated with a unique *reachability map*, which discretizes the workspace into cubic cells called **voxels**. Each voxel represents a local region of the workspace and acts as a container for all poses that can be sampled and evaluated within that region.

Around the center of every voxel, a set of **Points** is generated according to the spherical sampling procedure described in Sec. 3.4.1. Each point corresponds to a specific pose of the end-effector, defined by position and orientation, and carries a binary reachability flag indicating whether a valid inverse kinematics solution exists. This geometric layer, composed of robots, voxels, and points, provides a normalized and reusable representation of the workspace, independent of the particular metrics or experiments later performed on it.

The next level of abstraction enriches each point with kinematic evaluations. For every feasible configuration obtained from the inverse kinematics solver, a corresponding entry is stored

in the **IndexMap** table. This table encapsulates per-solution performance metrics such as manipulability, reachability, and collision state. Each entry in **IndexMap** is then linked one-to-one to a row in the **JointValues** table, which stores the actual joint configuration associated with that evaluation. This separation between geometry and evaluation is intentional: it allows the same geometric dataset to be reanalyzed or reinterpreted without recomputing the entire map. For example, if new metrics or environmental conditions are introduced, only the entries of **IndexMap** need to be updated, while the underlying voxel and point structure remains unchanged.

Such a design transforms the database from a static storage system into a dynamic analytical tool. By exploiting SQL queries and relational integrity, one can efficiently filter, aggregate, and compare results across different robots, configurations, or experimental conditions. The model also supports differential analyses, for instance, comparing manipulability distributions between two manipulators within the same workspace discretization, or identifying regions where collision risk dominates reachability. This relational structure ensures data consistency through foreign key constraints and enables selective loading of relevant subsets into memory for GPU-based post-processing.

In summary, the database provides a scalable and logically consistent foundation for representing robotic capability maps. It bridges the geometric domain of workspace discretization with the analytical domain of kinematic and dynamic evaluation, enabling reproducible and extensible studies on robot reachability, manipulability, and collision behavior.

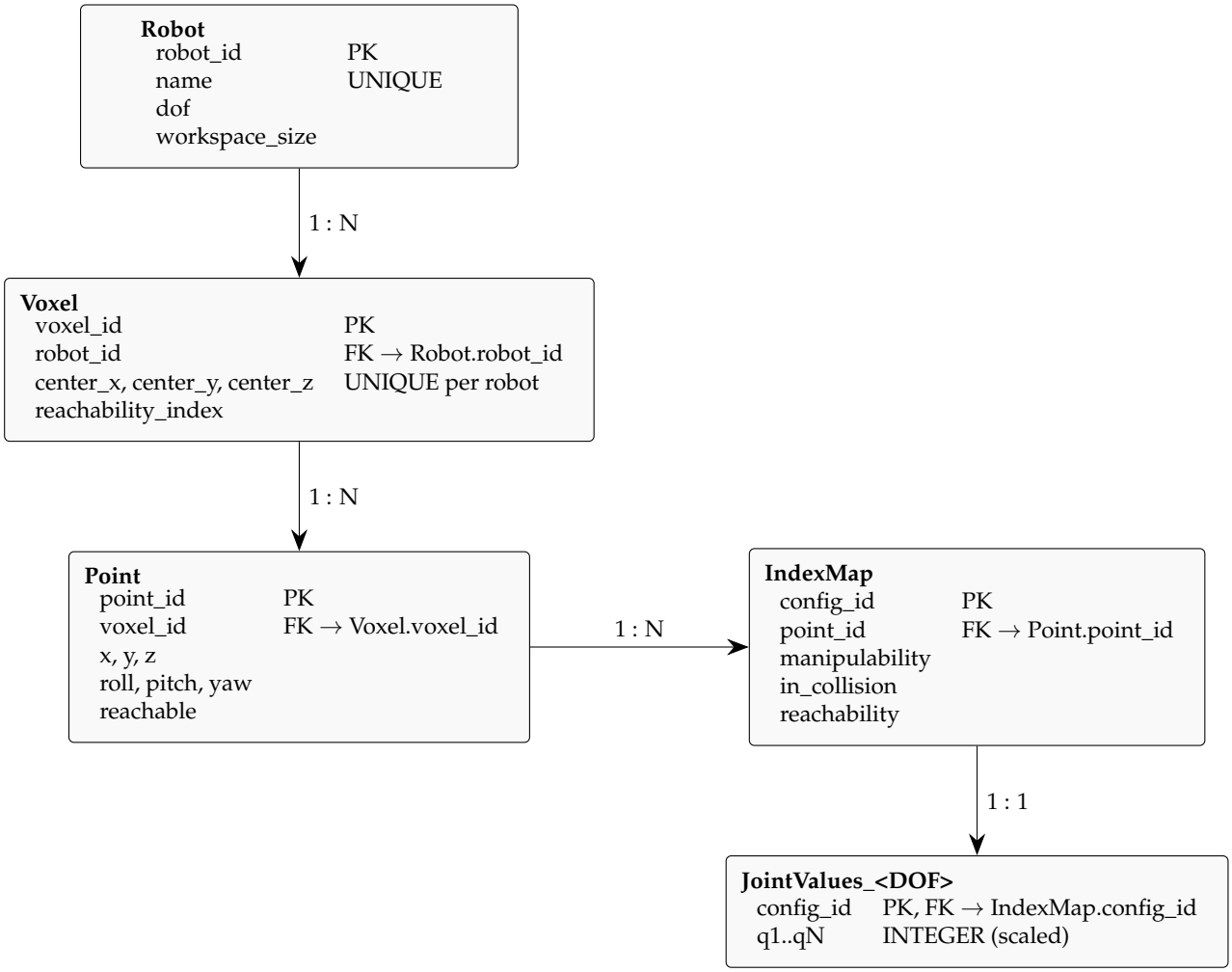


Figure 3.26. Updated database schema with Robot, Voxel, Point, IndexMap, and JointValues tables.

Table 3.9. Benchmark computation times for the UR5 manipulator (from [61]).

Method	Robot	Poses [$\times 10^5$]	Computation time [min]		
			Reach	Manipulability	Collision
CoReMap (proposed)	UR5	300	0.147	1.31	30.57
Diankov [18]	UR5	123.513	371.38	✗	✗
Reuleaux [61]	UR5	7.636	143.27	✗	✗
Zacharias [102]	UR5	38.18	413.23	✗	✗

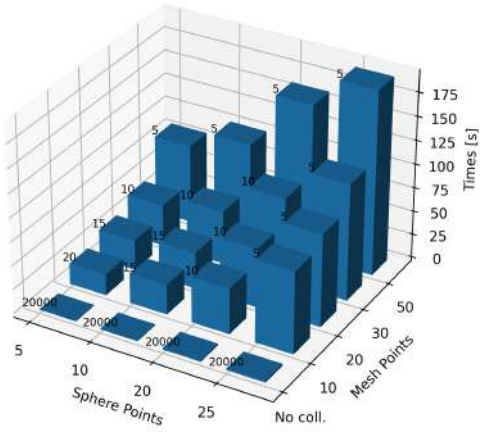


Figure 3.27. Batching comparison computation time. Higher sampling of mesh and sphere points increases the memory occupied and reduce the number of the batches. Workspace is discretized in 20^3 voxels

3.4.5 Results and Considerations

Benchmark Computation Times

To evaluate the performance of the proposed GPU–accelerated algorithms for generating reachability, manipulability, and collision maps, a series of benchmarks were conducted using the **UR5** manipulator as reference platform.

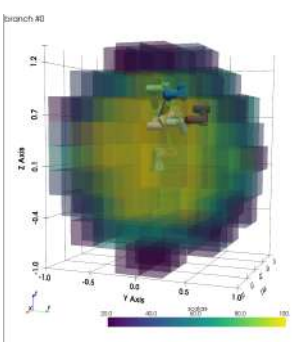
In this stage of development, the analysis focuses exclusively on the computational aspects of the proposed pipeline, without any physical execution on real robotic systems. The goal is to quantify the performance of the GPU–based implementation, validate its correctness against analytical expectations, and identify computational bottlenecks prior to real-world deployment.

For comparison, Table 3.9 reports computation times obtained from existing literature, notably the Reuleaux framework [61], which processes approximately 763,600 poses for the UR5 robot (corresponding to about 5,017 sampled spheres and ∼152 poses per sphere). In our current implementation, this configuration corresponds to a voxel resolution of 30 and 28 sampled points per sphere, evaluated in batches of 10,000 poses on the GPU.

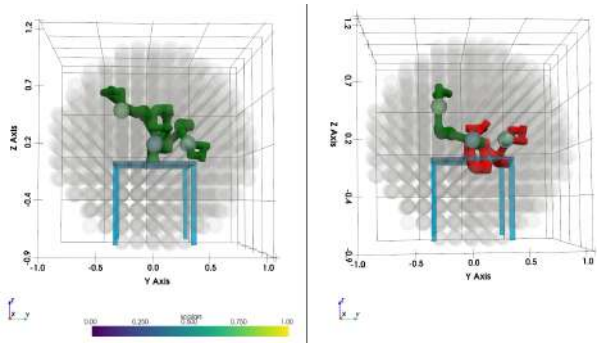
In our measurements, the reachability computation required approximately **8.8 seconds** (0.147 minutes) per map, while the manipulability evaluation took about **78.6 seconds** (1.31 minutes) using a batch size of 10,000. These results demonstrate a substantial reduction in computation time compared to CPU–based implementations reported in prior works, confirming the efficiency and scalability of the proposed GPU–accelerated approach.

Batching limits in collision computations

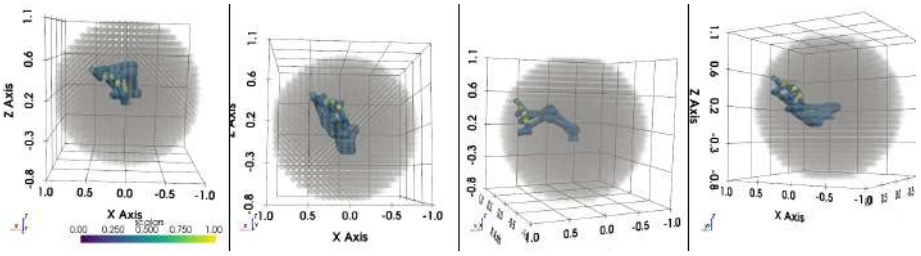
As already introduced in Sec.3.3.10 collision-aware reachability maps require evaluating a very large number of candidate configurations per voxel, especially when dense sampling is used.



(a) Reachability map by configuration



(b) Collisions per configuration



(c) Reach indices per configuration

Figure 3.28. Workspace analysis summary.

As illustrated in Fig. 3.27, the batching behavior changes significantly depending on whether collisions are considered.

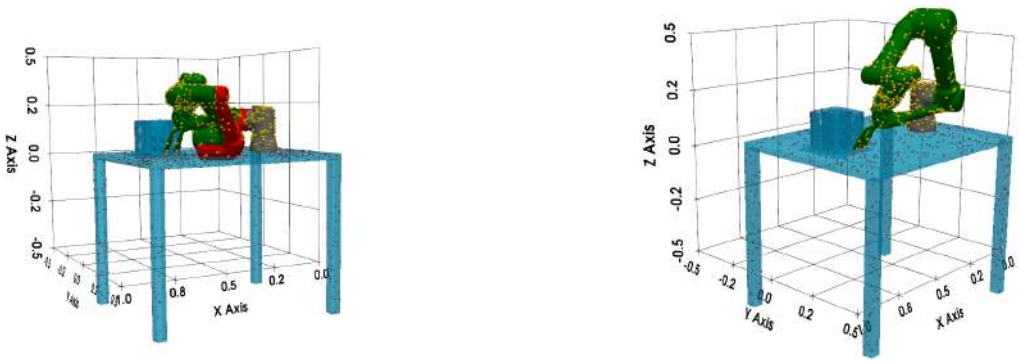
Without collision checking, the GPU can process large batches efficiently, resulting in many batches but very short computation times. When collisions are introduced, however, every configuration must be tested against environment and self-collision constraints, which reduces the feasible batch size (to avoid GPU OOM) and increases computation time.

Mathematically speaking, to check collisions each voxel requires up to $N_{\text{sphere}} \times N_{\text{solutions}} \times N_{\text{links}} \times N_{\text{coll}}$ collision tests, which grow rapidly with sampling density.

Usage

At the current stage, the reachability maps generated through large–scale inverse kinematics sampling should be interpreted as a *statistical and computational distribution* that encodes the local density of reachable configurations within the workspace. Each voxel index represents how "good" a region is in terms of feasible robot poses and can thus be used to guide path–planning or workspace analysis. However, this metric provides a static and averaged view of the robot's reachability, and its real–time applicability remains limited. Indeed, as visible in Fig. 3.28a, a voxel may exhibit a high reachability index and still include individual configurations that are not actually feasible or may result in collisions, but this information is not captured in the averaged index. For this reason, nowadays reachability maps are best suited for offline analyses, such as workspace exploration, base placement, or initial path planning. In our context, saving the branches of IK solutions allows to retrieve the actual configurations associated with each voxel, enabling more detailed analyses when needed. Indded if neeed, after a statistical approach in Fig.3.28 the user can go into detail separating solutions per configuration, allowing to retrieve which configuration is reachable Fig. 3.28c and/or which one is in collision in Fig.3.28b and for sure knowing the manipulability index associated with each configuration.

To provide an intuitive example, we emulate a welding application and compare two robot postures, as shown in Fig. 3.29: one that leads to collision (Fig. 3.29a) and one that remains collision-free during task execution (Fig. 3.29b).



(a) Welding configuration in collision (to avoid).

(b) Collision-free welding configuration (desired).

Figure 3.29. Comparison of two welding postures: the robot must maintain the collision-free configuration while executing the welding task.

Chapter 4

Task Priority

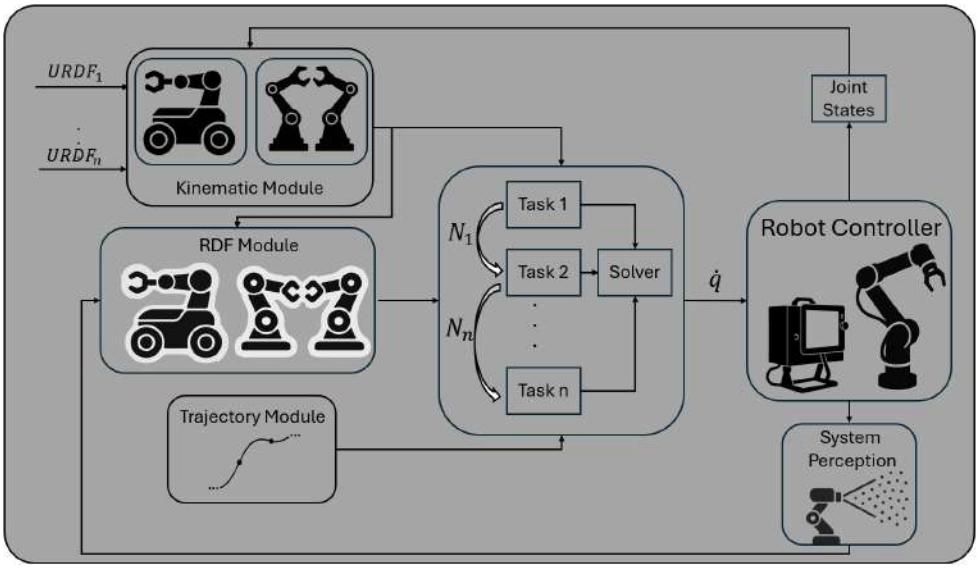


Figure 4.1. Overview of the Task-Priority control framework for multi-robot systems.

Modern robotic systems increasingly combine heterogeneous embodiments, from fixed-base manipulators to mobile and dual-arm platforms, requiring control architectures capable of managing different kinematic structures under a unified and adaptive formulation. Although several frameworks exist for robot control, most either focus on motion planning and trajectory execution (e.g., MoveIt [67]) or rely on static task hierarchies that do not react to intrinsic robot limitations such as singularities, joint limits, or task conflicts. As a consequence, their applicability to real-time, heterogeneous, or hybrid systems remains limited.

The library developed in this work provides a real-time Task-Priority (TP) control framework that explicitly addresses these limitations while offering a unified abstraction for mobile bases and manipulators within a single control layer and a common mathematical representation. Multiple tasks can be organized hierarchically, ensuring that high-priority objectives are preserved while lower-priority ones adapt compatibly via null-space projection activation. The framework reacts autonomously to configuration-dependent events (e.g., approaching singularities or constraint violations), smoothly reconfiguring task priorities without discontinuities.

Designed to be modular, lightweight, and platform-agnostic, the library can interface with any robotic system that provides joint position and velocity feedback. While it is particularly suited for redundant robots, where the null-space can be exploited to optimize secondary objectives, it also supports non-redundant manipulators (e.g., 6-DoF arms), with the inherent limitation of task locking due to the absence of redundancy. This latter aspect, however, lies beyond the scope of the present thesis.

In summary, the proposed TP library constitutes a real-time, adaptive, and unified control framework for both mobile and manipulator robots, bridging the gap between theoretical task-priority control and its practical deployment on complex robotic systems.

4.1 Theoretical Introduction

We now illustrate step by step how the control formulation evolves from the basic differential kinematics to the full Task-Priority framework with smooth task activation in the null-space.

4.1.1 Velocity tracking and regulation

Let n be the number of degrees of freedom (DoF) of the manipulator, and let m be the dimension of the task space, such that $m \leq n$. Let $q = [q_1, q_2, \ldots, q_n]^\top \in \mathbb{R}^n$ be the joint configuration vector and $x \in \mathbb{R}^m$ the task-space variable. The Jacobian $J(q) \in \mathbb{R}^{m \times n}$ describes the local mapping between joint and task velocities:

$$\dot{x}(t) = J(q)\dot{q}(t). \tag{4.1}$$

The control objective is to compute the joint velocity $\dot{q}$ that best reproduces a desired task velocity $\dot{x}_d$. This can be formulated as a least-squares problem:

$$\min_{\dot{q}} \|\dot{x}_d - J(q)\dot{q}\|^2, \tag{4.2}$$

whose minimum-norm solution is obtained through the Moore–Penrose pseudoinverse:

$$\dot{q} = J^\dagger(q)\dot{x}_d. \tag{4.3}$$

To regulate the task state x towards a desired value x^*, the velocity reference can be augmented with a feedback term based on the task error $\widetilde{x} = x^* - x(t)$:

$$\dot{q}^* = J^\dagger(q)\left(\dot{x}_d + K\,\widetilde{x}\right), \tag{4.4}$$

where $K \in \mathbb{R}^{m \times m}$ is a positive-definite gain matrix that ensures exponential convergence of the task when it is kinematically feasible.

4.1.2 Redundancy and secondary tasks.

When the robot has more joints than strictly needed for the task $m < n$, the manipulator is redundant and admits infinitely many joint velocity solutions.

Formally, since $\operatorname{rank}(J) \leq m$, the null space of J has dimension $n - \operatorname{rank}(J) \geq n - m$. All joint velocities achieving the same task velocity live on an affine manifold: In this case, the general solution can be written as in Eq. (4.5):

$$\dot{q}^* = J^\dagger(q)\left(\dot{x}_d + K\widetilde{x}\right) + Nz, \tag{4.5}$$

where $N = I - J^\dagger J$ is the null-space projector, and z can be used to encode secondary task objectives. N is the orthogonal projector onto $\ker(J)$, and $z \in \mathbb{R}^n$ parametrizes the *secondary motion*. Choosing z allows us to encode secondary objectives (e.g., joint-limit avoidance) *without* disturbing the primary task.

4.1.3 Task Priority.

The core idea of the Task-Priority (TP) framework is to recursively project lower-priority tasks in the null space of higher-priority ones, thereby avoiding interference. Formally, the minimization problem in Eq. (4.2) becomes:

$$\min_{\dot{q}} \|A\left(\dot{x}_d + K\,\widetilde{x} - J(q)\dot{q}(t)\right)\|^2, \tag{4.6}$$

whose solution is:

$$\dot{q}^* = (AJ)^{\dagger} A\left(\dot{x}_d + K\,\widetilde{x}\right) + Nz, \tag{4.7}$$

with $N = I - (AJ)^{\dagger}AJ$ and $A \in \mathbb{R}^{m\times m}$ being a diagonal activation matrix.

Task activation

Tasks may need to *fade in/out* depending on context (e.g., contact phases, safety margins), while ensuring that the robot remains well-behaved near singularities. To achieve this, each task k is associated with a diagonal *activation matrix*

$$A_k(s_k) = \mathrm{diag}\big(s_k\big) \in [0,1]^{m_k\times m_k},$$

whose diagonal entries $s_k = [\,s_{k,1},\ldots,s_{k,m_k}\,]^{\top}$ define the continuous activation level of the task components. These activations weight the contribution of each constraint to the overall control law, enabling smooth transitions between active and inactive phases.

Equality and inequality tasks. Both equality (ET) and inequality (IT) tasks can be expressed within this activation framework.

For **equality tasks (ET)**, the task must be always satisfied, which corresponds to full activation:

$$A_k = I_{m_k}, \text{if active} \qquad A_k = 0_{m_k}, \text{if inactive}$$

For **inequality tasks (IT)**, activation depends on the constraint margin:

$$A_k = \mathrm{diag}\big(s_k(x)\big)$$

where each scalar activation $s_{k,i}(x) \in [0,1]$ encodes whether the corresponding inequality is active ($s_{k,i} \approx 1$) or satisfied ($s_{k,i} \approx 0$).

Smooth activation law for inequality tasks. Each activation component $s_{k,i}(x)$ is computed via a cosine–sigmoid function:

$$s(x, x_{\text{th}}^-, x_{\text{th}}^+) = \begin{cases} 1, & \text{if } x \le x_{\text{th}}^-, \\ \dfrac{\cos\left(\dfrac{(x - x_{\text{th}}^-)\pi}{x_{\text{th}}^+ - x_{\text{th}}^-}\right) + 1}{2}, & \text{if } x_{\text{th}}^- \le x \le x_{\text{th}}^+, \\ 0, & \text{if } x \ge x_{\text{th}}^+, \end{cases} \tag{4.8}$$

where x_{th}^- and x_{th}^+ denote lower and upper activation thresholds, respectively. This function is C^1-continuous and ensures a smooth fading behavior between fully active and inactive regimes.

Finally, to maintain differentiability and numerical stability during task activation and deactivation, we adopt the smooth projection framework proposed in [86].

4.2 Unified Geometric Jacobians for Multi-Robot Systems

Before introducing the multi-robot Jacobian, this chapter formalizes the kinematics of common robot subsystems and shows how to compose them into a single, global geometric Jacobian suitable for task-priority control. In our framework, each robot is represented as one or more repeated *base–arm–gripper* compositions, or by a subset thereof, while always preserving the same ordering of subsystems. This convention ensures that the global Jacobian in Eq. (4.9) can be constructed systematically and applied uniformly across heterogeneous robotic platforms.

Let the task twist be $\dot{x} \in \mathbb{R}^6$ (linear and angular velocity of a chosen task frame in the world). For a single robot assembled from a planar base, a serial arm, and optionally a gripper, we stack the subsystems into a *global* geometric Jacobian:

$$\dot{x} = \underbrace{\begin{bmatrix} J_{\text{base}} & J_{\text{arm}} & J_{\text{grip}} \end{bmatrix}}_{J_{\text{global}} \in \mathbb{R}^{6\times n}} \begin{bmatrix} \dot{q}_{\text{base}} \\ \dot{q}_{\text{arm}} \\ \dot{q}_{\text{grip}} \end{bmatrix}. \tag{4.9}$$

All Jacobian blocks are expressed in the *same* spatial frame (world) and at the *same* task point.

When extending to a N multi-robot system composed of mobile bases and/or serial arms, and/or grippers, the same principle applies at a global scale. Let $\dot{x} \in \mathbb{R}^{6N}$ denote the stacked task twist, concatenating the spatial velocities of all task frames, and let $\dot{q}$ the vector that collect all generalized velocities of the system. The global kinematic relation becomes

$$\dot{x} = \underbrace{\text{blkdiag}\big(J_{\text{global},1},\, J_{\text{global},2},\, \ldots,\, J_{\text{global},N}\big)}_{J_{\text{multi}} \in \mathbb{R}^{6N\times n_{\text{tot}}}} \dot{q}, \tag{4.10}$$

where each local Jacobian block $J_{\text{global},i}$ corresponds to the i-th robot and maps its internal velocities to the spatial twist of its task frame, both expressed in the world reference frame. This block-diagonal structure naturally decouples the differential kinematics of the individual robots, while with proper adjustment allowing the subsequent task-priority control layers to introduce coupling terms.

In compact form, the same model can be written as

$$\dot{x} = J_{\text{multi}}(q)\, \dot{q}, \tag{4.11}$$

with J_{multi} implicitly including all active subsystems (bases, manipulators, and grippers) and serving as the unified interface for the higher-level task projector and control optimization layers.

The gripper is still not present in the current implementation, but the formulation is included for completeness.

4.2.1 Base Jacobians: Differential vs. Omnidirectional

We model the mobile base as a planar system commanded by body-frame velocities. The generic input is

$$u_b = \begin{bmatrix} v_x^b \\ v_y^b \\ \omega \end{bmatrix},$$

where v_x^b and v_y^b are linear velocities along the base-frame x and y axes, and ω is the yaw rate.

- For a *differential-drive* base, the nonholonomic constraint imposes $v_y^b = 0$. Hence the control input is

$$u_{\text{diff}} = \begin{bmatrix} v_x^b \\ \omega \end{bmatrix}.$$

- For an *omnidirectional* base, lateral velocity is allowed, so the full input vector is

$$u_{\text{omni}} = \begin{bmatrix} v_x^b \\ v_y^b \\ \omega \end{bmatrix}.$$

Mobile Jacobians

Let θ be the base yaw, $R_{wb}(\theta)$ the world-from-base rotation about z, and let $r_b \in \mathbb{R}^3$ be the displacement from the base origin to the task point *expressed in the base frame*. Equivalently, $r_w = R_{wb}\, r_b$ is the same displacement expressed in the world frame. The 6×3 Jacobian block that maps the body-frame input $u_b = \begin{bmatrix} v_x^b & v_y^b & \omega \end{bmatrix}^\top$ to the spatial twist of the task point (expressed in world) is

$$J_{\text{base}}^{\text{cart}}(\theta, r_b) = \begin{bmatrix} R_{wb}\, e_x & R_{wb}\, e_y & R_{wb}\big(-[r_b]_\times E_z\big) \\ 0 & 0 & E_z \end{bmatrix}, \tag{4.12}$$

with $e_x = [1,0,0]^\top$, $e_y = [0,1,0]^\top$, $E_z = [0,0,1]^\top$, and $[\,\cdot\,]_\times$ the skew-symmetric operator. Equivalently, using r_w:

$$J_{\text{base}}^{\text{cart}}(\theta, r_w) = \begin{bmatrix} R_{wb}\, e_x & R_{wb}\, e_y & -[r_w]_\times E_z \\ 0 & 0 & E_z \end{bmatrix}. \tag{4.13}$$

In both forms, the third column encodes the lever arm: $v_{\text{lin}}^{(\omega)} = \omega\,(E_z \times r) = -\omega\,[r]_\times E_z$.

The task-point twist is then

$$\dot{x} \;=\; J_{\text{base}}^{\text{cart}}(\cdot)\; u_b, \qquad \dot{x} \in \mathbb{R}^6 \text{ (linear \& angular in world).} \tag{4.14}$$

Differential-drive. Only v_x^b and ω are available:

$$u_{\text{diff}} = \begin{bmatrix} v_x^b \\ \omega \end{bmatrix}, \qquad \dot{x} = J_{\text{diff}}^{\text{cart}}(\theta, r_b)\, u_{\text{diff}}, \tag{4.15}$$

taking inspiration from [16], for a mobile manipulator the Jacobian is obtained by selecting columns 1 and 3 of Eq. (4.12) and then stacking it with the manipulator Jacobian if inversion is applied:

$$J_{\text{diff}}^{\text{cart}}(\theta, r_b) = \begin{bmatrix} R_{wb}\, e_x & R_{wb}\big(-[r_b]_\times E_z\big) \\ 0 & E_z \end{bmatrix} \in \mathbb{R}^{6\times 2}. \tag{4.16}$$

Omnidirectional. All planar body velocities are available:

$$u_{\text{omni}} = \begin{bmatrix} v_x^b \\ v_y^b \\ \omega \end{bmatrix}, \qquad \dot{x} = J_{\text{omni}}^{\text{cart}}(\theta, r_b)\, u_{\text{omni}}, \tag{4.17}$$

with

$$J_{\text{omni}}^{\text{cart}}(\theta, r_b) = \begin{bmatrix} R_{wb}\, e_x & R_{wb}\, e_y & R_{wb}\big(-[r_b]_\times E_z\big) \\ 0 & 0 & E_z \end{bmatrix}. \tag{4.18}$$

Remarks. (i) All blocks are expressed at the *same* task point and in the *world* frame. (ii) E_z in the angular part is sufficient (for planar motion $R_{wb}E_z = E_z$). (iii) This mapping is kinematic; feasibility of a 6-D task with a planar base requires *redundancy* (e.g., base+arm+gripper with overall rank $\geq$ 6). In the differential-drive case the base alone cannot span lateral velocity instantaneously ($v_y^b = 0$), so task realization relies on manipulator redundancy.

4.2.2 Serial manipulator (geometric Jacobian)

For the arm we used the standard geometric Jacobian formulation [85]. Consider an n-DOF serial chain with joint variables q. Let ${}^w p_i$ be the position of joint i and ${}^w z_i$ the unit axis, both in world. The *geometric* Jacobian $J_{\text{arm}}(q) \in \mathbb{R}^{6\times n}$ has columns

$$J_{\text{arm},i} = \begin{cases} \begin{bmatrix} {}^w z_i \times ({}^w p_e - {}^w p_i) \\ {}^w z_i \end{bmatrix}, & \text{if joint } i \text{ is revolute,} \\ \begin{bmatrix} {}^w z_i \\ 0 \end{bmatrix}, & \text{if joint } i \text{ is prismatic.} \end{cases} \tag{4.19}$$

This directly maps joint rates to the end-effector spatial twist: $\dot{x} = J_{\text{arm}}(q)\,\dot{q}$.

4.3 Trajectory Generation

The need for a unified framework arises from the task–priority control architecture, where the controller requires consistent reference signals regardless of the chosen trajectory type. For this reason, a dedicated trajectory library has been designed. Its main goal is to abstract away the differences between interpolation methods (polynomial, cubic spline, Bézier, trapezoidal, circular, quaternion-based, etc.) and provide a common interface.

In particular, given a selected trajectory type, the framework automatically returns both the reference position and the corresponding velocity, ensuring that these signals are always available and mutually consistent. This uniform structure greatly simplifies the integration within task–priority schemes, as the controller can rely on the same data format independently of the underlying trajectory generator.

4.3.1 Position Trajectories

Linear Interpolation.

The simplest trajectory between two points is defined by a linear law:

$$p(t) = p_0 + \frac{t - t_0}{t_f - t_0}\left(p_f - p_0\right), \tag{4.20}$$

with constant velocity

$$\dot{p}(t) = \frac{p_f - p_0}{t_f - t_0}. \tag{4.21}$$

Scaled Trajectory to Cartesian Trajectory

The *scaled trajectory* formalism allows the definition of motion profiles in a normalized scalar domain, where a single parameter $s(t) \in [0, 1]$ encodes the progression along a path. This formulation separates the time law from the geometric path, enabling the reuse of the same timing profile for different Cartesian trajectories.

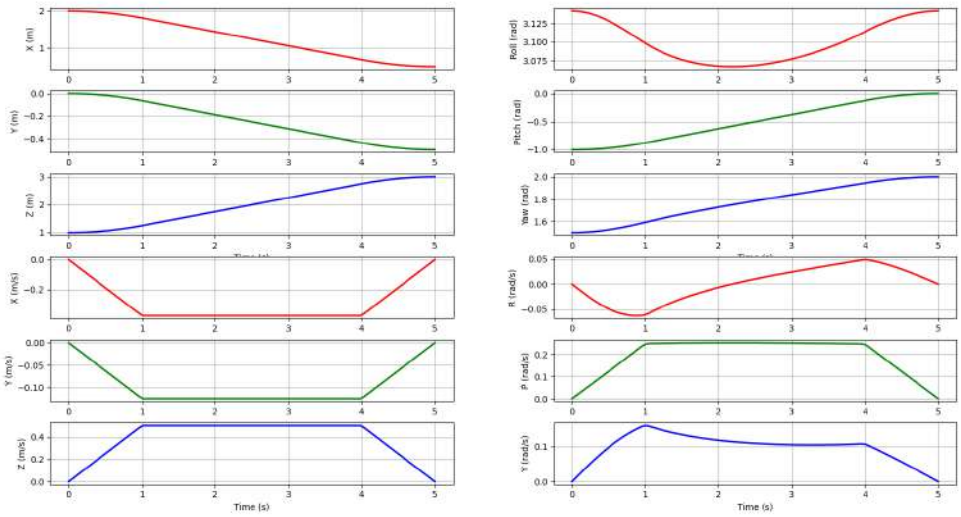


Figure 4.2. Example of a trapezoidal trajectory position and orientations(top) and linear and angular velocity (bottom).

Scaled Trapezoidal Velocity Profile. The motion is divided into acceleration, constant velocity, and deceleration phases. The velocity law is

$$\dot{s}(t) = \begin{cases} at, & 0 \le t < t_a, \\ v_{\max}, & t_a \le t < t_f - t_a, \\ a(t_f - t), & t_f - t_a \le t \le t_f, \end{cases} \tag{4.22}$$

and the position parameter $s(t)$ is obtained by integration:

$$s(t) = \int_0^t \dot{s}(\tau)\, d\tau. \tag{4.23}$$

The parameters a and t_a are chosen to guarantee a continuous velocity profile and that $s(t_f) = 1$ at the end of motion:

$$a = \frac{v_{\max}}{t_a}, \qquad 1 = v_{\max}(t_f - t_a). \tag{4.24}$$

Scaled Asymmetric Trapezoidal Velocity Profile. Let $s(t) \in [0, 1]$ parametrize motion along a fixed path (e.g., $q(t) = q_0 + \Delta q\, s(t)$, $\Delta q := q_f - q_0$). With possibly different acceleration/deceleration times $t_a \neq t_d$, define

$$a_\uparrow := \frac{v_{\max}}{t_a}, \qquad a_\downarrow := \frac{v_{\max}}{t_d}.$$

The velocity/acceleration laws are

$$\dot{s}(t) = \begin{cases} \dfrac{v_{\max}}{t_a}\, t, & 0 \le t < t_a, \\ v_{\max}, & t_a \le t < t_f - t_d, \\ \dfrac{v_{\max}}{t_d}\, (t_f - t), & t_f - t_d \le t \le t_f, \end{cases} \quad \ddot{s}(t) = \begin{cases} a_\uparrow, & 0 \le t < t_a, \\ 0, & t_a \le t < t_f - t_d, \\ -a_\downarrow, & t_f - t_d \le t \le t_f. \end{cases} \tag{4.25}$$

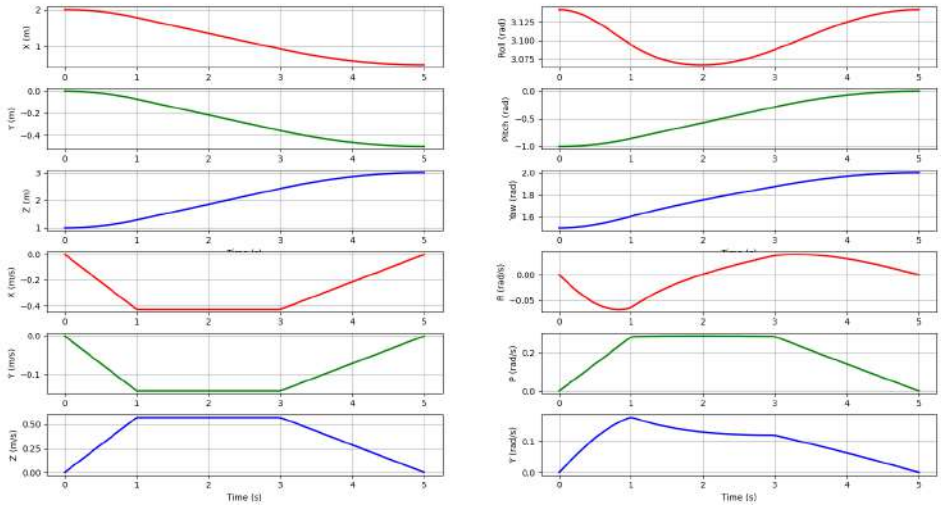


Figure 4.3. Example of asymmetric trapezoidal trajectory position and orientations(top) and linear and angular velocity (bottom).

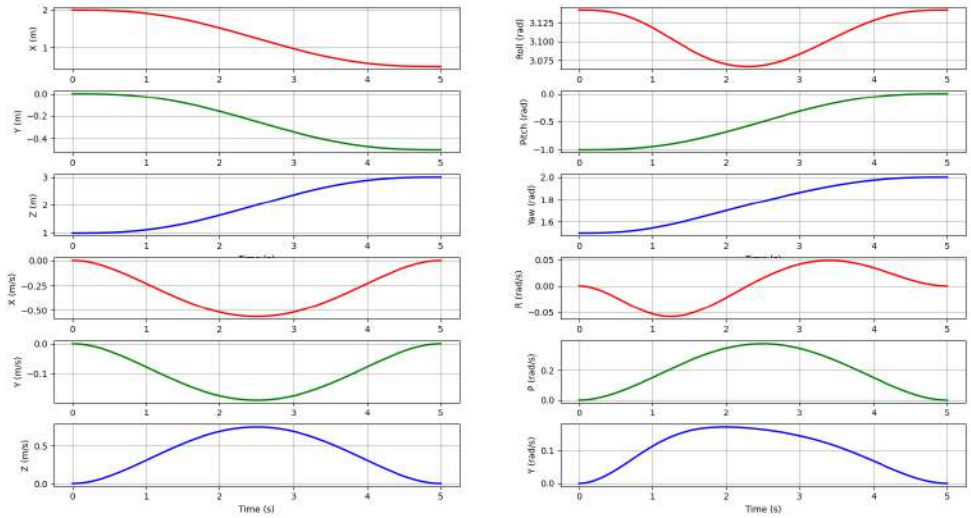


Figure 4.4. Example of a fifth-order polynomial trajectory position and orientations(top) and linear and angular velocity (bottom).

Integrating with $s(0) = 0$ gives

$$s(t) = \begin{cases} \dfrac{v_{\max}}{2t_a}\, t^2, & 0 \le t < t_a, \\ \dfrac{v_{\max}}{2} t_a + v_{\max}(t - t_a), & t_a \le t < t_f - t_d, \\ 1 - \dfrac{v_{\max}}{2t_d}(t_f - t)^2, & t_f - t_d \le t \le t_f. \end{cases} \tag{4.26}$$

Closure and feasibility. Area under $\dot{s}$ must be 1:

$$\boxed{1 = v_{\max}\left(t_f - \frac{t_a + t_d}{2}\right)}, \qquad t_c := t_f - t_a - t_d \ge 0 \text{ (constant-velocity segment exists).} \tag{4.27}$$

The symmetric case $t_a = t_d$ is the special case $1 = v_{\max}(t_f - t_a)$.

Scaled Fifth-Order Polynomial Profile. To ensure smoothness and continuity up to the acceleration level, a fifth-order polynomial time law is used:

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5, \tag{4.28}$$

with velocity

$$\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4. \tag{4.29}$$

The coefficients are obtained by imposing the six boundary conditions

$$\begin{aligned} s(t_0) &= 0, \quad & \dot{s}(t_0) &= 0, \quad & \ddot{s}(t_0) &= 0, \\ s(t_f) &= 1, \quad & \dot{s}(t_f) &= 0, \quad & \ddot{s}(t_f) &= 0, \end{aligned} \tag{4.30}$$

which ensure C^2 continuity and smooth start–stop behavior.

From Scaled to Cartesian Trajectories. Once a smooth scalar profile $s(t)$ is defined, it can be mapped to any Cartesian motion between two points p_0 and p_f as

$$p(t) = p_0 + s(t)\,(p_f - p_0), \tag{4.31}$$

with corresponding velocity

$$\dot{p}(t) = \dot{s}(t)\,(p_f - p_0). \tag{4.32}$$

In this way, the geometric path remains linear in space, while the temporal evolution is dictated by the chosen scaling law. This abstraction allows the same time-scaling profile to be reused for arbitrary Cartesian directions or to modulate more complex trajectories parameterized by $s(t)$.

Bezier and Spline Curves.

Bezier curves are expressed as a weighted sum of control points P_i:

$$p(t) = \sum_{i=0}^{n} B_{i,n}(\tau)\, P_i, \quad \tau = \frac{t - t_0}{t_f - t_0}, \tag{4.33}$$

where $B_{i,n}(\tau)$ are Bernstein polynomials. The corresponding velocity can be expressed analytically as

$$\dot{p}(t) = \frac{n}{t_f - t_0} \sum_{i=0}^{n-1} B_{i,n-1}(\tau)\,(P_{i+1} - P_i), \tag{4.34}$$

which results directly from the derivative of the Bernstein basis functions.

Cubic Splines.

Spline interpolation allows to connect a sequence of waypoints while ensuring continuity of position and derivatives. A cubic spline segment between two consecutive waypoints p_i and p_{i+1} is generally written as

$$p_i(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3, \quad t \in [t_i, t_{i+1}], \tag{4.35}$$

where the coefficients a_i, b_i, c_i, d_i are chosen to satisfy position and derivative constraints at the boundaries. The velocity is directly obtained as

$$\dot{p}_i(t) = b_i + 2c_i(t - t_i) + 3d_i(t - t_i)^2. \tag{4.36}$$

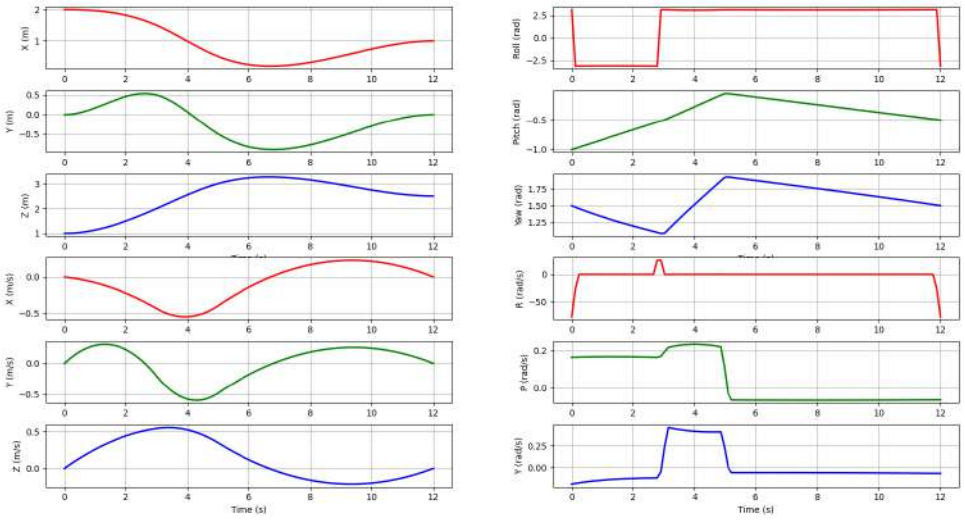


Figure 4.5. Example of a cubic spline trajectory position and orientations(top) and linear and angular velocity (bottom). Orientation used is Slerp.

The spline coefficients a_i, b_i, c_i, d_i are computed by enforcing continuity of position, velocity, and acceleration at the junctions between segments:

$$p_i(t_{i+1}) = p_{i+1}(t_{i+1}), \qquad \dot{p}_i(t_{i+1}) = \dot{p}_{i+1}(t_{i+1}), \qquad \ddot{p}_i(t_{i+1}) = \ddot{p}_{i+1}(t_{i+1}). \tag{4.37}$$

This ensures a globally C^2 continuous trajectory.

Given as input a set of waypoints (and optionally boundary conditions on velocity or acceleration), the spline coefficients a_i, b_i, c_i, d_i are automatically computed so that the overall trajectory is continuous in position and velocity across all segments.

4.3.2 Orientation Trajectories

In addition to position trajectories, the library provides dedicated tools for the generation of smooth orientation trajectories. Rotations are represented with unit quaternions $q \in \mathbb{H}$, which are free from singularities and discontinuities typical of Euler angles. Each interpolation method produces both the interpolated quaternion $q(t)$ and the corresponding angular velocity $\omega(t)$, obtained analytically or numerically from the time derivative.

Spherical Linear Interpolation (SLERP)

The simplest and most widely used method for quaternion interpolation is SLERP (Spherical Linear intERPolation). Given two unit quaternions q_0 and q_1, the interpolation is defined as

$$q(t) = \frac{\sin((1-\tau)\theta)}{\sin(\theta)} q_0 + \frac{\sin(\tau\theta)}{\sin(\theta)} q_1, \quad \tau = \frac{t - t_0}{t_f - t_0}, \tag{4.38}$$

where $\theta = \arccos(q_0 \cdot q_1)$ is the angle between the two quaternions. The angular velocity is obtained from the quaternion logarithmic map:

$$\omega(t) = 2\,\mathrm{Im}\Big(\dot{q}(t)\, q^{-1}(t)\Big), \tag{4.39}$$

where $\mathrm{Im}(\cdot)$ denotes the vector (imaginary) part of a quaternion. This definition guarantees that $\omega(t) \in \mathfrak{so}(3)$.

De Casteljau Algorithm

For more complex trajectories passing through multiple waypoints, the library implements a quaternion-based version of the De Casteljau algorithm. This method generalizes Bézier interpolation to the unit quaternion space by recursively applying SLERP between control quaternions until a single interpolated quaternion is obtained.

Given a segment with control quaternions $\{q_0, q_1, q_2, q_3\}$, the interpolation proceeds as

$$q_i^{(k)}(\tau) = \mathrm{slerp}\big(q_i^{(k-1)}(\tau),\, q_{i+1}^{(k-1)}(\tau),\, \tau\big), \tag{4.40}$$

where $\tau \in [0, 1]$ is the normalized parameter, and the recursion continues until only one quaternion remains:

$$q(\tau) = q_0^{(n)}(\tau). \tag{4.41}$$

This final quaternion $q(\tau)$ represents the interpolated orientation at the desired time instant.

To compute the angular velocity, the interpolated orientation is differentiated. This is achieved by evaluating neighboring quaternions and projecting the result onto the Lie algebra $\mathfrak{so}(3)$ through the quaternion logarithmic map:

$$\omega(t) \approx \frac{1}{\Delta t}\,\mathrm{Log}\Big(q(t+\Delta t)\,q^{-1}(t)\Big), \tag{4.42}$$

The quaternion logarithmic map is defined as

$$\mathrm{Log}(q) = \frac{u}{\|u\|}\,2\,\arccos(q_w), \qquad q = [q_w,\, u] \in \mathbb{H},\ \|u\| = \sin(\theta), \tag{4.43}$$

which maps the quaternion increment to the corresponding rotation vector in $\mathfrak{so}(3)$.

Kochanek–Bartels Splines

To better control the shape of the rotational trajectory, the library also supports Kochanek–Bartels splines. This method introduces three shaping parameters:

$$T \quad \text{(Tension)}, \quad C \quad \text{(Continuity)}, \quad B \quad \text{(Bias)},$$

which allow the user to tune the "tightness" of the curve, the smoothness of tangent continuity, and the asymmetry of the interpolation.

The spline is constructed by generating additional control quaternions from the original waypoints and the chosen TCB parameters. The resulting sequence of control points is then interpolated using the De Casteljau algorithm as described above, producing smooth quaternion trajectories with continuous first derivatives.

As for the De Casteljau scheme, the corresponding angular velocity is obtained by mapping consecutive orientations to the tangent space as in Eq. (4.42); this ensures C^1 continuity and smooth angular velocity profiles along the trajectory. Since quaternions do not form a vector space under subtraction, the tangent is computed in the log-mapped tangent space. Let ξ_i denote the corresponding rotation vectors; then the tangent at each waypoint can be computed as

$$T_i = \frac{(1-T)(1+C)(1+B)}{2}(\xi_i - \xi_{i-1}) + \frac{(1-T)(1-C)(1-B)}{2}(\xi_{i+1} - \xi_i), \tag{4.44}$$

4.3.3 Handling of Position and Orientation with velocities

Each trajectory setup method in the wrapper takes as input both the *position* $p \in \mathbb{R}^3$ and the *orientation* of the end-effector, expressed in roll–pitch–yaw or quaternion form.

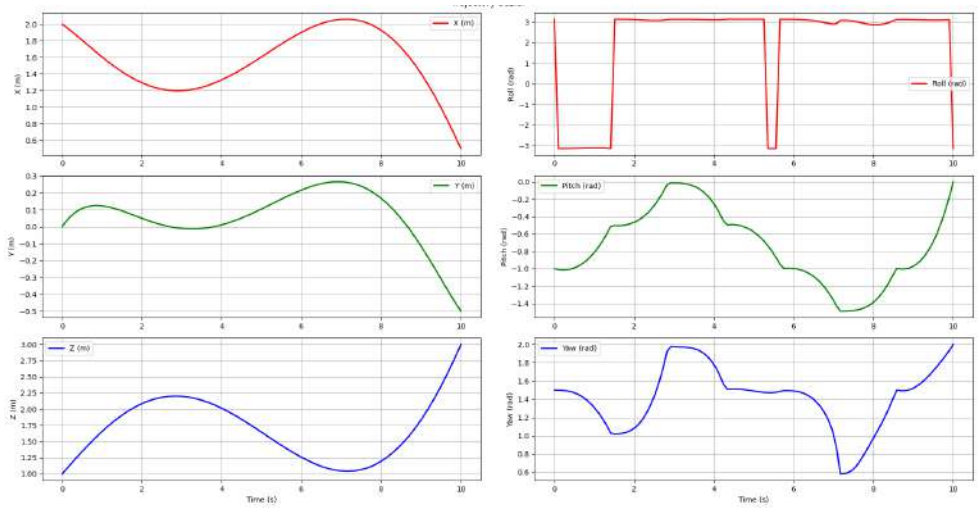


Figure 4.6. Example of a Bezier trajectory position (right) and (left) Kochanek-Bartels interpolation for orientation.

Internally, the wrapper automatically separates the two components:

- **Position:** managed by interpolation methods such as polynomial profiles, cubic splines, trapezoidal laws, Bézier curves or circular arcs. These return position $p(t)$ and linear velocity $\dot{p}(t)$.
- **Orientation:** handled through quaternion-based interpolation (e.g., SLERP, De Casteljau, Kochanek–Bartels). These return the interpolated quaternion $q(t)$ and the angular velocity $\omega(t)$, computed consistently in $\mathfrak{so}(3)$.

The two parts are then combined into a single trajectory description

$$\mathcal{T}(t) = \big(p(t), \dot{p}(t), q(t), \omega(t)\big),$$

which provides the complete kinematic state required by the end-effector controller.

To make the selection of the trajectory type straightforward, the wrapper exposes specific `set` methods such as `.poly5()`, `.cubic_spline()`, `.bezier()`, `.trapezoidal()`, and `.circular()`. Each of these methods automatically configures the appropriate interpolation for position and orientation, ensuring that the controller always receives coherent references.

This design hides the complexity of handling different spaces ($\mathbb{R}^3$ for translations and $\mathbb{H}$ for rotations) while offering the user a uniform and intuitive interface for trajectory generation.

4.3.4 Cache for Real-Time Execution.

To further improve efficiency, the library implements a caching mechanism. Whenever the trajectory is evaluated at a given time t, the corresponding state (position, velocity, acceleration, orientation, angular velocity) is stored and can be retrieved later with negligible cost. The cache keys are quantized using an expected jitter tolerance, so that repeated queries with slightly different timestamps still map to the same stored value:

$$k(t) = \text{round}\left(\frac{t}{\Delta t_{\text{jitter}}}\right) \Delta t_{\text{jitter}}. \tag{4.45}$$

An adaptive strategy is also available: the library continuously monitors the sampling intervals and automatically adjusts the jitter tolerance Δt_{jitter} based on the observed variability. This ensures robustness to real-time scheduling noise while keeping the cache efficient. In practice,

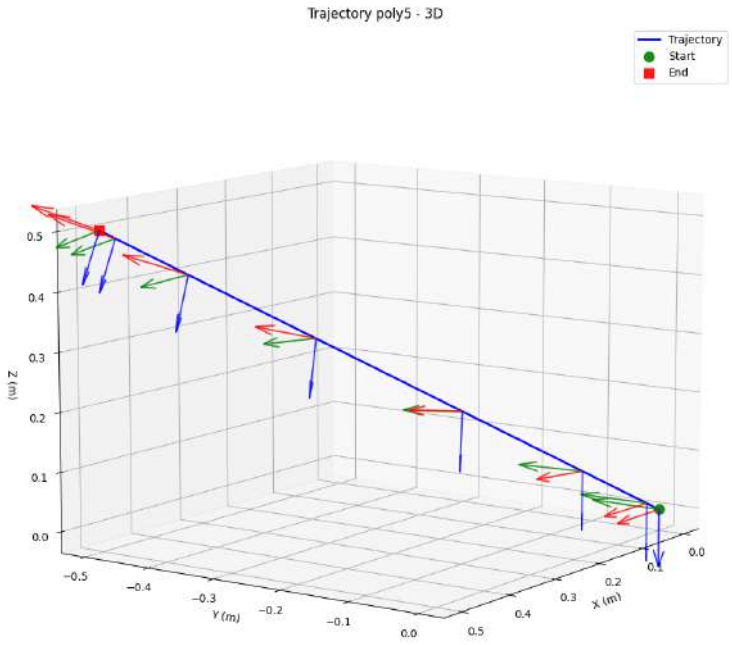


Figure 4.7. Built-in trajectory visualization tool.

this mechanism significantly reduces redundant computations, making the trajectory generator suitable for high or low-frequency control loops.

Visualization Tool

To facilitate debugging and analysis, the library includes a built-in visualization tool. Fig.4.2,4.3, 4.4, 4.5, and 4.6 show some examples of the generated trajectories. These figures are taken directly from the visualization tool. This tool can plot the generated trajectories in both position and orientation spaces, allowing users to visually inspect the smoothness and correctness of the interpolations. It supports also the interactive visualization of position and orientation trajectories over time in 3D. If user spcifies a time range, the tool animates the end-effector frame motion along the trajectory. Fig.4.7 showns an example of the visualization tool interface, that catches a samples of a trajectory over time.

4.4 Tasks used in Task-Priority Library

4.4.1 End-effector Control of Multiple Robots

The End-Effector (EE) control task drives each manipulator (or mobile manipulator) towards a desired Cartesian pose, which may correspond to a fixed target or to a time-varying trajectory. Within the task-priority framework, it is modeled as an ET and often assigned a lower priority.

For each robot, the actual EE pose $x = [p_{\text{EE}}, r_{\text{EE}}]$ is obtained from forward kinematics, where $p_{\text{EE}} \in \mathbb{R}^3$ is the position and $r_{\text{EE}} \in \mathbb{H}$ is the quaternion orientation. The reference pose $x^*(t) = [p^*(t), r^*(t)]$ is provided either by a trajectory generator described in Sec. 4.3, or as a fixed target.

When tracking a trajectory, the velocity reference must also include the feedforward term $\dot{x}_{\text{EE}}$, i.e., the desired Cartesian velocity. According to the task-priority scheme, the EE control law is expressed as

$$\dot{x}^*_{\text{EE}} = \dot{x}_{\text{EE}} + K_{\text{EE}}\, \widetilde{e} \tag{4.46}$$

with $\widetilde{e} = [e_p^\top \; e_r^\top]^\top$. In practice, K_{EE} is chosen block-diagonal (e.g., $K_{\text{EE}} = \text{diag}(K_p, K_r)$). The control law combines two terms:

- a *feedforward contribution* ($\dot{x}_{\text{EE}}$), when tracking a trajectory, given by the desired Cartesian velocity, obtained by the trajectory generator, discussed in Sec. 4.3;
- a *feedback correction*($\widetilde{e}$), based on the position and orientation errors,

For each robot defined in the framework and for each EE the correction term related the position error is computed as the difference between reference and actual position.

$$e_p = p^*(t) - p_{\text{EE}}(q). \tag{4.47}$$

Similarly, the orientation error is derived from the quaternion representation and approximated as

$$e_r = \frac{2}{\Delta t} E(r^*)\, r_{\text{EE}}, \tag{4.48}$$

where Δt is the controller sampling time, and $E(r) \in \mathbb{R}^{3\times 4}$ is the matrix constructed from the unit quaternion, represented as the vector $r = [r_0\ r_1\ r_2\ r_3]^\top$ with r_0 being the scalar part:

$$E(r) = \begin{bmatrix} -r_1 & r_0 & -r_3 & r_2 \\ -r_2 & r_3 & r_0 & -r_1 \\ -r_3 & -r_2 & r_1 & r_0 \end{bmatrix}.$$

The single Jacobian associated with the EE task is defined in the standard way, as already discussed in Sec. 4.2. When multiple robots are involved, the formulation is naturally extended using Eq.4.10 by stacking the Jacobians of each end-effector into a block structure.

For the i-th robot with configuration $q^{(i)}$ and Jacobian $J^{(i)}(q^{(i)}) \in \mathbb{R}^{6\times n_i}$, the global multi-robot Jacobian is defined as

$$J(q) = \begin{bmatrix} J^{(1)}(q^{(1)}) & 0 & \cdots & 0 \\ 0 & J^{(2)}(q^{(2)}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J^{(N)}(q^{(N)}) \end{bmatrix}, \tag{4.49}$$

where each block corresponds to one end-effector, and n_i denotes the number of joints of the i-th robot. The Jacobians are constructed as described in Sec. 4.2, taking into account the specific kinematics of each robot (e.g., mobile base, manipulator arm and so on).

In this way, the error vectors and reference velocities of all end-effectors can be stacked consistently, yielding a unified formulation for the multi-robot EE task:

$$\dot{x} = \begin{bmatrix} \dot{x}_{\text{EE}}^{(1)} \\ \vdots \\ \dot{x}_{\text{EE}}^{(N)} \end{bmatrix}, \quad \widetilde{e} = \begin{bmatrix} e^{(1)} \\ \vdots \\ e^{(N)} \end{bmatrix}.$$

This block-diagonal structure preserves the modularity of the framework: adding or removing robots simply amounts to adding or removing blocks in the stacked Jacobian, while the robots are able to execute their trajectory tasks independently.

4.4.2 Joint Trajectory Control

In robotic systems combining a mobile base and a manipulator, the concept of *joint control* must be interpreted differently for the two subsystems. In industrial manipulators, it is common to command each actuator directly in joint space, as joint encoders are readily available and the kinematic structure is fixed. Conversely, for mobile bases, direct joint–space control (e.g., wheel angle or wheel velocity control) is seldom implemented in practice, since navigation and motion

planning are more naturally defined in Cartesian coordinates (position and orientation of the base frame).

For this reason, without loss of generality, in this work we treat the joint control $q_{\text{ref}}(t)$ task as a *hybrid* ET, where:

- the **manipulator** using joint–space control, where each joint is assigned a time parameterized reference 1D-trajectory;
- the **mobile base** using Cartesian–space control, where reference trajectories are defined in terms of the base frame pose and velocities.

This separation allows the overall control framework to remain unified at the task level. Manipulator joint–space trajectories are generated using smooth 1D trajectory profiles introduced in Sec. 4.3.1 and scaled by means of joint actual and target information. For the base, Cartesian trajectories are transformed from world into body–frame velocity commands through the corresponding kinematic strategies, depending on the base type (differential or omnidirectional).

In joint-space trajectory control the reference input of the joint must be mapped directly without any need for transformation. For this reason, the associated task Jacobian reduces to the identity matrix:

$$J_{\text{JT}}(q) = I_n, \tag{4.50}$$

with n the total number of joints (base + arm + gripper).

Mobile Base Actuation Control

The objective in mobile base trajectory control is to ensure that the robot follows a given time-varying reference trajectory $(x_r(t), y_r(t))$, using only position and velocity measurements.

To recap, we model the mobile base joints as a planar system commanded by body-frame velocities. $u_b^{\text{omni}} = \left[v_x^b, v_y^b, \omega\right]^T$ in case where v_x^b and v_y^b are the linear velocities along the x and y cartesian axes of the base frame, and ω is the yaw rate, and $u_b^{\text{diff}} = \left[v_x^b, \omega\right]^T$ in case of a differential-drive base, where $v_y^b = 0$, since the nonholonomic constraint imposes that the base can only move forward or backward and rotate about its vertical axis.

Differential Drive Base. For non-holonomic systems—such as differential-drive robots— computation through jacobian is challenging because their motion constraints prevent movement in certain directions. In particular, differential-drive robots cannot generate lateral motion, since their Jacobian is rank-deficient; thus, only trajectories compatible with their kinematics can be tracked. Ensuring that the reference path satisfies these constraints is essential, as infeasible trajectories (e.g., requiring sideways motion) cannot be executed without reparameterization. These non-holonomic restrictions introduce nonlinearities that demand dedicated control strategies rather than standard linear controllers. To address these challenges, a control strategy proposed by [97] was adopted. This approach employs a state observer to estimate the robot's orientation from position and velocity measurements only, avoiding the need for direct orientation sensing. A controller is then designed to minimize the position error between the robot and the reference trajectory, and the proof of the concept is validated in [97]. The tracking controller is composed of two main components:

1-Orientation Estimator using State Observer. Consider the kinematic model of a nonholonomic wheeled robot, the mapping from base to world frame is given by:

$$\dot{x}(t) = v_x(t)\cos\theta, \quad \dot{y}(t) = v_x(t)\sin\theta, \quad \dot{\theta}(t) = \omega(t). \tag{4.51}$$

where $x(t)$, $y(t)$ denote the Cartesian position, $\theta(t)$ the robot orientation, and $v_x(t)$, ω are the linear and angular velocity inputs, respectively. Since the orientation $\theta(t)$ is not directly measurable, an observer is introduced to estimate the values of $\cos\theta$ and $\sin\theta$. Without loss of generality the variable of time is omitted for clarity.

Define the augmented state vector $z = [x, y, \cos\theta, \sin\theta]^T$, and the observer dynamics are given by:

$$\begin{aligned}
\dot{\hat{z}}_1 &= \hat{z}_3 v_x + k_1(x - \hat{z}_1)v_x, \\
\dot{\hat{z}}_2 &= \hat{z}_4 v_x + k_1(y - \hat{z}_2)v_x, \\
\dot{\xi}_3 &= -\hat{z}_4\omega - k_1\hat{z}_3 v_x + (x - \hat{z}_1)v_x, \\
\dot{\xi}_4 &= \hat{z}_3\omega - k_1\hat{z}_4 v_x + (y - \hat{z}_2)v_x, \\
\hat{z}_3 &= \xi_3 + k_1 x, \\
\hat{z}_4 &= \xi_4 + k_1 y,
\end{aligned} \tag{4.52}$$

where $k_1 > 0$ is a tuning parameter, and ξ_3, ξ_4 are internal observer states. Under mild conditions (including persistent excitation), it can be shown that the estimation errors decay exponentially, ensuring convergence of $\hat{z}_3$ and $\hat{z}_4$ to $\cos\theta$ and $\sin\theta$, respectively.

2-Trajectory Tracking Control Law. The objective is to ensure that the robot follows a given time-varying reference trajectory $(x_r(t), y_r(t))$ expressed in world frame, using only position and velocity measurements. Define the position errors:

$$\bar{x} = x - x_r(t), \quad \bar{y} = y - y_r(t). \tag{4.53}$$

The linear velocity input $v_x(t)$ is computed as:

$$v_x(t) = \sqrt{(\dot{x}_r - k_2\tanh(\bar{x}))^2 + (\dot{y}_r - k_2\tanh(\bar{y}))^2}, \tag{4.54}$$

with $k_2 > 0$. This formulation ensures that $v_x(t)$ remains strictly positive and bounded, which is essential to guarantee the stability of the observer and the convergence of the control scheme.

To guide the robot towards the reference trajectory, the angular velocity command is defined as:

$$\omega(t) = \dot{\hat{\theta}}^* - k_3\left(\hat{z}_4\cos\theta^* - \hat{z}_3\sin\theta^*\right), \tag{4.55}$$

where θ^* is an auxiliary orientation defined through:

$$\cos\theta^* = \frac{\dot{x}_r - k_2\tanh(\bar{x})}{v_x(t)}, \quad \sin\theta^* = \frac{\dot{y}_r - k_2\tanh(\bar{y})}{v_x(t)}. \tag{4.56}$$

The term $\dot{\hat{\theta}}^*$ represents an estimate of the derivative of θ^* and is computed based on reference accelerations and the output of the observer.

To ensure convergence, the control design must satisfy specific conditions. In particular, the linear velocity must remain strictly positive, i.e., $v_x(t) \geq v_x m > 0$, to preserve observer stability and guarantee persistent excitation. Moreover, the gain k_2 must satisfy

$$0 < k_2 < \frac{\sqrt{2}}{2} v_{\text{rm}}, \tag{4.57}$$

where v_{rm} is a known lower bound on the reference linear velocity. These conditions ensure that the velocity command $v_x(t)$ does not approach zero, preventing loss of observability. Finally, under suitable initial conditions and appropriate tuning of gains k_1, k_2, and k_3, a Lyapunov-based analysis proves that both orientation and position errors converge asymptotically to zero.

Omnidirectional Base. In contrast to differential platforms, an omnidirectional base is *fully actuated* in $SE(2)$, meaning it can generate independent translational and rotational velocities. Its control problem is therefore less constrained, as any feasible Cartesian trajectory can be tracked directly in the workspace.

Given a Cartesian trajectory reference $p_r(t) = [x_r(t), y_r(t), \theta_r(t)]^T$, and define the tracking error as $e_p = p_r - p_b$, where $p_b = [x_b(t), y_b(t), \theta_b(t)]^T$ is the current base pose. The desired control law in the Cartesian frame is computed by means of feedforward and feedback terms:

$$u_r = \dot{p}_r(t) + K_{\text{omni}}\, e_p, \tag{4.58}$$

where $K_{\text{omni}} = \text{diag}(k_x, k_y, k_\theta)$ is a positive-definite gain matrix. The corresponding desired cartesian control input u_r is then mapped into body-frame velocities u_b through the inverse of the appropriate Jacobian.

$$u_b = J_{\text{omni}}^{-1}(\theta_b)\, u_r. \tag{4.59}$$

For an omnidirectional base, the Jacobian is full rank, allowing direct inversion and it is typically defined as

$$J_{\text{omni}}(\theta_b) = \begin{bmatrix} \cos\theta_b & -\sin\theta_b & 0 \\ \sin\theta_b & \cos\theta_b & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.60}$$

This Jacobian differs from the implementation reported in Eq. (4.18), as it is defined with respect to the origin of the base frame rather than the end-effector position. Moreover, it models the planar kinematics of the mobile base, thus describing motions in the configuration space $SE(2)$ instead of the full spatial domain $SE(3)$.

Manipulator Joint Control. The manipulator joints are directly commanded in joint space, following smooth time–parameterized reference trajectories. For each joint i, the reference position $q_{i,r}(t)$ and velocity $\dot{q}_{i,r}(t)$ are generated by one of the interpolation methods presented in Sec. 4.3.

The control law is defined as

$$\dot{q}_i = \dot{q}_{i,r}(t) + k_i\big(q_{i,r}(t) - q_i(t)\big), \tag{4.61}$$

with $k_i > 0$ the proportional gain for the i-th joint. In compact form:

$$\dot{q} = \dot{q}_r + K_q(q_r - q), \tag{4.62}$$

where $K_q = \text{diag}(k_1, \ldots, k_n)$.

4.4.3 Singularity Avoidance

The Singularity Avoidance (SA) task is formulated as an IT aimed at keeping the robot away from singular configurations. Given the end-effector Jacobian $J(q)$, it can be decomposed via Singular Value Decomposition (SVD) as

$$J(q) = U\Sigma V^\top, \tag{4.63}$$

where Σ is the diagonal matrix of singular values $\sigma_1 \geq \cdots \geq \sigma_r$, with $r = \min(m, n)$. These values quantify the manipulability of the system along different task-space directions.

The SA task specifically monitors the smallest singular value $\sigma^{\min}(J) = \sigma_r$, which tends to zero near singularities, indicating loss of controllability. The corresponding joint-space direction is given by the last right-singular vector v_r (last column of V). In this library, we encode this direction as a Jacobian-like row $J_{\text{SA}} \in \mathbb{R}^{1\times n}$, i.e., $J_{\text{SA}} = v_r^\top$.

To modulate this task near singularities, the activation matrix $A_{\text{SA}} \in \mathbb{R}^{1\times 1}$ is defined according to the general formulation in eq. (4.8). The activation depends on the smallest singular value and smoothly transitions from full activation (close to singularities) to complete deactivation (far from them). This behavior is obtained through the sigmoid function:

$$a_{\text{SA}} = s\left(\sigma^{\min}(J), \sigma_{\text{th}}^{-}, \sigma_{\text{th}}^{+}\right), \tag{4.64}$$

where $\sigma_{\text{th}}^{-} < \sigma_{\text{th}}^{+}$ are user-defined activation thresholds.

Finally, to regulate the system, a proportional control law drives $\sigma^{\min}(J)$ toward a desired threshold x^*, with the task error defined as:

$$\tilde{x}_{\text{SA}} = K_{\text{SA}}(x^* - \sigma^{\min}(J)). \tag{4.65}$$

4.4.4 Joint Limits Avoidance

The Joint Limit Avoidance (LA) task prevents the robot joints from approaching their mechanical boundaries by introducing repulsive actions near the limits. This is achieved through a diagonal activation matrix $A_{\text{LA}} \in \mathbb{R}^{n\times n}$, where each diagonal entry $a_{j,j}$ represents the activation of joint j.

The activation value is computed as the maximum between two sigmoid-based functions:

$$\begin{aligned} a_{LA_{j,j}} &= \max\left(s_j^{+}, s_j^{-}\right) \\ s_j^{+} &= 1 - s\left(q_j, q_j^{+} - \Delta q_j^{+}, q_j^{+}\right) \\ s_j^{-} &= s\left(q_j, q_j^{-}, q_j^{-} + \Delta q_j^{-}\right), \end{aligned} \tag{4.66}$$

where q_j^{+} and q_j^{-} denote the upper and lower joint limits, while Δq_j^{+} and Δq_j^{-} define smooth activation margins near the bounds. This formulation ensures gradual engagement and prevents discontinuous control responses.

The task Jacobian is the identity, $J_{\text{LA}} = I_n$, as each joint is regulated independently. A proportional control law then pushes joints away from the boundaries. With $K_{\text{LA}} \in \mathbb{R}^n$ denoting proportional gains, the task error is defined as:

$$\tilde{x}_{LA_j} = \begin{cases} K_j\left(q_j^{+} - \Delta q_j^{+} - q_j\right), & \text{if } q_j > q_j^{+} - \Delta q_j^{+} \\ K_j\left(q_j^{-} + \Delta q_j^{-} - q_j\right), & \text{if } q_j < q_j^{-} + \Delta q_j^{-} \\ 0, & \text{otherwise.} \end{cases} \tag{4.67}$$

This task is available just for manipulators, as mobile bases typically do not have joint limits, but cartesian limits task needs to be considered.

4.4.5 Obstacle Avoidance Through Robotic Distance Function

Collision Avoidance (CA) is formulated as an IT to ensure a minimum safety distance between the robot and potential obstacles, represented by a point cloud $\mathcal{P}$ in the world frame. The goal of the CA task is to generate a repulsive action between the point with the highest collision risk and the corresponding point on the robot surface.

To compute distances, the RDF method introduced in Sec. 3.3 is employed. Since the SDF weights of each link are trained in its own local frame, every point cloud must first be transformed from world to link coordinates before applying Eq. 3.29.

For both computational and conceptual clarity, the overall point cloud is divided into two subsets: the environment cloud $\mathcal{P}^{\text{env}}$, shared across all links, and the self cloud $\mathcal{P}^{\text{self}}$, filtered for

each link using a precomputed collision map. From a computational standpoint, the environment cloud has a fixed size and can therefore be processed in parallel, whereas the link-specific cloud has variable size and must be handled individually. From a control perspective, this separation also enables assigning distinct safety margins for Self-Collision Avoidance (SCA) and Environment Collision Avoidance (ECA), denoted as d^{self} and d^{env}, respectively. This allows the user to independently tune the relative clearance between self and environment collisions. Critical points are then defined as

$$\mathcal{P}_{\text{crit}}^{\text{self}} = \{p \in \mathcal{P}^{\text{self}} \mid f(p) < d_{\text{min}}^{\text{self}}\}, \quad \mathcal{P}_{\text{crit}}^{\text{env}} = \{p \in \mathcal{P}^{\text{env}} \mid f(p) < d_{\text{min}}^{\text{env}}\},$$

and their union yields the global set of critical points, $\mathcal{P}_{\text{crit}}$. The most critical point is finally obtained as

$$\bar{p} = \arg \min_{p \in \mathcal{P}_{\text{crit}}} f(p). \tag{4.68}$$

Once identified, the critical point is projected onto the robot link surface using Eq. (3.69). The resulting points are expressed in the world frame as $p_{\text{robot}}^{\text{world}}$ (robot side) and $p_{\text{coll}}^{\text{world}}$ (obstacle side). Since $\nabla f(\cdot)$ is computed in the link local frame, it is rotated into the world frame through the link transformation.

The Jacobian associated with the robot critical point is defined as

$$J_j(q) = -\nabla f^\top(p_{\text{coll}}^{\text{world}}) J_j^{\text{lin}}(q), \tag{4.69}$$

where J^{lin} is the linear part of the Jacobian at the collision point and the $-\nabla f^\top(\cdot)$ term, computed in Sec.3.3.7, is the gradient direction pointing away from the obstacle. The overall CA Jacobian is then given by:

$$J_{\text{CA}}(q) = \begin{bmatrix} J_1(q) \\ \vdots \\ J_j(q) \\ \vdots \\ J_n(q) \end{bmatrix}. \tag{4.70}$$

To separately account for self- and environment-collision risks, the entries of the activation matrix are defined as:

$$\begin{aligned} a_{\text{CA}_{j,j}} &= \max\left(s_{j_{\text{env}}}, s_{j_{\text{self}}}\right) \\ s_{j_{\text{env}}} &= s\left(f(\bar{p}_{\text{env}}), d_{j_{\text{env}}}^{\text{min}}, d_{j_{\text{env}}}^{\text{max}}\right) \\ s_{j_{\text{self}}} &= s\left(f_j(\bar{p}_{\text{self}}), d_{j_{\text{self}}}^{\text{min}}, d_{j_{\text{self}}}^{\text{max}}\right), \end{aligned} \tag{4.71}$$

where $d_{j_{\text{env}}}^{\text{min}}, d_{j_{\text{env}}}^{\text{max}}$ and $d_{j_{\text{self}}}^{\text{min}}, d_{j_{\text{self}}}^{\text{max}}$ are tunable thresholds.

Finally, to prevent links from reaching collisions, a barrier function control law is employed:

$$\widetilde{x}_{j_{\text{CA}}} = K_{j_{\text{CA}}} \frac{1}{f_j(p^*) - d_j^{\text{safe}}}, \tag{4.72}$$

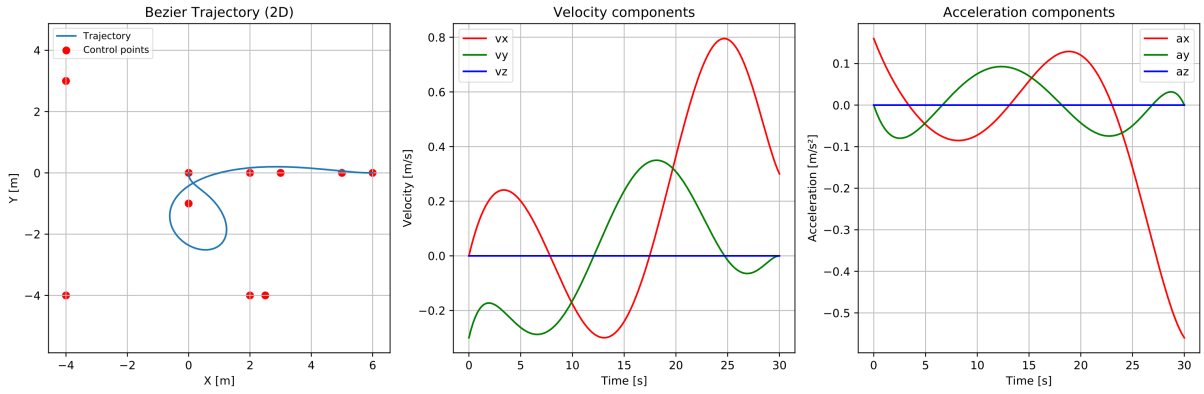
where for the i-th link, d_i is the SDF value and d_i^{safe} is the minimum admissible distance.

4.5 Simulation and Experiments

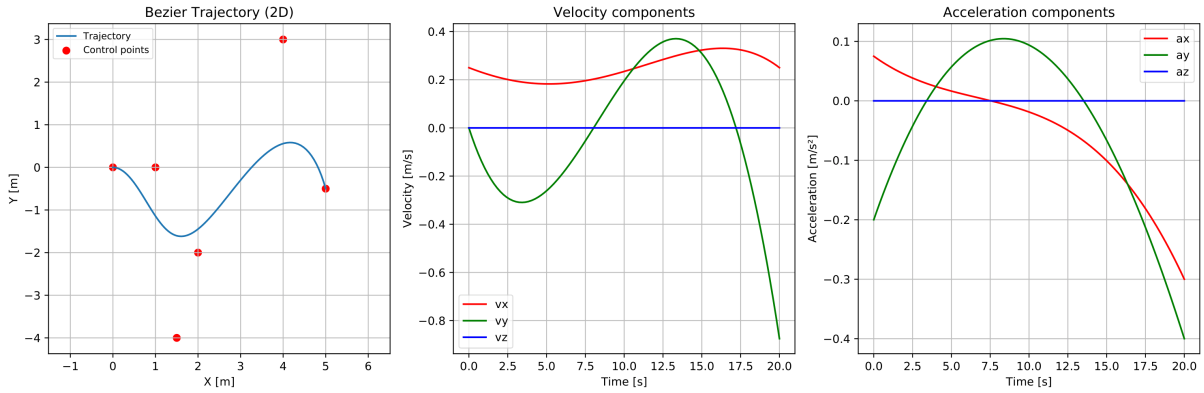
4.5.1 Joint Trajectory Task

Differential-drive base.

In this experiment, the Tiago-robot is commanded to follow a simple trajectory, so the jacobian of this task is $J_{\text{JT}} = I_2$, with two degrees of freedom of command of the differential-drive base.



(a) First Trajectory profile.



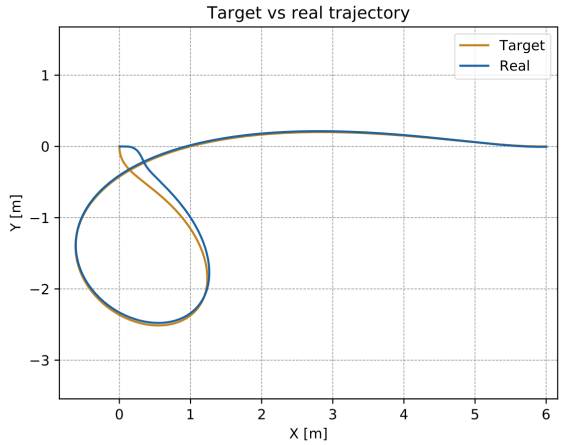
(b) Second Trajectory profile.

Figure 4.8. Trajectory profiles for differential-drive base, comprising position, velocity, and acceleration over time.

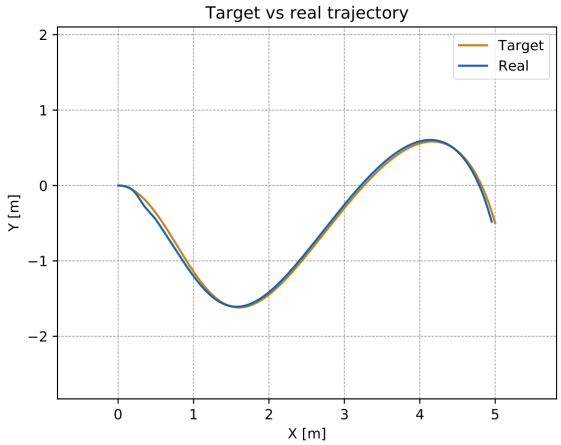
The experiments presented in Fig. 4.8–4.10 assess the capability of the differential-drive base to accurately follow two distinct Bezier trajectories while maintaining consistent estimation of its orientation.

Figure 4.8 illustrates the profiles of both trajectories, including position, velocity, and acceleration components. The first trajectory (4.8a) spans approximately 12.5 m and is executed over 30 s, whereas the second one (4.8b) covers 7.8 m in 20 s. The corresponding velocity and acceleration plots reveal smooth transitions and bounded accelerations, ensuring dynamically feasible motions without exceeding the platform's actuation limits.

The resulting Cartesian tracking performance is depicted in Fig. 4.9. In both cases, the robot successfully follows the desired path (orange) with minimal deviation, as shown by the near overlap between the desired and real trajectories. In the first experiment (4.9a), a larger transient error is visible at the beginning of the motion, due to the nonholonomic constraint preventing the robot from immediately reproducing the lateral component of the reference velocity. After

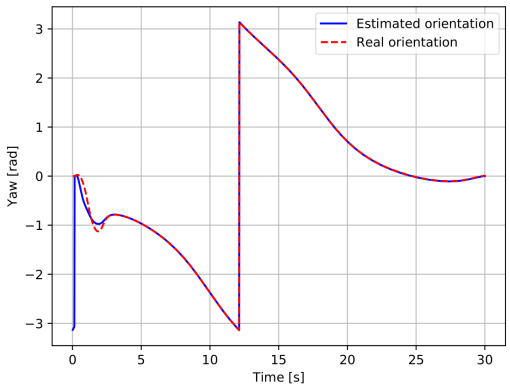


(a) First trajectory execution in the Cartesian plane.

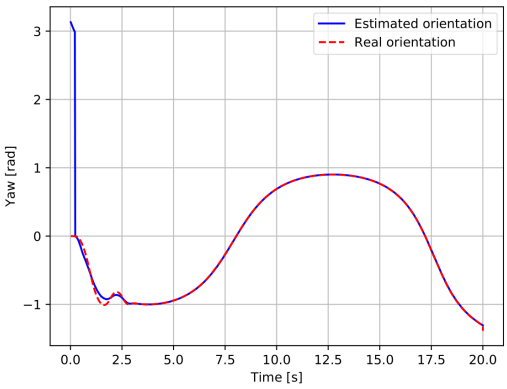


(b) Second trajectory execution in the Cartesian plane.

Figure 4.9. Trajectory tracking for a differential-drive base, showing the desired (orange) and actual (blue) paths for two different executions.



(a) First trajectory: estimated vs. real position angle.



(b) Second trajectory: estimated vs. real position angle.

Figure 4.10. Observer estimation of the position angle for a differential-drive base, comparing the estimated (blue) and actual (red) angles over time.

this initial adjustment, the controller ensures tight convergence to the target curve. In contrast, the second trajectory (4.9b), which contains predominantly longitudinal motion, exhibits a smoother and faster convergence.

Quantitatively, the final position error for the first trajectory is approximately 0.7 cm over a path length of 12.5 m (relative error 0.05%), while the second trajectory yields a final error of 4.5 cm over 7.8 m (0.57%). These results confirm the high accuracy of the control architecture even under nonholonomic motion constraints.

Finally, Fig. 4.10 compares the estimated and real yaw angles during the two experiments. The observer effectively reconstructs the orientation throughout the motion, with only a small transient discrepancy at the beginning of the first trajectory. The estimated orientation quickly converges to the true one and remains well aligned for the remainder of the execution, demonstrating the robustness of the estimation process.

Overall, these results validate the proposed control and observation framework, highlighting its ability to ensure accurate trajectory tracking and reliable orientation estimation for a differential-drive mobile base.

Dual-Subsystem Task Execution: Omni-Base and UR Arm

Figure 4.11. Simulation model of the mobile manipulator, consisting of an omnidirectional base and a 6-DOF UR robotic arm.

In this experiment takes into exam the system Ur5+Omnidirectional in Fig.4.11, the omnidirectional mobile base and the UR robotic arm were commanded to execute coordinated reference trajectories while remaining decoupled in control space. The base was initialized at the origin of the workspace, whereas the arm started from a random but valid configuration within its joint limits. The Jacobian is set as $J_{\text{JT}} = I_9$ to enable independent control of both subsystems three degrees of freedom for the base and six for the arm.

The mobile platform followed a planar Cartesian trajectory generated via a cubic spline interpolation of four waypoints (see Sec.4.3):

$$p_i = \{[0.5, -0.3, 0], [1.0, -0.7, 0], [1.5, -1.2, 0], [1.7, -1.5, 0]\},$$

executed over a 30 s period with a uniform gain vector $K_{\text{omni}} = [0.5, 0.5, 0.5]$. As shown in Fig. 4.12, the base starts from the origin and reaches the final target smoothly, exhibiting consistent tracking of both position and orientation references.

The UR arm was driven independently using a fifth-order polynomial (poly5) trajectory connecting its initial pose to the desired joint configuration

$$q_d = [0.0, -1.57, 1.57, -1.57, -1.57, 0.0],$$

with $K_{\text{arm}} = [2, 2, 2, 2, 2, 2]$ and a total execution time of 30 s. As reported in Fig. 4.12b, all six joints closely follow their references, confirming accurate trajectory generation and joint-level control performance.

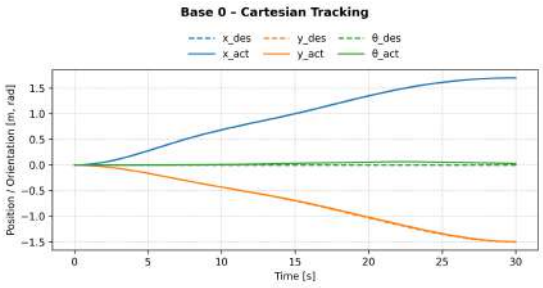
Joint-Space Trajectory Execution with Dual Panda Arms

In this experiment, two Panda arms were commanded independently through the Joint Trajectory Control task to follow defined joint-space trajectories. The jacobian is built as a block-diagonal matrix including both arms, allowing simultaneous control of all 14 joints, so $J_{\text{JT}} = I_{14}$, 7 for each panda arm The trajectory duration was set to 5 s with a uniform proportional gain $K_{\text{arm}} = 2.0$ for all active joints.

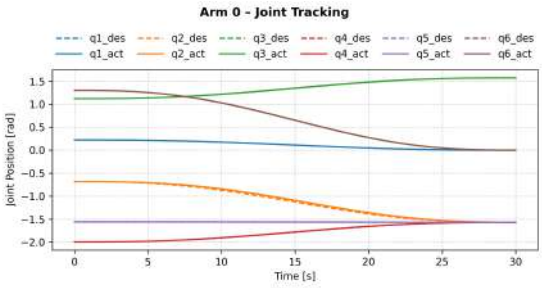
The function `set_arm_trajectories_pnts` allows the definition of target joint positions where entries set to `None` are automatically ignored. When a joint entry is `None`, that degree of freedom remains fixed at its initial position throughout the motion, whereas all other joints follow a smooth fifth-order polynomial (*poly5*) trajectory toward the specified target.

The desired joint configurations for the two arms were:

$$q_d^{(0)} = [0.1, -0.5, \text{None}, -2.0, 0, \text{None}, 0], \quad q_d^{(1)} = [\text{None}, \text{None}, 0, -2.0, 0, 1.0, 0].$$

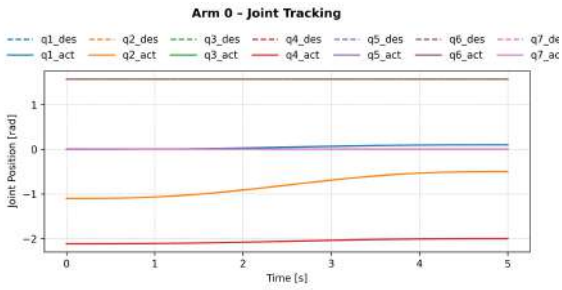


(a) Base 0 – Cartesian tracking (cubic spline).

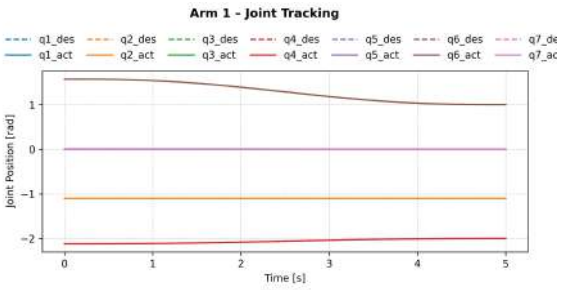


(b) Arm 0 – Joint tracking (poly5 trajectory).

Figure 4.12. Coordinated task execution of the omnidirectional base and UR arm. The base follows a spline-defined planar path from the origin, while the arm moves along a fifth-order polynomial joint trajectory.



(a) Arm 0 – joint-space trajectory.



(b) Arm 1 – joint-space trajectory.

Figure 4.13. Execution of partially defined joint-space trajectories for the two Panda arms. Joints with specified targets follow a fifth-order polynomial (*poly5*) profile, while those set to `None` remain fixed. The tracking performance is nearly perfect, as desired and actual curves are fully overlapping. Each trajectory is completed within 5 s.

As shown in Fig. 4.13, only the joints with defined targets evolve along continuous trajectories, while those set to `None` remain constant. The desired and actual profiles are nearly indistinguishable, confirming an almost perfect tracking performance with minimal steady-state error. Both trajectories complete in 5 s, demonstrating the controller's capability to handle partial joint activation within a unified trajectory generation framework.

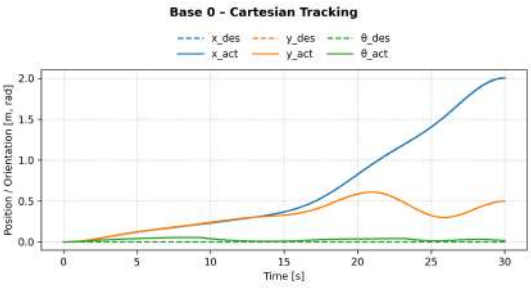
Hierarchical Base–Arm Cartesian Control

In this experiment, a dual–layer control hierarchy is implemented to demonstrate redundancy resolution between a mobile base and a manipulator arm. The mobile platform is a holonomic base with three planar degrees of freedom, while the manipulator is a 6–DOF UR–type arm. The resulting system therefore exhibits kinematic redundancy, with nine total controllable joints.

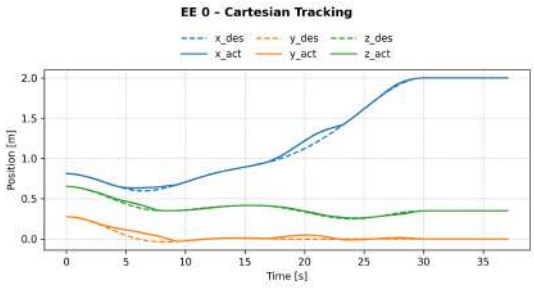
To achieve a structured task decomposition, two priority levels are defined:

- **Upper level:** *JointTrajectoryControl* — responsible for the base motion in the Cartesian plane.
- **Lower level:** *ArmCartesianControl* — responsible for maintaining the desired end–effector trajectory.

At the upper level, the activation matrix $A^{(1)}$ selectively activates only the first three joints (corresponding to the base), while deactivating the arm joints. Conversely, the lower–level activation matrix $A^{(2)}$ is fully active, allowing the manipulator to exploit its six degrees of freedom



(a) Base Cartesian tracking. The mobile platform follows a cubic–spline trajectory along X and Y.



(b) End–effector Cartesian tracking. The manipulator maintains $Y = 0$ while following the desired X–direction trajectory using a fifth–order polynomial profile.

Figure 4.14. Hierarchical control of the mobile manipulator. The base and manipulator tasks are executed concurrently, demonstrating redundancy resolution and decoupled Cartesian motion.

within the null space of the upper task:

$$A^{(1)} = \text{diag}(1,1,1,0,0,0,0,0,0), \qquad A^{(2)} = I_9.$$

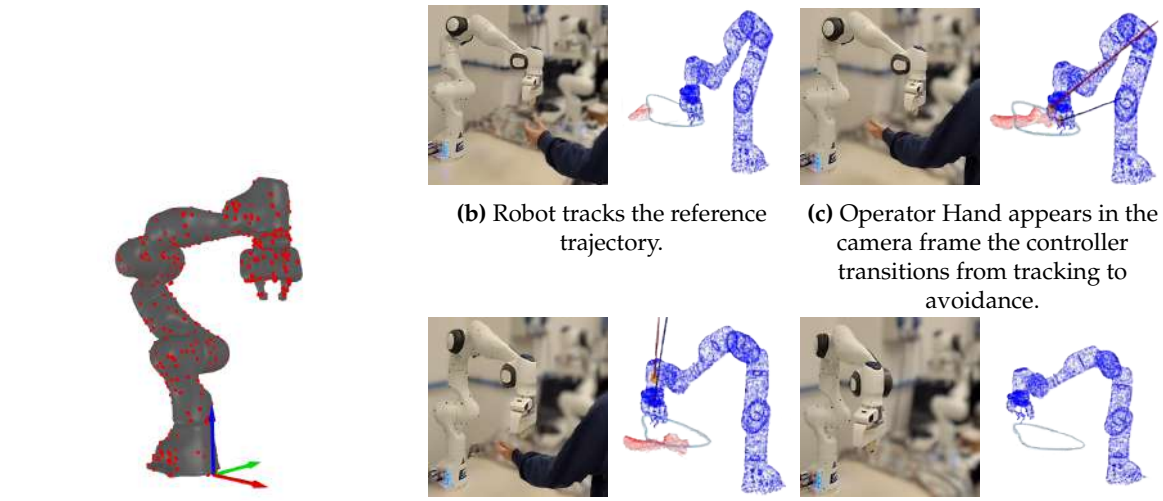
This hierarchical composition ensures that the base trajectory task is prioritized, while the arm task operates in the null space, preserving task feasibility and smooth motion coordination.

The demonstrative scenario consists of two sequential phases. First, the base is commanded to reposition such that the end–effector aligns with the Y–axis (i.e., $Y = 0$). Then, the base moves along the Y–direction while both the base and the manipulator move forward along the X–axis. During this coordinated motion, the end–effector tries to maintain its $Y = 0$ constraint, effectively decoupling its lateral motion from the base displacement. This configuration highlights the redundancy exploitation and smooth task projection between base and arm.

Figures 4.14a–4.14b show the corresponding Cartesian tracking results. The base trajectory (Fig. 4.14a) follows a smooth path along X and Y, while the manipulator (Fig. 4.14b) once reached $Y = 0$ tries to mantain this value constant while moving along X. Although rapid base accelerations may induce transient errors in the end–effector tracking, the hierarchical scheme successfully achieves the overall task objectives, demonstrating the feasibility and robustness of the proposed redundancy structure.

Collision Avoidance using a Camera-based Point Cloud

This experiment focuses on the application of CA for a 7-DoF Panda robot from Franka Emika, represented in Fig. 4.15, operating in a dynamic environment. The scene is continuously monitored by a ZED 2i depth camera, which provides real-time 3D perception of both the robot and its surroundings. The camera is mounted in an elevated viewpoint and oriented to maximize the coverage of the robot movement, ensuring continuous visibility of the interaction region throughout the experiment. The camera produces a raw point cloud representing the entire workspace. Since the robot enters in the camera field of view, points belonging to the robot must be removed. We address this by applying a filtering pipeline that removes points corresponding to the robot's structure from the raw point cloud $\mathcal{P}$, using the link meshes $\mathcal{D}$ and their current world transformations $\mathcal{T}_l^w$. The mesh models are first converted into point clouds and transformed into the world frame to reconstruct the robot's shape at each time step. A bounding box is then computed around the resulting cloud to define a region of interest and a *CropBox* filter is applied to remove all scene points within it. To further refine the data, a combination of Statistical and Radius Outlier Removal (SROR) filters is applied to suppress sparse or inconsistent noise. During testing, we observed that residual artifacts can still appear: robot motion may



(a) Robot representation in the scene .

(b) Robot tracks the reference trajectory.

(c) Operator Hand appears in the camera frame the controller transitions from tracking to avoidance.

(d) Robot deviates from the trajectory.

(e) Operator hand disappears and Robot tracks the reference trajectory.

Figure 4.15. Camera-based collision avoidance scenarios for the Panda robot arranged in a 2×3 layout, with the RDF/point-cloud representation occupying the first column across two rows and each remaining cell split into camera view and filtered point-cloud view.

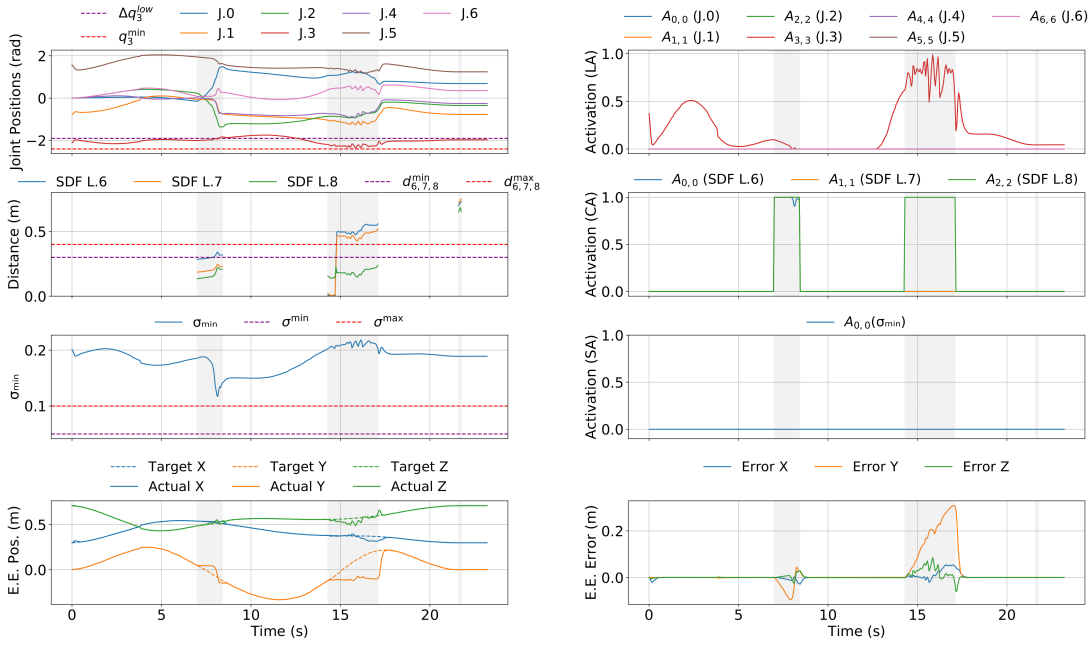


Figure 4.16. Collision avoidance scenarios with depth camera. Task-Priority activations: Joint Limit Avoidance (LA), Singularity Avoidance (SA), Collision Avoidance (CA), End-Effector (EE) task.

induce lighting changes and specular reflections that generate spurious points, often projected near or even onto the robot surface. To mitigate this effect, we leverage the robot SDF: points that lie on (or sufficiently close to) the robot's zero level set are classified as self-artifacts and removed. This prevents them from being mistakenly interpreted as external obstacles. Reflections can produce point-cloud clusters that resemble obstacles. In this example, the operator's hand is correctly detected and triggers collision avoidance at the gripper, whereas the reflected points are suppressed by the iso-surface filtering, avoiding false avoidance behaviors. As a result, only the right link is activated and the robot safely avoids the hand while remaining robust to reflective noise. The CA parameters are empirically determined. The activation thresholds $d_{\min} = 200$

mm and $d_{\max}$ = 400 mm, while the barrier limit $d_{\text{safe}} = 50$ mm. Regarding joint limits, the physical limits of the Panda robot are given by default: upper limits = [2.89, 1.76, 2.89, 0.069, 2.897, 3.75, 2.9] rad and lower limits = [-2.89, -1.76, -2.89, -2.4, -2.6, -0.017, -2.89] rad. For the chosen trajectory, the inverse kinematics solutions lie well within these bounds. However, to explicitly test the system's ability to exploit redundancy and respond to joint limit constraints, we artificially tightened the lower limit of joint 3 by setting $q^- = -2.4$ rad forcing $\Delta q^- = 0.3$ rad. The other joints remain constrained within their natural limits, all set with margins $\Delta q^{\pm} = \pm 0.1$ rad. For the SA we selected activation ranges deemed reasonable by monitoring the smallest singular value within a distance interval between 0.001 and 0.1.

Fig. 4.16 report only the last three robot links, i.e., those most frequently visible in the camera's field of view, while the analysis of SCA is deferred to a dedicated subsection. The proposed CA framework is evaluated in three representative scenarios, illustrated in Fig. 4.15b, 4.15c, and 4.15d. In these figures, blue points represent the robot's 3D point cloud, red points denote potential collision candidates (including both obstacles and spurious detections), and colored arrows on the robot surface indicate normalized repulsion gradients, highlighting the avoidance directions computed by the algorithm. The blue line rappresents a cubic spline trajectory for the end effector task (EE), but can be any trajectory task described in Sec. 4.3. In Fig. 4.16, the gray regions correspond to time intervals in which links enter in the camera's field of view, thereby triggering CA responses. In the first two scenarios shown in Fig. 4.15b and 4.15c, an operator's hand intrudes into the safety zone around the end effector. Consistent with the gray-shaded intervals in Fig. 4.16, the ECA task is promptly activated as soon as the hand is detected, generating repulsive gradients on the distal links (L6–L8) most exposed to the camera. The resulting corrective action forces the end effector to deviate smoothly from its nominal path. Once the obstacle retreats, the activations decay to zero and the trajectory reconverges to the reference, demonstrating the reactive behavior of the system.

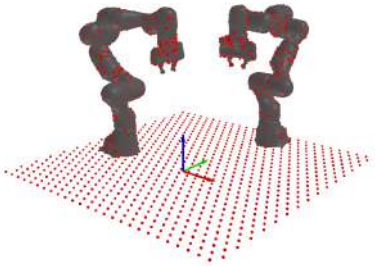
The second and third scenarios evaluate robustness to sensor noise, where spurious points from reflections may appear as obstacles. These artifacts are suppressed by iso-surface filtering and the activation matrix, preventing false avoidance. Consequently, the robot maintains its trajectory without disturbance, confirming that sdfs can be used to filter noise.

We further analyze the interaction between the CA module and other secondary tasks. In the SA case, the system tracks the desired trajectory without entering the activation region, and thus no interference with the avoidance behavior is observed. In contrast, the LA task imposes a hard constraint on the position of joint 3, directly influencing the system's redundancy resolution, as shown in Fig. 4.16. Indeed, the controller leverages kinematic redundancy to simultaneously satisfy the joint constraint and the CA objective.

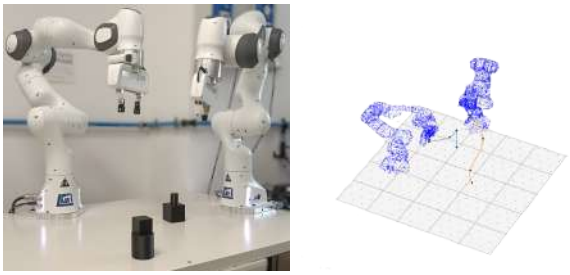
Double manipulator Pick and Place Task

This scenario involves setup composed of two Franka Emika Panda manipulators. This section aims to validate the method of multi robot end effector control introduced in Sec. 4.4.1, along with the CA and LA tasks described in Sec. 4.4.5 and Sec. 4.4.4, respectively. The SA task from Sec. 4.4.3 is also included, but not shown since it is not activated during the chosen task. The task requires the two robots to perform a cross pick-and-place operation, such that each robot must reach into the other's workspace. This naturally creates the need for mutual collision avoidance, as both arms must dynamically adapt their trajectories to prevent interference while completing their pick-and-place actions.

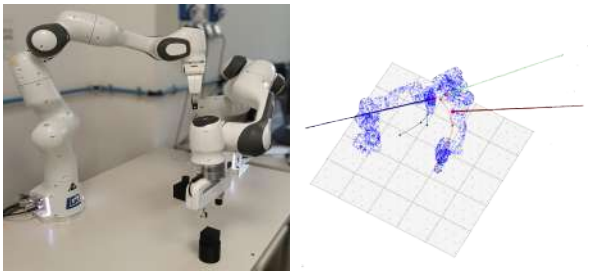
Figure 4.18 reports the evolution of the LA, CA, and EE tasks. The plots show that, while joint-limit activations occur, particularly for the elbow joints, the overall task remains feasible. This is achieved thanks to the redundancy resolution in the TP framework, which allows the robots to satisfy joint constraints while still prioritizing collision avoidance and end-effector motion. To maintain joints as centered as possible, the margins are set to $\Delta q = 0.7$ rad for both upper and lower bounds. The activation distances are uniform across all links for both ECA



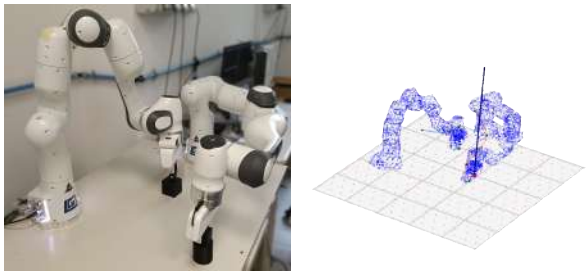
(a) Initial configuration of the two Panda manipulators.



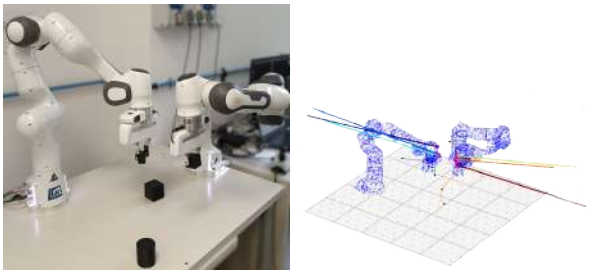
(b) Initial configuration of the two Panda manipulators.



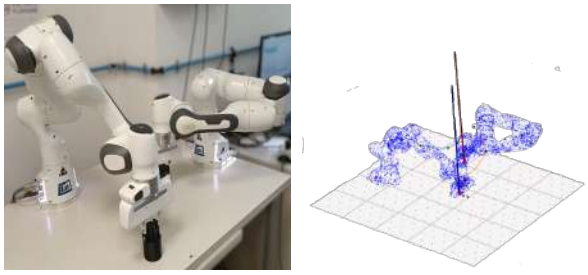
(c) Elbow–forearm proximity triggers self-collision avoidance.



(d) Ground proximity detected during the pick phase.



(e) Hand collision avoidance during the crossing motion.



(f) Final configuration — task completed.

Figure 4.17. Comparison between real-world execution (left images in each pair) and simulation (right images) for the dual-arm cooperative manipulation sequence. Each pair corresponds to a different task phase.

and SCA, with $d_{\text{min}} = 0.075$ m and $d_{\text{max}} = 0.11$ m, while a hard safety margin of $d_{\text{safe}} = 0.001$ m guarantees a minimal clearance at all times.

The evaluation is carried out within a unified TP framework, in which the two manipulators, as explained in Sec. 4.4.1, are modeled as a single composite system rather than totally independent robots. In this formulation, ECA is restricted to the ground plane represented as red points, whereas SCA encompasses both intra-robot and inter-robot interactions. In practice, each manipulator link is aware of other links in the chain as a potential collision source but also perceives the geometry of the other manipulator. Particular attention is paid to the distal links, which are the most likely to interfere during the crossing motion. For clarity, in the activation diagrams of Fig. 4.18, link 5 corresponds to the elbow, link 6 to the forearm, link 7 to the wrist, and link 8 to the gripper for both robots.

Figure 4.17 shows representative frames of the execution. During the crossing phase, potential collisions (e.g., robot A's gripper with robot B's elbow or forearm) trigger repulsive gradients that adjust robot B's posture while preserving feasibility. In the pick phase, activations on robot B's gripper remain small, so the end-effector task dominates. Finally, at $t \approx 20$ s in the place phase, close interactions at both grippers are again smoothly resolved, and both robots

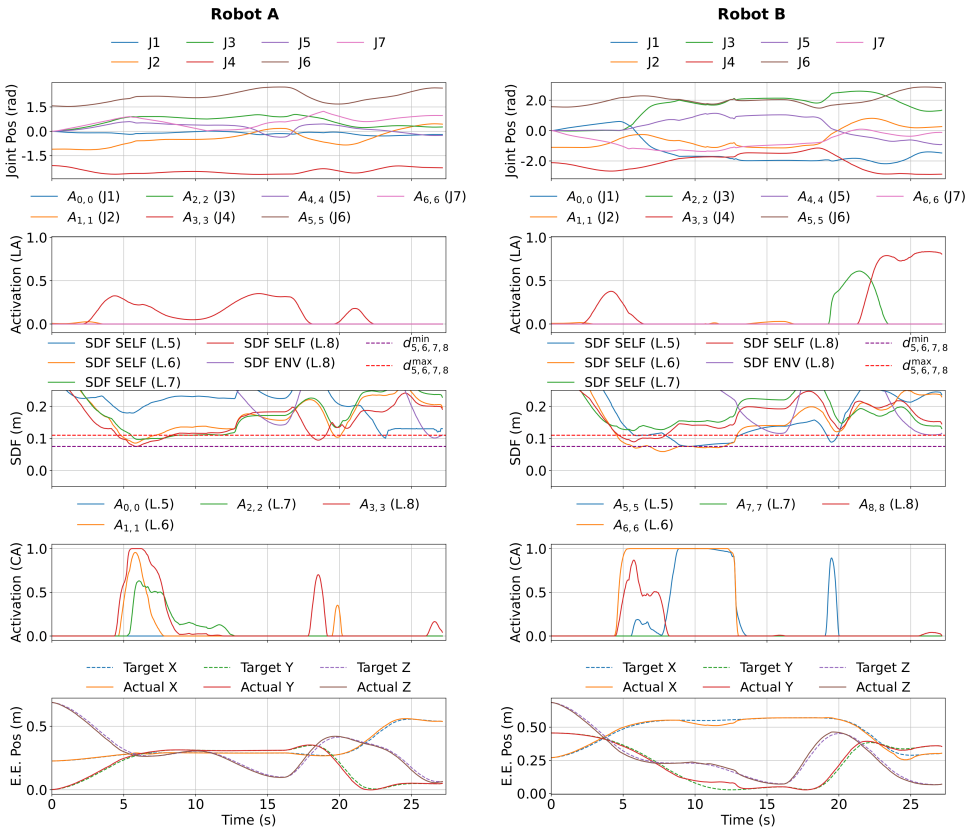


Figure 4.18. Task-Priority Activations in collision avoidance scenarios. The plot shows the activation levels of the tasks over time for both manipulators during the pick-and-place operation.

complete the task.

Overall, the plots in Fig. 4.18 together with the execution frames in Fig. 4.17 confirm that the proposed unified TP framework can successfully manage multi-arm coordination under shared collision constraints. The results highlight not only the alternating dominance between SCA and ECA but also the ability of the system to cope with joint-limit activations without compromising task feasibility.

Chapter 5

Conclusions

This dissertation consolidates advances across deformable object manipulation, real-world and simulated task scenarios, and multi-robot control. Leveraging GPU-accelerated computation, tactile sensing, and priority-based control, it addresses several open gaps in the state of the art. The contributions are validated in both simulation and hardware, using a broad sensor suite and show that embedding physical structure into the control stack yields robust performance under sensing uncertainty and hard safety constraints. The following sections summarize the addressed gaps and the resulting methods and systems, starting from DLO manipulation and concluding with a unified library for multi-robot control in unstructured environments.

DLO manipulation is frequently examined through idealized or purely algorithmic formulations, leaving a gap between theory and deployable industrial practice. In this work, several open issues left by prior research, particularly the application-oriented aspects of DLO manipulation, are addressed by closing the loop with tactile sensing (Ch. 1), which remains informative when vision degrades and predictive DLO models are uncertain.

In Sec.1.2, in-hand tactile feedback is used to robustly guide the connector insertion into its shielded cage. This task is brittle under vision-only pipelines since small misalignments or unexpected contacts can lead to jamming or misinsertion. By regulating robot motion with tactilely estimated cable pose and contact forces, the system can achieve high task success rates.

In Sec.1.3, tactile measurements are leveraged to estimate hose–environment interactions for compliant rubber tubes; contact forces that are too small to be reliably captured by a wrist-mounted F/T sensor become observable at the fingertip, enabling stable estimation and regulation.

Autonomous electrical wiring remains challenging due to deformability and cable occlusions; nevertheless, it remains an ambitious task in industrial automation. Beyond isolated pick-and-insert demonstrations, it is crucial to decide which cable to grasp, how to stage and route it, and when to insert under uncertainty. Sec.1.5 addresses this by introducing a cable-handling subsystem that captures and guides freshly extruded cables from production machinery into a structured storage module. This staging step standardizes initial conditions for wiring and serves as a practical bridge toward fully autonomous panel wiring.

A key gap concerns the interpretation of tactile signals. Prior studies often map sensor voltages to forces via auxiliary force sensors or adopt fully data-driven pipelines that require large datasets and offer limited physical guarantees. In this work, tactile feedback is structured through a Hertzian contact model, calibrated with a single probing experiment and Bayesian Optimization, yielding compact parameter identification without extensive data collection. The resulting estimates are physically interpretable, transferable across cable radii/materials within bounded residuals, and sufficiently stable under moderate rate effects.

In unstructured environments, Wire Arc Additive Manufacturing (WAAM) in open spaces poses two coupled challenges: sustaining geometric consistency while building large-scale structures, and enabling fabrication in hard-to-reach locations. WAAM suffers from deposition irregularities and evolving, uncertain geometry. Ch.2 addresses the challenge with a mobile,

climbing robot that reconfigures stance and ascends the printed artifact. Sensor fusion (motor-current readings as a proxy for contact load and an IMU) supports robust plug-in-hole anchoring and maintains torch-to-workpiece orientation with respect to the printing plane. Results show efficient, stable climbing and process-aware motion despite WAAM-induced uncertainties.

In unstructured environments, simulators provide a safe proving ground where new methods can be evaluated before deployment on hardware. Building on this premise, simulation becomes the engine of Sim2Real: first identify the plant and sensors, then randomize the parameters that matter, and finally close the loop with hardware-in-the-loop tests. The goal is not photorealism, but a sort of digital twin that captures the physical phenomena that matter for control. Ch. 3 presents a methodology that uses simulators to synthesize data for learning-based perception and control pipelines.

In Sec. 3.1, we develop a cable simulator to generate large-scale datasets. A naive approach to bridging the sim-to-real gap via parameter randomization led to unstable simulation behaviour; we analyze these failure modes and derive stability-aware randomization strategies.

Sec. 3.2 introduces a general real2sim2real framework that closes the loop between hardware and simulation. Using real data to identify simulation parameters, then randomizing only the sensitivity-dominant ones, enables the simulator to support the learning of robust control policies.

Beyond simulation fidelity, computer-graphics tools can efficiently represent geometry for motion generation and collision avoidance. Sec. 3.3 introduces Robotic Distance Functions (RDF): a differentiable signed distance representation parameterized by tensor-product Bernstein polynomials and accelerated on GPUs via CP (PARAFAC) low-rank decomposition.
Sec. 3.4 exploits kinematic constraints to build capability maps, coupling RDF-based collision checks with classical kinematics and GPU batching for high-throughput computation. This yields a library that collects geometric and kinematic data efficiently in a fast and scalable manner.

Finally, leveraging the tools of Sec. 3.3, Ch. 4 presents a unified framework for multi-robot control based on task prioritization. The resulting library executes real-time control of mobile manipulators and coordinated multi-arm systems with hard constraints for collision avoidance, joint limits, and singularity avoidance. We validate the framework in simulation and on hardware across multiple platforms, including dual-arm and mobile manipulation.

Bibliography

[1] Ismail T. Ahmed, Chen Soong Der, and Baraa Tareq Hammad. "A Survey of Recent Approaches on NoReference Image Quality Assessment with Multiscale Geometric Analysis Transforms". In: *International Journal of Scientific & Engineering Research* 7 (Dec. 2016), pp. 1146–1155. ISSN: 2229-5518.

[2] Jaume Albardaner et al. "Sim-to-Real gap in RL: Use Case with TIAGo and Isaac Sim/Gym". In: *arXiv preprint arXiv:2403.07091v2* (Mar. 2024). URL: https://arxiv.org/abs/2403.07091v2.

[3] Javier Alonso-Mora et al. "Collision avoidance for aerial vehicles in multi-agent scenarios". In: *Autonomous Robots* 39.1 (June 2015), pp. 101–121. ISSN: 1573-7527. DOI: 10.1007/s10514-015-9429-0. URL: https://doi.org/10.1007/s10514-015-9429-0.

[4] Marco Attene. "A lightweight approach to repairing digitized polygon meshes". In: *The Visual Computer* (2010). ISSN: 1432-2315. DOI: 10.1007/s00371-010-0416-3. URL: https://doi.org/10.1007/s00371-010-0416-3.

[5] Davide Bargellini et al. "Closing the Sim-to-Real Gap for Dynamics-Static Friction and Inertial Parameters: A Franka Robot Case Study". In: *European Robotics Forum 2024*. Ed. by Cristian Secchi and Lorenzo Marconi. Cham: Springer Nature Switzerland, 2024, pp. 333–337. ISBN: 978-3-031-76424-0.

[6] Wendwosen Bellete Bedada and Gianluca Palli. *Real Time Collision Avoidance with GPU-Computed Distance Maps*. 2024. arXiv: 2407.02363 [cs.RO]. URL: https://arxiv.org/abs/2407.02363.

[7] Alexey I. Boyko et al. "TT-TSDF: Memory-Efficient TSDF with Low-Rank Tensor Train Decomposition". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10116–10121. DOI: 10.1109/IROS45743.2020.9341464.

[8] Jonathan Cacace, Fabio Ruggiero, and Vincenzo Lippiello. "Hierarchical Task-Priority Control for Human-Robot Co-manipulation". In: *Human-Friendly Robotics 2019*. Ed. by Federica Ferraguti et al. Cham: Springer International Publishing, 2020, pp. 125–138. ISBN: 978-3-030-42026-0.

[9] Alessio Caporali et al. "Deformable Linear Objects Manipulation With Online Model Parameters Estimation". In: *IEEE Robotics and Automation Letters* 9.3 (2024), pp. 2598–2605.

[10] Alessio Caporali et al. "RT-DLO: Real-time deformable linear objects instance segmentation". In: *IEEE Transactions on Industrial Informatics* 19.11 (2023), pp. 11333–11342.

[11] Patryk Cieślak et al. "Practical formulation of obstacle avoidance in the Task-Priority framework for use in robotic inspection and intervention scenarios". In: *Robotics and Autonomous Systems* (2020). ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2019.103396. URL: https://www.sciencedirect.com/science/article/pii/S0921889019306104.

[12] A. Cirillo et al. "Modeling and Calibration of a Tactile Sensor for Robust Grasping". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 6843–6850. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.1205. URL: https://www.sciencedirect.com/science/article/pii/S2405896317317111.

[13] Andrea Cirillo et al. "Tactile Sensors for Parallel Grippers: Design and Characterization". In: *Sensors* 21.5 (2021). ISSN: 1424-8220. DOI: 10.3390/s21051915. URL: https://www.mdpi.com/1424-8220/21/5/1915.

[14] D. Eberly. *Perspective Projection of an Ellipsoid*. https://www.geometrictools.com/Documentation/PerspectiveProjectionEllipsoid.pdf. Last revised November 12, 2013. 1999. URL: https://www.geometrictools.com/Documentation/PerspectiveProjectionEllipsoi pdf.

[15] D. De Gregorio et al. "Integration of robotic vision and tactile sensing for wire-terminal insertion tasks". In: *IEEE Transactions on Automation Science and Engineering* 16.2 (2019), pp. 585–598.

[16] A. De Luca, G. Oriolo, and P.R. Giordano. "Kinematic modeling and redundancy resolution for nonholonomic mobile manipulators". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* 2006, pp. 1867–1873. DOI: 10.1109/ROBOT.2006.1641978.

[17] G. De Maria, C. Natale, and S. Pirozzi. "Force/tactile sensor for robotic applications". In: *Sensors and Actuators A: Physical* 175 (2012), pp. 60–72. ISSN: 0924-4247. DOI: https://doi.org/10.1016/j.sna.2011.12.042. URL: https://www.sciencedirect.com/science/article/pii/S0924424711007564.

[18] Rosen Diankov. "Automated Construction of Robotic Manipulation Programs". PhD thesis. Pittsburgh, PA: Carnegie Mellon University, Sept. 2010.

[19] Danny Driess et al. *Learning Models as Functionals of Signed-Distance Fields for Manipulation Planning*. 2021. arXiv: 2110.00792 [cs.RO]. URL: https://arxiv.org/abs/2110.00792.

[20] Yuqing Du et al. "Auto-Tuned Sim-to-Real Transfer". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1290–1296. DOI: 10.1109/ICRA48506.2021.9562091.

[21] David Eberly. *Distance from a Point to an Ellipse, an Ellipsoid, or a Hyperellipsoid*. https://www.geometrictools.com/Documentation/DistancePointEllipseEllipsoidHyperellipsoid.pdf. Geometric Tools, version 1.0. 2020.

[22] Mark Nicholas Finean, Wolfgang Merkt, and Ioannis Havoutis. *Predicted Composite Signed-Distance Fields for Real-Time Motion Planning in Dynamic Environments*. 2021. arXiv: 2008.00969 [cs.RO]. URL: https://arxiv.org/abs/2008.00969.

[23] Rukhsar Firdousi and Shaheen Parveen. "Local Thresholding Techniques in Image Binarization". In: *International Journal Of Engineering And Computer Science* 3 (Mar. 2014), pp. 4062–4065.

[24] Jeremy A. Fishel and Gerald E. Loeb. "Sensing tactile microvibrations with the BioTac — Comparison with human sensitivity". In: *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. 2012, pp. 1122–1127. DOI: 10.1109/BioRob.2012.6290741.

[25] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML]. URL: https://arxiv.org/abs/1807.02811.

[26] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *International Conference on Machine Learning (ICML)*. 2018, pp. 1587–1596.

[27] Kevin Galassi et al. "Scalable Shared Encoding Architecture for Learning-Based Error Detection in Robotic Wiring Harness Assembly". In: *2024 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2024.

[28] P. Gamage and S.Q. Xie. "A real-time vision system for defect inspection in cast extrusion manufacturing process". In: *Int J Adv Manuf Technol* 40 (2009), pp. 144–156. DOI: 10.1007/s00170-007-1326-z.

[29] Claudio Gaz et al. "Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization". In: *IEEE Robotics and Automation Letters* PP (July 2019), pp. 1–1. DOI: 10.1109/LRA.2019.2931248.

[30] Javier González Huarte, Maite Ortiz de Zarate, and Aitor Ibarguren. "CAD-Based Robot Programming Solution for Wire Harness Manufacturing in Aeronautic Sector". In: *Robotics* 12.5 (2023). DOI: 10.3390/robotics12050130.

[31] A. Govoni et al. "Parallel Gripper Cable Manipulation through Tactiles and Hertzian Modeling". In: *Under Submission on IEEE Robotics and Automation Letters*. Under submission. IEEE, pp. 1–8.

[32] Andrea Govoni et al. "A Robotic System for Medical Hoses Manipulation and Quality Check". In: *IEEE Transactions on Automation Science and Engineering* (2025).

[33] Andrea Govoni et al. "Design and Integration of a Robotic Gripper and Warehouse System for Automated Cable Assembly". In: *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*. 2025, pp. 596–601. DOI: 10.1109/CASE58245.2025.11163907.

[34] Andrea Govoni et al. "Performance Analysis of a Mass-Spring-Damper Deformable Linear Object Model in Robotic Simulation Frameworks". In: *European Robotics Forum 2025*. Ed. by Marco Huber, Alexander Verl, and Werner Kraus. Cham: Springer Nature Switzerland, 2025, pp. 187–192. DOI: 10.1007/978-3-031-89471-8_29.

[35] Andrea Govoni et al. "Towards the Automation of Wire Harness Manufacturing: A Robotic Manipulator with Sensorized Fingers". In: *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. 2023, pp. 974–979. DOI: 10.1109/CoDIT58514.2023.10284109.

[36] Amos Gropp et al. "Implicit geometric regularization for learning shapes". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2020, pp. 3789–3799.

[37] Chun Guo et al. "Research Progress and Application Scenarios of Wire Arc Additive Manufacturing". In: *Micromachines* (2025). URL: https://www.mdpi.com/2072-666X/16/7/749.

[38] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning (ICML)*. 2018, pp. 1861–1870.

[39] I. Ibraheem and A. Binder. "An automated inspection system for stents". In: *Int J Adv Manuf Technol* 47 (2010), pp. 945–951. DOI: 10.1007/s00170-009-2133-5.

[40] Nick Jakobi, Phil Husbands, and Inman Harvey. ""Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics,"" in: vol. 929. Jan. 1995, pp. 704–720.

[41] Nick Jakobi, Phil Husbands, and Inman Harvey. "Noise and the reality gap: The use of simulation in evolutionary robotics". In: *Advances in artificial life: Third European conference on artificial life granada, Spain, june 4–6, 1995 proceedings 3*. Springer. 1995, pp. 704–720.

[42] Mojtaba Karamimoghadam et al. ""Wire-arc additive manufacturing of low alloyed steels, copper aluminum alloys, and their bimetals"". In: *International Journal of Advanced Manufacturing Technology* (2025). URL: https://link.springer.com/article/10.1007/s00170-025-09123-3.

[43] Peter Kaufmann et al. “Flexible simulation of deformable models using discontinuous Galerkin FEM”. In: *Graphical Models* 71.4 (2009), pp. 153–167.

[44] Henrik Keller, Peter Kuhn, and Dirk Meiners. *Method of Constructing a Tower*. US Patent 2017/0016244 A1. 2017.

[45] Aaron Kemmer, Michael Snyder, and Michael Chen. “Extended Structures of Additive Manufacturing”. US 2015/0076732 A1. Publication Date: March 19, 2015. Mar. 2015.

[46] Marian Kleineberg. *mesh_to_sdf: Calculate signed distance fields for arbitrary meshes*. GitHub repository. 2021. URL: https://github.com/marian42/mesh_to_sdf.

[47] Mikhail Koptev, Nadia Figueroa, and Aude Billard. “Neural Joint Space Implicit Signed Distance Functions for Reactive Robot Manipulator Control”. In: *IEEE Robotics and Automation Letters* (2022). DOI: 10.1109/LRA.2022.3227860.

[48] Alexey Kuznetsov et al. “Truncated Quantile Critics: Improving Quantile Regression for Distributional Reinforcement Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[49] Riddhiman Laha et al. “S*: On Safe and Time Efficient Robot Motion Planning”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 12758–12764. DOI: 10.1109/ICRA48891.2023.10161248.

[50] Scott Lahteine et al. *Marlin*. 2021.

[51] Gianluca Laudante et al. “Mechatronic Integration of a Dual-Arm Robotic System for Wiring Harness Manufacturing”. In: *IEEE/ASME Transactions on Mechatronics* (2025), pp. 1–12. DOI: 10.1109/TMECH.2025.3536627.

[52] J. Lechuz-Sierra and et al. “Bayesian Optimization for Grasp Calibration with Tactile Feedback”. In: *Proc. of ICRA*. 2024.

[53] Dingquan Li and Tingting Jiang. “Blur-Specific No-Reference Image Quality Assessment: A Classification and Review of Representative Methods”. In: *2007 International Conference on Field-Programmable Technology*. Jan. 2019, pp. 45–68. DOI: DOI:10.1007/978-3-319-91659-0_4.

[54] Yiming Li et al. *Configuration Space Distance Fields for Manipulation Planning*. 2024. arXiv: 2406.01137 [cs.RO]. URL: https://arxiv.org/abs/2406.01137.

[55] Yiming Li et al. “Representing Robot Geometry as Distance Fields: Applications to Whole-body Manipulation”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 15351–15357. DOI: 10.1109/ICRA57147.2024.10611674.

[56] Timothy P. Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *International Conference on Learning Representations (ICLR)*. 2016.

[57] Xi Lin et al. “Curvature sensing with a spherical tactile sensor using the color-interference of a marker array”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 603–609. DOI: 10.1109/ICRA40945.2020.9197050.

[58] Puze Liu et al. *Regularized Deep Signed Distance Fields for Reactive Motion Generation*. 2022. arXiv: 2203.04739 [cs.RO]. URL: https://arxiv.org/abs/2203.04739.

[59] Naijing Lv et al. “Physically based real-time interactive assembly simulation of cable harness”. In: *Journal of Manufacturing Systems* 43 (2017). Special Issue on the 12th International Conference on Frontiers of Design and Manufacturing, pp. 385–399. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2017.02.001. URL: https://www.sciencedirect.com/science/article/pii/S0278612517300146.

[60] Ning Lv et al. “A review of techniques for modeling flexible cables”. In: *Computer-Aided Design* 122 (2020), p. 102826. DOI: 10.1016/j.cad.2020.102826.

[61] Abhijit Makhal and Alex K. Goins. "Reuleaux: Robot Base Placement by Reachability Analysis". In: *2018 Second IEEE International Conference on Robotic Computing (IRC)*. 2018, pp. 137–142. DOI: 10.1109/IRC.2018.00028.

[62] Pablo Malvido Fresnillo et al. "An Approach for the Automatic Assembly of Multi-Branch Wire Harnesses in the Automotive Industry Using a Dual-Arm Robot". In: *SSRN Electronic Journal* (2024).

[63] N. Markiz, E. Horváth, and P Ficzere. "Influence of printing direction on 3D printed ABS specimens". In: *Production Engineering Archives* 26.3 (2020), pp. 127–130. DOI: 10.30657/pea.2020.26.24.

[64] Nicolas Marticorena et al. *RMMI: Enhanced Obstacle Avoidance for Reactive Mobile Manipulation using an Implicit Neural Map*. 2024. arXiv: 2408.16206 [cs.RO]. URL: https://arxiv.org/abs/2408.16206.

[65] Yecheng Mei et al. "Robotic Assembly Strategy With Wrist Force Sense for Narrow Clearance Peg-in-Hole". In: *IEEE Transactions on Automation Science and Engineering* 22 (2025), pp. 8709–8724. DOI: 10.1109/TASE.2024.3487988.

[66] Andrea Monguzzi et al. "Tactile based robotic skills for cable routing operations". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 3793–3799. DOI: 10.1109/ICRA48891.2023.10160729.

[67] *MoveIt 2: An Open Source Robot Manipulation Framework*. https://moveit.picknik.ai. 2024.

[68] K. Navaneetha et al. "Modeling of two link manipulator used for 3D printing". In: *IOP Conference Series: Materials Science and Engineering* 1057.1 (2021), p. 012023. DOI: 10.1088/1757-8395/1057/1/012023.

[69] Huong Giang Nguyen, Marlene Kuhn, and Jörg Franke. "Manufacturing automation for automotive wiring harnesses". In: *Procedia CIRP* 97 (2020), pp. 379–384. DOI: 10.1016/j.procir.2020.05.254.

[70] C. Nogueira and et al. "Safe Bayesian Optimization for Tactile Robot Control". In: *Robotics and Autonomous Systems* (2024).

[71] Stefan Olbrich and Julia Lackinger. "Manufacturing Processes of automotive high-voltage wire harnesses: State of the art, current challenges and fields of action to reach a higher level of automation". In: *Procedia CIRP* 107 (2022), pp. 653–660. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2022.05.041.

[72] Kalavathi Palanisamy. "A Thresholding Method for Color Image Binarization". In: *International Journal on Computer Science and Engineering* 1 (Sept. 2014), pp.31–40. DOI: https://doi.org/10.14445/23488387/IJCSE-V1I7P107.

[73] Michele Palermo, Tomaso Trombetti, and Vittoria Laghi. "Stampante tridimensionale, relativo procedimento per la realizzazione di un oggetto stampato ed oggetto così ottenuto". IT102020000024109. 2022.

[74] Jeong Joon Park et al. *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*. 2019. arXiv: 1901.05103 [cs.CV]. URL: https://arxiv.org/abs/1901.05103.

[75] Alex Pasquali, Kevin Galassi, and Gianluca Palli. "A Fast Score-Based Method for Robotic Task-Free Point-to-Point Path Learning". In: *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2023, pp. 1159–1164. DOI: 10.1109/AIM46323.2023.10196238.

[76] Xue Bin Peng et al. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3803–3810. DOI: 10.1109/ICRA.2018.8460528.

[77] Anastasia Puzatova et al. "Large-Scale 3D Printing for Construction Application by Means of Robotic Arm and Gantry 3D Printer: A Review". In: *Buildings* 12.11 (2022). ISSN: 2075-5309. DOI: 10.3390/buildings12112023.

[78] Charles R. Qi et al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV].

[79] Gulnaaz Rasiya, Abhinav Shukla, and Karan Saran. "Additive Manufacturing-A Review". In: *Materials Today: Proceedings* 47 (2021). International Conference on Advances in Design, Materials and Manufacturing, pp. 6896–6901. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2021.05.181.

[80] V. Román Ibáñez et al. "Collaborative robotics in wire harnesses spot taping process". In: *Computers in Industry* 125 (2021), p. 103370. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2020.103370.

[81] Erica Salvato et al. "Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning". In: *IEEE Access* 9 (2021), pp. 153171–153187. DOI: 10.1109/ACCESS.2021.3126658.

[82] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[83] Aidar Shakerimov, Tohid Alizadeh, and Huseyin Atakan Varol. "Efficient Sim-to-Real Transfer in Reinforcement Learning Through Domain Randomization and Domain Adaptation". In: *IEEE Access* 11 (2023), pp. 136809–136823. DOI: 10.1109/ACCESS.2023.3339568. URL: https://ieeexplore.ieee.org/document/10397579.

[84] Yu She et al. "Cable manipulation with a tactile-reactive gripper". In: *International Journal of Robotics Research* 40.12-14 (2021), pp. 1385–1401. DOI: 10.1177/02783649211027233.

[85] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. London: Springer, 2009.

[86] Enrico Simetti and Giuseppe Casalino. "A Novel Practical Technique to Integrate Inequality Control Objectives and Task Transitions in Priority Based Control". In: *Journal of Intelligent & Robotic Systems* (Dec. 2016). DOI: 10.1007/s10846-016-0368-6.

[87] Sudhanshu Ranjan Singh and Pradeep Khanna. ""Wire arc additive manufacturing (WAAM): A new process to shape engineering materials"". In: *Materials Today: Proceedings* (2021). URL: https://www.sciencedirect.com/science/article/pii/S2214785320358922.

[88] Gabriele Tiboni, Karol Arndt, and Ville Kyrki. "DROPO: Sim-to-real transfer with offline domain randomization". In: *Robotics and Autonomous Systems* 166 (2023), p. 104432. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2023.104432. URL: https://www.sciencedirect.com/science/article/pii/S0921889023000714.

[89] A. Tiwari and B. N. J. Persson. "Cylinder-Flat Contact Mechanics with Surface Roughness". In: *Tribology Letters* 69.1 (2020), p. 4. ISSN: 1573-2711. DOI: 10.1007/s11249-020-01380-z.

[90] J. Trommnau et al. "Overview of the state of the art in the production process of automotive wire harnesses, current research and future trends". In: *Procedia CIRP* 81 (2019), pp. 387–392. DOI: 10.1016/j.procir.2019.03.067.

[91] Miguel A. Trujillo et al. "Priority Task-Based Formation Control and Obstacle Avoidance of Holonomic Agents with Continuous Control Inputs". In: *IFAC-PapersOnLine* (2018). 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.07.281. URL: https://www.sciencedirect.com/science/article/pii/S2405896318310346.

[92] Nikolaus Vahrenkamp et al. "Manipulability analysis". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 2012, pp. 568–573. DOI: 10.1109/HUMANOIDS.2012.6651576.

[93] Hristo Vrigazov. *modern_robotics_torch: Implementation of modern robotics Kinematics Engine in PyTorch*. https://github.com/hristo-vrigazov/modern_robotics_torch. Version and access date to be added. 2025.

[94] Hristo Vrigazov and Kaloyan Yovchev. "Parallelized Forward Kinematics Using Product of Exponentials in PyTorch". In: *International Conference on Robotics in Alpe-Adria Danube Region*. Springer. 2022, pp. 44–51. DOI: 10.1007/978-3-031-04870-8_6.

[95] Yanzhao Wu and Ling Liu. "Selecting and composing learning rate policies for deep neural networks". In: *ACM Transactions on Intelligent Systems and Technology* 14.2 (2023), pp. 1–25.

[96] Yi Wu et al. "A real-time collision avoidance method for redundant dual-arm robots in an open operational environment". In: *Robotics and Computer-Integrated Manufacturing* 92 (2025), p. 102894. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2024.102894. URL: https://www.sciencedirect.com/science/article/pii/S0736584524001819.

[97] Lixia Yan, Baoli Ma, and Yingmin Jia. "Trajectory Tracking Control of Nonholonomic Wheeled Mobile Robots Using Only Measurements for Position and Velocity". In: *Automatica* 159 (2024), p. 111374. DOI: 10.1016/j.automatica.2023.111374.

[98] Yiming Yang et al. "iDRM: Humanoid motion planning with realtime end-pose selection in complex environments". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 271–278. DOI: 10.1109/HUMANOIDS.2016.7803288.

[99] Yuxuan Yang, Johannes A. Stork, and Todor Stoyanov. "Learning differentiable dynamics models for shape control of deformable linear objects". In: *Robotics and Autonomous Systems* 158 (2022), p. 104258. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2022.104258. URL: https://www.sciencedirect.com/science/article/pii/S0921889022001518.

[100] Wenzhen Yuan, Siyuan Dong, and Edward Adelson. "GelSight: High-resolution robot tactile sensors for estimating geometry and force". In: *Sensors* 17.12 (2017), p. 2762.

[101] Wenzhen Yuan et al. "Measurement of shear and slip with a GelSight tactile sensor". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 304–311. DOI: 10.1109/ICRA.2015.7139016.

[102] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. "Capturing robot workspace structure: representing robot capabilities". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 3229–3236. DOI: 10.1109/IROS.2007.4399105.

[103] Franziska Zacharias et al. "Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories". In: *2009 9th IEEE-RAS International Conference on Humanoid Robots*. 2009, pp. 55–61. DOI: 10.1109/ICHR.2009.5379601.

[104] Hengshuang Zhang et al. "CenterNet: Keypoint Networks for Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46 (May 2024), pp. 3509–3521. DOI: 10.1109/TPAMI.2023.3342120.

[105] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey". In: *arXiv preprint arXiv:2009.13303* (Sept. 2020). DOI: 10.48550/arXiv.2009.13303. URL: https://arxiv.org/abs/2009.13303.