POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering



Master's Degree Thesis

Coordinated Control of a Bimanual Soft Continuum Robot

Supervisors

prof. Diego Regruto Tomalino
prof. Josie Hughes

Candidate

Salvatore Loizzo

Academic Year 2025/2026

To my family, thanks for everything

Summary

Soft continuum manipulators offer compliance and adaptability for delicate grasping: they can bend through confined and unstructured spaces that rigid arms cannot reach, and unexpected contact is absorbed by the structure rather than resisted, making interaction inherently safe. These properties are rarely exploited in coordinated manipulation, which requires regulating motion and contact forces at the same time. This thesis develops a unified Virtual Model Control framework, based on impedance control, for a fully tendon-driven bimanual soft continuum platform, whose structure was designed, 3D-printed and assembled as part of this work. Building on a system identification of the arms' structural stiffness and damping, the approach regulates whole-body motion and contact forces through model-based compliant parameters, with adaptation bounded by a passivity-preserving energy tank. The framework is validated experimentally on both arms: the reachable workspace is measured directly, and trajectory tracking is evaluated along circular, figure-eight and polygonal paths, together with the step response. Contact-force regulation is then characterized per arm against a load cell, in open and closed loop, identifying the tendon transmission efficiency. Finally, coordinated bimanual operation is demonstrated through whole-body pick-and-place, without custom end-effectors.

Acknowledgements

I would like to express my deepest gratitude to my daily advisors at EPFL, Lorenzo Vignoli and Kyle Walker, who interviewed me and gave me the opportunity to join the CREATE Lab. Thank you for acting as my co-supervisors and for constantly supporting me throughout my thesis work. I loved the lab, always open-minded to any kind of project and idea. A special thanks to Professor Josie Hughes, Head of the lab, whose guidance was invaluable, and to Professor Diego Regruto Tomalino for his supervision.

Thanks to my Mother, my Father, and Nicola. You taught me the true meaning of family, each of you from a different perspective. I am incredibly grateful: you are part of who I am.

Contents

List of Tables 10

List of Figures 11

Nomenclature 14

I Introduction and Background **17**

1 Introduction 19
- 1.1 Robotics . . . 19
- 1.2 Rigid vs Soft Robotics . . . 20
- 1.3 Control . . . 21
- 1.4 Goal of the thesis . . . 21
- 1.5 Main contributions . . . 21

2 State of the Art 23
- 2.1 Soft arms . . . 23
- 2.2 Actuation system . . . 24
- 2.3 Bimanual continuum platforms . . . 25

II Theoretical Framework **27**

3 State Parametrization 29
- 3.1 PCC Model . . . 29
- 3.2 Forward Kinematics . . . 32
- 3.3 Differential Kinematics and Jacobian Stability . . . 33

4 Virtual Model Control: Background 35
- 4.1 Impedance Control and Task-Space Compliance . . . 35
- 4.2 Virtual Model Control (VMC) . . . 36
- 4.3 VMC for Soft Continuum Robots . . . 36

III System Development 39

5 System Architecture and Project Workflow 41

6 Hardware 45
6.1 Motors . . . 45
6.1.1 Mechanical characteristics . . . 45
6.1.2 Communication and control modalities . . . 46
6.1.3 Dynamixel Wizard . . . 47
6.1.4 C++ Node Implementation . . . 47
6.2 IMUs . . . 49
6.2.1 Electronic circuit . . . 49
6.2.2 Arduino code . . . 50
6.2.3 Soldering . . . 51

7 Mechanical Design 53
7.1 CAD and 3D Printing . . . 53
7.1.1 Soft Arms . . . 53
7.1.2 Chassis . . . 55
7.1.3 Assembly . . . 56

8 Kinematics Implementation 59
8.1 Tendon-to-Configuration Mapping for a Three-Tendon Section . . . 59
8.2 Forward Kinematics and Arc-Length Parametrization . . . 61
8.3 Differential Kinematics and Computational Implementation . . . 62
8.3.1 Actuator to Configuration Space Mapping . . . 62
8.3.2 Configuration to Task Space Mapping . . . 63

9 State Estimation 65
9.1 Sensors . . . 65
9.1.1 Encoders . . . 65
9.1.2 Motion Capture . . . 66
9.2 Kalman Filter . . . 68
9.2.1 General Kalman Filter Algorithm . . . 68
9.2.2 Filter Tuning and Covariances . . . 68
9.2.3 Prediction Model . . . 69
9.2.4 Initialization, Correction, and Innovation Gate . . . 70

10 Virtual Model Control Implementation 71
10.1 From a virtual element to motor torques . . . 71
10.2 Torque composition . . . 72
10.2.1 Gravity compensation . . . 72
10.2.2 Pretensioning and the unilateral constraint . . . 73
10.2.3 Saturation . . . 73

10.3 Default placement and gains . . . 73
10.4 Library of virtual elements . . . 74
10.4.1 Force laws . . . 74
10.4.2 Constrained elements . . . 75
10.5 Smooth transitions . . . 76
10.5.1 Moving goal filter . . . 76
10.5.2 Gain ramping . . . 76
10.5.3 Bumpless transfer . . . 77
10.6 Graphical user interface . . . 77
10.6.1 Force modulation panel . . . 79
10.7 Passivity of the fixed-gain controller . . . 80

11 Force modulation and passivity 83
11.1 Task-space force under contact . . . 83
11.1.1 The problem, and why it is not a sensing problem . . . 83
11.1.2 Three power flows . . . 84
11.1.3 From power balance to force balance . . . 84
11.1.4 Specializing to static contact . . . 85
11.1.5 Inverting a non-square map . . . 85
11.2 Adapting the virtual model . . . 85
11.2.1 Gradient descent on the virtual parameters . . . 85
11.2.2 Reference modulation . . . 86
11.2.3 Stiffness modulation . . . 86
11.2.4 Choosing between the strategies . . . 87
11.3 Energy accounting and the tank . . . 87
11.4 Bounding the step size . . . 88
11.4.1 The passivity bound . . . 88
11.4.2 Two further bounds . . . 89
11.4.3 The applied step . . . 89
11.5 Implementation . . . 90
11.6 What the guarantee covers . . . 91

IV Experimental Validation **93**

12 Experimental Overview 95
12.1 Common setup . . . 95

13 System Identification 97
13.1 Experimental Protocol . . . 97
13.2 Dynamic Model and Least-Squares Regression . . . 98
13.3 Results and Model Selection . . . 98

14 Force control 105

14.1 η calibration . . . 105
14.1.1 What is being fitted . . . 105
14.1.2 Protocol . . . 106
14.1.3 Results . . . 107
14.2 Open-loop force tracking . . . 107
14.2.1 Protocol . . . 108
14.2.2 Results . . . 108
14.3 Closed-loop force tracking . . . 112
14.3.1 What changes . . . 112
14.3.2 Results . . . 112
14.3.3 Why the descent needs a deadband . . . 113

15 Position control 115
15.1 The measured workspace . . . 115
15.2 Step response . . . 117
15.3 Trajectory tracking . . . 119

16 Conclusion 125

List of Tables

6.1 Main characteristics of the DYNAMIXEL XC330-T288-T and of the bus configuration adopted on this platform. 46
7.1 Optimized 3D printing parameters for TPU soft modules on the Bambu Lab X1-Carbon. 54
8.1 Tendon routing angles for each continuum section. 61
9.1 Default covariances of the state estimator. 69
10.1 Default stiffness and damping assigned to a virtual element according to its anchor point. 74
10.2 Implemented spring and damper laws 75
10.3 Main interface topics. The main controller constantly republishes its active configuration so that the GUI dynamically rebuilds itself around the running layout. 78
11.1 Force-modulation parameters. All are adjustable at runtime from the interface; η is the value identified against the load cell in Section 14.1. . . 90
13.1 Commanded step torque amplitudes applied above the 0.03 N m baseline. 97
13.2 Identified stiffness and damping parameters for the right arm, comparing encoder and MoCap datasets. 99
13.3 The two arms compared on the MoCap identification. 104
14.1 Closed-loop force tracking (Figure 14.6), normal component. The error is the mean over the last two seconds of each level; the small negative bias is the tangential share of the magnitude the loop regulates. The reference travel at 1.5 N is the recovery from the slip event. 113
14.2 Steady-state error against the load cell, open loop (the two runs of Figures 14.3 and 14.5) and closed loop (Figure 14.6). Normal component. The open-loop error nearly quadruples above 2 N. 113
15.1 Step response averaged over 24 steps per gain setting. The steady-state error is taken on the most demanding targets, at 200 and 300 mm from the rest pose. 117

List of Figures

1.1 Biological structures inspiring soft and continuum robotics. 20
2.1 Geometric parameters defining the Trimmed Helicoid (TH) structure. Adapted from [Guan et al., 2023]. 24
2.2 Comparison between coupled and decoupled tendon actuation. Adapted from [Guan et al., 2023]. 24
3.1 Piecewise Constant Curvature model of a soft robotic arm. Adapted from [Della Santina et al., 2020a]. 30
3.2 Kinematic singularities of the classical parametrization 30
3.3 Geometry of the proposed state parametrization 31
4.1 Spring Flamingo robot 36
4.2 Graphical representation of VMC on a soft arm 37
5.1 System architecture 42
6.1 Dynamixel XC330-T288-T used for the project 45
6.2 Nested control loops of the Dynamixel servos 47
6.3 Electronic prototyping setup and the main components of the IMU acquisition system. 50
6.4 Soldering setup and final hardware verification. 51
7.1 Detailed CAD view of a single soft module highlighting the routing holes. 55
7.2 CAD models of the rigid connecting links, including the IMU housing version and the Bowden cable sockets. 55
7.3 Detailed CAD view of the bimanual chassis, illustrating the dual-layer motor arrangement and Bowden cable routing. 56
7.4 Key stages of the physical assembly process. 57
7.5 Final CAD render (left) and the physically assembled bimanual robot (right) 57
8.1 Geometric projection of the tendons onto the bending direction 60
8.2 The world frame and the two base frames on the platform on CAD. The bases differ only by the $\pm 125\,\mathrm{mm}$ offset along x; their axes are parallel, so $R_{wb} = I$. 62
9.1 The Motive Software during a session 66
10.1 Graphical user interface featuring a real-time digital twin of the bimanual system, per-element gain sliders, a control-point spatial editor, and a feedback-source selector. 78

10.2 The interface with the force-modulation panel enabled. Left: the standard control column and per-element gains. Centre: the digital twin of both arms. Right: the force-modulation strip, here showing the contact arm, the descent mode, the commanded force and normal, and the calibration group. 80
12.1 The load-cell rig used throughout this part, shown without the arm: two single-axis cells on orthogonal guides resolve the normal and the tangential component of the contact force. During the experiments the tip of the right arm presses on the plate, and the rig serves as ground truth for every force reported in the following chapters. 96
13.1 Measured and simulated PCC curvature strain during the bending excitation, right arm. Section 1 is the proximal one. 100
13.2 Measured and simulated axial compression δL during the compression excitation, right arm. 101
13.3 Measured and simulated PCC curvature strain during the bending excitation, left arm. 102
13.4 Measured and simulated axial compression δL during the compression excitation, left arm. 103
14.1 Experimental setup 106
14.2 Left: the regression of Equation (14.1). Each point is one settled plateau; the dashed line is the fit pooled over all three poses. Right: the factor fitted independently for each pose, displaying the mean and its spread. . . 107
14.3 Open-loop force tracking, reference modulation 109
14.4 Open-loop tracking with the stiffness carrying the adaptation 110
14.5 Open-loop tracking, both channels free 111
14.6 Closed-loop tracking of three force levels. Top: load cell against target, with the detected slip event shaded. Middle: the reference the descent is moving. Bottom: the virtual stiffness, held constant — this mode adapts the reference only. 112
15.1 Measured workspace of both arms 116
15.2 Workspace test - bending of the left arm. 116
15.3 Step response of Table 15.1 117
15.4 Outward steps on the x axis, every gain setting overlaid. Top row: steps away from the rest pose in the positive direction; bottom row: the negative direction. Amplitude grows left to right. Dotted lines mark the level each run settles at. The returns to the rest pose, omitted here, all settle within a few millimetres. 118
15.5 Outward steps on the y axis, every gain setting overlaid. Top row: steps away from the rest pose in the positive direction; bottom row: the negative direction. Amplitude grows left to right. Dotted lines mark the level each run settles at. The returns to the rest pose, omitted here, all settle within a few millimetres. 119
15.6 Reference and measured tip path for the three shapes. 119

15.7 Circle. 120
15.8 Figure-eight. 121
15.9 Square. 122
16.1 Video of some parts of the experiments. 127

Nomenclature

Greek Symbols

α	Helical angle (Chapter 2); angle in classical PCC parametrization (Chapter 3); rate of the moving-goal and gain-ramping filters (Chapter 10); slope of the sigmoid spring law (Table 10.2); gradient descent step size (Chapter 11)
β	Weight of the previous estimate in the MoCap velocity filter (Chapter 9)
$\gamma_{j,i}$	Routing angle of the j-th tendon of section i with respect to the local x axis
$\boldsymbol{\Delta}$	Configuration state vector in the curvature space, $[\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \boldsymbol{\Delta}_3]^T \in \mathbb{R}^9$
$\boldsymbol{\Delta}_i$	Configuration of the i-th section, $[\Delta_{x,i}, \Delta_{y,i}, \delta L_i]^T$
Δ_i	Scalar curvature magnitude of the i-th section ($\sqrt{\Delta_{x,i}^2 + \Delta_{y,i}^2}$)
ΔL	Tendon length variation (negative when the tendon shortens)
δL_i	Variation in length of the central axis of section i
ΔE	Energy injected by one adaptation step, debited from the tank
δd	Virtual spring deflection, $d_{ref} - d$
η	Lumped quasi-static transmission factor
θ	Angle of curvature
ϕ	Direction of bending
$\boldsymbol{\tau}$	Motor torque vector
$\boldsymbol{\tau}_0$	Constant pretension torque

Roman Symbols

A_i	Tendon-to-configuration matrix of section i ($\Delta L_i = A_i \boldsymbol{\Delta}_i$)
D	Structural damping matrix of the soft manipulator
D_d	Virtual (control) damping matrix
d	Radial distance of the tendons from the neutral axis (Chapters 3 and 8); position of a virtual element anchor (Chapters 10 and 11)
d_{ref}	Virtual spring anchor rest position
e	Position error of a virtual element, $p_d - p$ (Chapter 10); force error, $f_x - f_{x,ref}$ (Chapter 11)
f_x	Cartesian interaction force at the contact point
J	Positional task-space Jacobian matrix
J_Δ	Constant analytical map from actuator to curvature space
J_d, J_x	Positional Jacobians at the anchor points and at the contact point
K	Structural stiffness matrix of the soft manipulator
K_d	Virtual (control) stiffness matrix
K_t	Motor torque constant
$L_{0,i}$	Rest length of section i
n, P	Unit contact normal and the associated projector $P = nn^T$
p	Cartesian position vector
$\boldsymbol{q}$	Actuator joint-space position vector
R_{motor}	Motor pulley radius
s	Normalized arc-length parameter ($s \in (0,1]$)
s_f	Safety factor dividing the passivity bound on the step size
T, T_{init}, T_{max}	Energy tank level, initial charge and capacity
Z_i	Energy budget available at step i, $T_i + \Delta t\, P^{IN}$

Part I

Introduction and Background

Chapter 1

Introduction

1.1 Robotics

Robotics is one of the most rapidly evolving fields of engineering; its development will impact a vast number of sectors, shaping the future of technology and sociology, and changing the way humans live their lives. A brief introduction to the state of the art of Robotics is necessary in order to understand what the challenges are, the different approaches applied nowadays, the problems that have not yet been solved, and the approach proposed in the current thesis.

A first categorization of Robotics can be made based on the operating environment:

- **Factories** have already successfully implemented specific zones in which manipulators can perform specific and repetitive tasks with extremely high precision compared to what a human can do. However, if the factory has to change the design or the manufacturing process of a component, the entire production line must be re-designed. Some companies are trying to use humanoid robots inside factories in order to be flexible: they can perform actions with the precision of a robot, but at the same time they are adaptable to changes in production lines and can perform multiple tasks. Currently, however, both manipulators and humanoids work in delimited areas as far as possible from humans. Due to their high mass and structural stiffness, accidental collisions can cause severe injuries to human operators, strictly limiting safe human-robot interaction.
- Regarding the **domestic** environment, robots are typically designed as rigid entities dedicated to specific tasks (e.g., dishwashers, vacuum cleaners). Even here, the ultimate concept is a single humanoid robot performing all the domestic tasks. While in factories rigid robots can still be applied in delimited areas, in the domestic environment the house must be shared with the robot, which must interact actively with people and delicate objects.

- The **medical** environment is similar to the two previously mentioned. There are rigid robots in delimited areas (operating rooms), but it would be dangerous interacting with humans autonomously.

The key idea is that robotics is changing its paradigm, moving from classical rigid robots to soft and compliant robots, which can share space, emotions, and contact with humans.

1.2 Rigid vs Soft Robotics

The natural consequence is analyzing the comparison between rigid and soft robots. What are the positive and negative aspects of these two approaches to designing robots?

Rigidity means high precision in task-space position tracking, reduced vibrations and disturbances, and simpler mathematical kinematic and dynamic modelling of the robot, which consequently facilitates its control. Indeed, the control of manipulators widely applied in factories is based on models (i.e., Denavit–Hartenberg) whose main assumptions include treating joints and links as rigid bodies. The downside, as previously mentioned, is that they are rigid and heavy, making them dangerous during human interaction and the manipulation of delicate objects.

On the other hand, most animals and humans are mainly characterized by a rigid skeleton, an elastic muscle structure, and a skin (soft or thick depending on the environmental conditions in which their species evolved). They can perform a vast range of activities with unmatched performance in terms of manipulation and compliance compared to rigid bodies. However, they are difficult to model and control due to high non-linearity.

This difference in performance motivated researchers to find new ways to design and control robots. This paradigm shift led to the emergence of Soft Robotics. The goal is to introduce elastic elements to reproduce the muscle structure and attempt to get closer to the optimal performance mentioned before. A branch of this field of research is continuum robotics, in which this concept is brought to an extreme: the rigid skeleton is removed, trying to imitate invertebrate animals or the soft appendages of vertebrate animals (e.g., an elephant's trunk, the tail of a monkey, octopus tentacles). This also implies a huge improvement in terms of dexterity.



(a) Musculoskeletal structure (b) Octopus tentacles

Figure 1.1: Biological structures inspiring soft and continuum robotics.

1.3 Control

As explained before, rigid manipulators are easy to model and control through kinematic inversion. On the other hand, humanoids or complex dexterous robots increasingly rely on learning-based approaches (such as Reinforcement Learning) or Whole-Body Control. The concept is that the robot must learn by itself what is correct or incorrect by maximizing a reward function. RL relies on a huge amount of data to train the robot for any specific task in real environments or simulations. Applying inverse kinematics or RL presents significant challenges for soft robots: IK requires a highly reliable kinematic model, is computationally heavy due to hyper-redundancy, and is strictly prone to mathematical singularities. Conversely, RL becomes unpredictable when dealing with highly non-linear soft materials.

1.4 Goal of the thesis

Following the ongoing research in Soft Robotics conducted at the CREATE Lab (EPFL), the primary objective of this thesis is the design and control of a dual-arm soft robotic system. The two arms are conceived to perform cooperative manipulation tasks, governed by a Virtual Model Control (VMC) architecture to handle and exploit the intrinsic compliance and non-linearities of the soft structure, effectively bypassing the aforementioned limitations of Inverse Kinematics.

1.5 Main contributions

This section outlines the main steps and contributions of the project, which will be analyzed in detail in the following chapters:

- **Mechanical design and fabrication:** Design of the bimanual robot in Autodesk Fusion 360, 3D-printing in PLA and TPU, and assembly of the mechanical and transmission components.
- **Sensor integration:** Design and assembly of the IMU acquisition circuit — multiplexed I2C bus, microcontroller firmware and wiring — left ready for future proprioception in the state estimator.
- **Kinematics:** Adaptation of the constant-curvature kinematics to the three-tendon routing of this platform, and compilation of the analytical Jacobian for use inside the control cycle.
- **System identification:** Experimental identification of the structural stiffness and damping of both arms, from two independent sensing sources.
- **State estimation:** A Kalman filter fusing encoders and Motion Capture in curvature space.

- **Control architecture:** Implementation of the Virtual Model Control strategy and of the force-regulation layer that adapts the virtual parameters online under a passivity constraint, whose theory is due to L. Vignoli (Chapter 11).
- **Sensorless force control:** Contact force inferred from the virtual model and the identified structural reaction, with no force sensor in the control path, and regulated in open and closed loop.
- **User interface:** A graphical station with a real-time digital twin of both arms, runtime configuration of the virtual elements, and a panel for the force-regulation parameters.
- **Experimental validation:** Calibration against a load cell, force tracking, direct measurement of the reachable workspace, step response and trajectory tracking — together with the physical limits these revealed.

Chapter 2

State of the Art

2.1 Soft arms

The design goal of soft continuum robots is achieving the ability to geometrically tune the two natural modes of deformation: the axial and bending stiffness of the structure. Several techniques can be adopted to obtain this. The traditional approach is based on the use of silicone or composites like meta-materials, inducing customized properties through internal microstructure changes. However, defining a unique correspondence with physical properties in meta-materials is not trivial and remains complex to manage.

A more effective approach is using architectured structures, exploiting the macroscopic spatial geometry to define the physical properties of a homogeneous material. While various architectured structures have been explored for soft robotics, a highly innovative contribution has been recently made by Guan et al. [2023]. They introduced the concept of trimmed helicoids (TH), demonstrating that this specific geometry allows for the independent regulation of bending and axial stiffness. For this reason, the TH geometry was chosen as the foundational building block to design the soft arms for this project.

The TH structure is obtained by stacking a certain number of helicoids along the central axis. A geometric peculiarity of helicoids is that they present a surface gradient (dependent on the helical angle α) that decreases with the radius. The inner geometry is steeper with respect to the outer one. Consequently, the material reacts to different external stimuli by exploiting different parts of the structure: compression is counterbalanced by the inner material, while bending stress—being concentrated on the external part—is counterbalanced by the outer material.

By simply adjusting geometric parameters such as thickness and trimming ratio, the architectured structure can be tailored for specific task requirements.

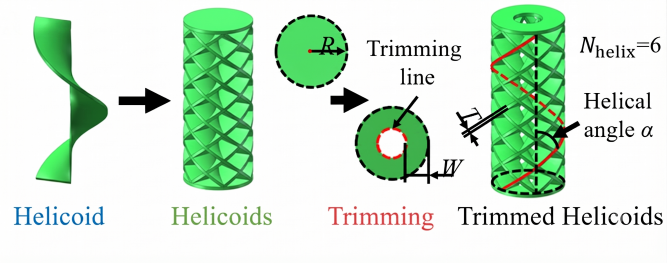


Figure 2.1: Geometric parameters defining the Trimmed Helicoid (TH) structure. Adapted from [Guan et al., 2023].

2.2 Actuation system

Soft arms are typically actuated pneumatically or through tendons. In the reference architecture [Guan et al., 2023], the soft arm is composed of three continuous sections, each actuated by 3 tendons, totaling 9 motors per arm.

When using standard tendon routing, the movements between the sections are heavily coupled: a tendon pulling the distal section (third one) inevitably compresses and bends the proximal sections it passes through. To resolve this, that architecture employs Bowden cables to mechanically decouple the sections. This decoupling was shown to provide a significant increase in performances in terms of reachability, shape reconstruction, and sensitivity to the control action. Figure 2.2 depicts the difference between coupled and decoupled actuation, alongside a photograph of the fabricated arm of the paper.

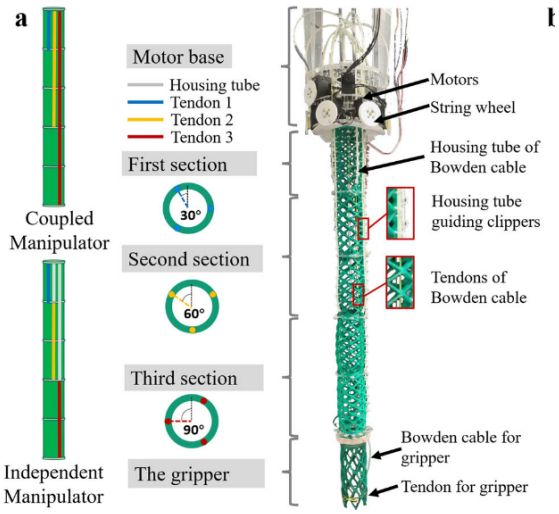


Figure 2.2: Comparison between coupled and decoupled tendon actuation. Adapted from [Guan et al., 2023].

In the independent actuation scheme, Bowden cables are utilized to route the tendons through the proximal sections without applying any compressive or bending loads to them. For instance, the tendon driving the distal section runs freely inside a Bowden sheath anchored at the end of the second section; only upon exiting the sheath does the tendon exert tension on the third section, ensuring complete actuation decoupling across the manipulator.

2.3 Bimanual continuum platforms

Bimanual continuum platforms remain rare, and the reported ones are almost entirely surgical: cooperative knot-tying [Quaicoe et al., 2024], transendoscopic dissection [Gao et al., 2024], and underwater grasping with two soft arms [Shen et al., 2024]. In all of them coordination is posed as a position or trajectory problem; the force exchanged at the contact is not the regulated quantity.

The trimmed-helicoid arms described above are already deployed across several projects at the CREATE Lab, typically operated through position control. Regulating the force they exchange with what they touch — without adding a force sensor — is what this work adds.

Part II

Theoretical Framework

Chapter 3

State Parametrization

Usually, the state of a rigid robot is defined by its discrete joint positions and velocities. However, for tendon-driven soft continuum robots, the lack of rigid links and discrete joints requires a fundamentally different approach to describe their configuration in space: a continuum body has infinitely many degrees of freedom, so any model-based controller must first commit to a finite, reduced-order description of its shape [Della Santina et al., 2023].

3.1 PCC Model

Piecewise Constant Curvature (PCC) is the reduced-order hypothesis adopted here, and the most widespread one in the continuum-robotics literature [Jones and Walker, 2006]. The core idea is to approximate the soft arm as a sequence of continuous segments, each with a curvature that is constant in space (CC) but variable in time. Each segment is free to bend in any direction, and they are connected to each other such that the resulting curve is everywhere differentiable.

The hypothesis is an approximation, and its validity is not unconditional. Under external load there is a risk that it no longer holds, since a loaded section need not deform as a circular arc; reduced-order models that relax the constant-curvature assumption have been proposed for exactly that regime [Stella et al., 2023a].

By assigning a local reference frame to the base and to the end of each segment, the classical state parametrization historically used for PCC models is the α-parametrization. For a single segment, this is defined as $\alpha_i = [\phi_i, \theta_i, \delta L_i]^T$. In this representation, ϕ_i is the angle expressing the direction of bending, θ_i is the angle of curvature, and δL_i is the change in length of the central axis.

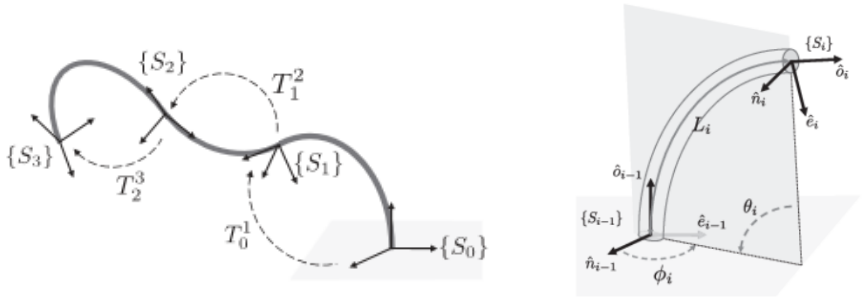


Figure 3.1: Piecewise Constant Curvature model of a soft robotic arm. Adapted from [Della Santina et al., 2020a].

However, this parametrization introduces critical kinematic singularities and discontinuities. For example in case a) of Figure 3.2, in the perfectly straight configuration ($\theta_i = 0$), the direction of bending ϕ_i becomes mathematically undefined. This means there are infinite choices of the vector α_i describing a single physical configuration.

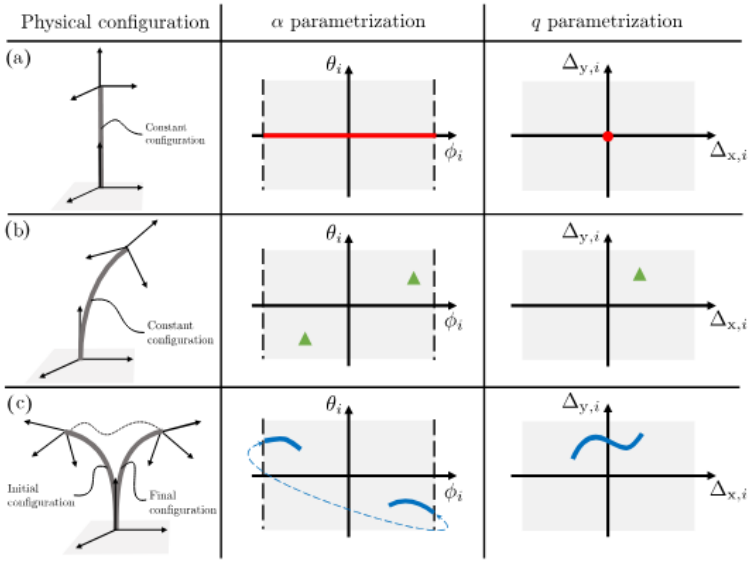


Figure 3.2: Kinematic singularities introduced by the classical α-parametrization. Adapted from [Della Santina et al., 2020a].

To overcome these topological limitations, an alternative state parametrization was proposed by Della Santina et al. [2020a], which is the foundational framework adopted in this thesis work. The derivation that follows is theirs; it is reproduced here because every later chapter builds directly on the two closed-form expressions it produces.

To maintain strict notational consistency in the following chapters—reserving the letter q exclusively for the actuator space—the new configuration vector for the i-th segment is defined as $\mathbf{\Delta}_i = [\Delta_{x,i}, \Delta_{y,i}, \delta L_i]^T$. In this formulation, $\Delta_{x,i}$ and $\Delta_{y,i}$ represent the shortening that a hypothetical tendon would experience if placed exactly on the local X and Y axes, while δL_i remains the variation in length of the central axis. Additionally, the scalar curvature magnitude is defined as $\Delta_i \triangleq \sqrt{\Delta_{x,i}^2 + \Delta_{y,i}^2}$.

The physical intuition behind this state parametrization can be derived geometrically. When the soft segment bends, the central neutral axis does not undergo any compression or elongation due to the bending itself. Its length is $L_i = L_{0,i} + \delta L_i$, and its radius of curvature is linked to the parameter θ_i of the α-parametrization by the relation $r = \frac{L_i}{\theta_i}$.

Based on the physical routing of the tendons, each cable sits at a certain distance from the neutral bending plane, denoted as $\delta_{j,i}$. This distance can be computed through trigonometric projections and depends on the orientation of the bending plane, expressed by ϕ_i. Consequently, the local radius of curvature for the j-th tendon is the radius of the central axis minus its specific offset: $r_j = r - \delta_{j,i}$. The length of the tendon arc then becomes:

$$L_{j,i} = r_j \theta_i = (r - \delta_{j,i})\theta_i = L_i - \theta_i \delta_{j,i} \tag{3.1}$$

Therefore, the term $\theta_i \delta_{j,i}$ represents the exact amount of tendon that must be pulled or released to bend the segment by an angle θ_i along the direction ϕ_i.

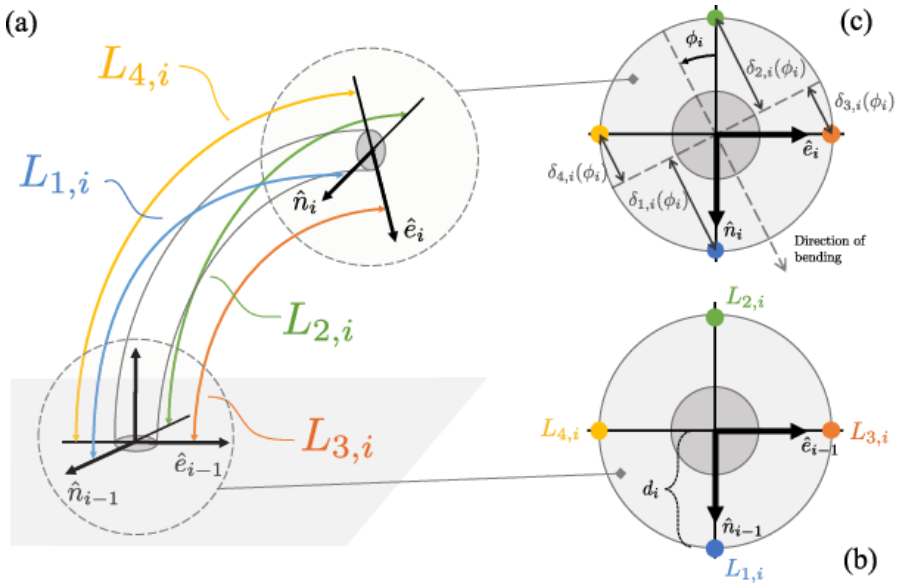


Figure 3.3: Geometric quantities used for the proposed state parametrization. Adapted from [Della Santina et al., 2020a].

The reference paper originally demonstrates this mapping using an example with 4 orthogonally routed tendons, where $\Delta_{x,i}$ and $\Delta_{y,i}$ can be obtained simply by computing the difference between opposite cables.

However, this is not the most generalized method to uniquely extract these parameters. Because the bimanual platform developed in this thesis utilizes 3 tendons per section spaced at 120°, a specific geometric transformation based on a non-orthogonal matrix inversion has been implemented. A deeper mathematical analysis of this mapping will be detailed in the kinematics implementation chapter.

3.2 Forward Kinematics

To determine the position and orientation of the soft arm in the Cartesian space, a direct mapping from the configuration space $\mathbf{\Delta}_i$ is required. The transformation matrix $T^i_{i-1} \in SE(3)$ maps the local frame of the $(i-1)$-th segment to the i-th segment.

Instead of computing sequential rotations using Euler angles, which inevitably lead to representation singularities, the rotation matrix R^i_{i-1} is derived using the Rodrigues' rotation formula. This approach rotates the frame by an angle $\theta_i = \Delta_i/d_i$ around an axis mutually perpendicular to the neutral axis and the direction of bending. By substituting the geometric relations $\cos(\phi_i) = \Delta_{x,i}/\Delta_i$ and $\sin(\phi_i) = \Delta_{y,i}/\Delta_i$ into the Rodrigues' formula, the rotation matrix takes the following explicit form:

$$R^i_{i-1} = \begin{bmatrix} 1 + \frac{\Delta^2_{x,i}}{\Delta^2_i}\left(\cos\left(\frac{\Delta_i}{d_i}\right) - 1\right) & \frac{\Delta_{x,i}\Delta_{y,i}}{\Delta^2_i}\left(\cos\left(\frac{\Delta_i}{d_i}\right) - 1\right) & -\frac{\Delta_{x,i}}{\Delta_i}\sin\left(\frac{\Delta_i}{d_i}\right) \\ \frac{\Delta_{x,i}\Delta_{y,i}}{\Delta^2_i}\left(\cos\left(\frac{\Delta_i}{d_i}\right) - 1\right) & 1 + \frac{\Delta^2_{y,i}}{\Delta^2_i}\left(\cos\left(\frac{\Delta_i}{d_i}\right) - 1\right) & -\frac{\Delta_{y,i}}{\Delta_i}\sin\left(\frac{\Delta_i}{d_i}\right) \\ \frac{\Delta_{x,i}}{\Delta_i}\sin\left(\frac{\Delta_i}{d_i}\right) & \frac{\Delta_{y,i}}{\Delta_i}\sin\left(\frac{\Delta_i}{d_i}\right) & \cos\left(\frac{\Delta_i}{d_i}\right) \end{bmatrix} \tag{3.2}$$

Similarly, the translation vector t^i_{i-1} defines the position of the segment's tip. Geometrically, this vector describes an arc of a circle with length $L_i = L_{0,i} + \delta L_i$ and radius $r = L_i/\theta_i$. By projecting the arc equations onto the local Cartesian axes and applying the proposed $\mathbf{\Delta}_i$ parametrization, the translation vector is obtained without relying on the singular variable ϕ_i:

$$t^i_{i-1} = \frac{d_i\left(L_{0,i} + \delta L_i\right)}{\Delta^2_i} \begin{bmatrix} \Delta_{x,i}\left(1 - \cos\left(\frac{\Delta_i}{d_i}\right)\right) \\ \Delta_{y,i}\left(1 - \cos\left(\frac{\Delta_i}{d_i}\right)\right) \\ \Delta_i \sin\left(\frac{\Delta_i}{d_i}\right) \end{bmatrix} \tag{3.3}$$

3.3 Differential Kinematics and Jacobian Stability

The most significant advantage of this state parametrization emerges in the differential kinematics. In the classical α-parametrization, the analytical Jacobian matrix loses rank when $\theta_i \to 0$, making model-based control mathematically unstable in the straight configuration.

Conversely, the new configuration vector $\mathbf{\Delta}_i$ ensures that the spatial derivatives of both R^i_{i-1} and t^i_{i-1} remain well-defined and continuous across the entire workspace. Because the limits for $\Delta_i \to 0$ are well-defined, the resulting Jacobian matrix maintains full rank even when the soft manipulator is perfectly straight. This structural stability prevents discontinuities in the commanded torques and enables robust, singularity-free task-space control for the entire soft arm.

Chapter 4

Virtual Model Control: Background

4.1 Impedance Control and Task-Space Compliance

In classical rigid manipulators, safe interaction with the environment is managed through software-defined compliance. The foundational technique for this is Impedance Control [Hogan, 1985]. The key idea is to make the robot's end-effector behave as if it were linked to a target point in the task space by means of a virtual spring and damper. The desired dynamic behavior of the robot is described by the following second-order error equation:

$$M_d\ddot{e} + D_d\dot{e} + K_d e = F_{ext} \tag{4.1}$$

where M_d is the desired apparent inertia of the robot, K_d and D_d are the virtual spring and damping matrices respectively, $e = x_d - x$ is the Cartesian tracking error, and F_{ext} is the external force applied by the environment.

To enforce this desired inertia M_d on a physical system, the controller must actively compensate for the robot's natural dynamics. This requires full knowledge of the robot's mass matrix $M(q)$, Coriolis and centrifugal terms $C(q, \dot{q})$, gravity vector $g(q)$, and the external force F_{ext}:

$$\tau = M(q)J^{-1}\left[\ddot{x}_d + M_d^{-1}(D_d\dot{e} + K_d e - F_{ext}) - \dot{J}\dot{q}\right] + C(q,\dot{q})\dot{q} + g(q) - J^T F_{ext} \tag{4.2}$$

Because this approach relies on a perfect dynamic model and the exact inversion of the kinematic Jacobian J^{-1}, full impedance control is typically applied only to highly sensorized, expensive rigid manipulators.

By giving up the requirement to actively shape the robot's inertia, the dynamic compensation simplifies drastically. The control law reduces to simulating the restoring forces of the virtual spring and damper, projected back into the joint space, alongside gravity

compensation:

$$\tau = J^T \underbrace{(K_d e + D_d \dot{e})}_{F_{vmc}} + \tau_g(q) \tag{4.3}$$

This formulation avoids computing inverse kinematics or inverse dynamics entirely, relying strictly on the transposed Jacobian to map Cartesian forces into actuator torques.

4.2 Virtual Model Control (VMC)

Virtual Model Control (VMC) is a natural extension of simplified impedance control. Originally introduced by Pratt et al. [2001] for the control of bipedal walking robots, VMC generalizes the concept of task-space compliance. Rather than restricting the virtual spring and damper to the end-effector alone, VMC allows virtual mechanical components to be attached to *any* point on the entire body of the robot. In Pratt's early work, virtual components were distributed across the Spring Flamingo robot to regulate posture, ride height, and swing-leg trajectories simultaneously. The same construction has since been applied to quadrupedal locomotion [Yang et al., 2026].



Figure 4.1: Spring Flamingo robot. From [MIT Leg Laboratory, 2000].

4.3 VMC for Soft Continuum Robots

For soft continuum robots driven by tendons, VMC is particularly advantageous. Soft manipulators possess complex, nonlinear, and uncertain dynamics that complicate classical inverse-dynamics control [Della Santina et al., 2023]. VMC bypasses the need to formulate and invert these rigid-body dynamic models. Instead, it shapes the closed-loop

behavior entirely in the Cartesian space. By superimposing virtual forces at multiple discrete points along the continuum backbone, VMC implicitly resolves the manipulator's kinematic redundancy without requiring pseudo-inverses. It works synergistically with the physical flexibility of the soft arms, exploiting their intrinsic compliance to achieve robust, safe, and coordinated manipulation.

Virtual Model Control is an established framework, but its applications have remained on rigid platforms: bipedal [Pratt et al., 2001], quadrupedal [Yang et al., 2026], humanoid reaching [Zhang et al., 2024] and rigid surgical instruments [Larby and Forni, 2023], with recent work formalising its passivity-based design and tuning [Larby and Forni, 2026]. Soft continuum arms, meanwhile, have been given interaction controllers of other kinds — model-based dynamic feedback [Della Santina et al., 2020b] or task-space stiffness prescribed by co-optimising shape and physical segment stiffness [Stella et al., 2023b].

To the best of our knowledge, that combination has not been reported: a virtual model whose elements are distributed along a continuum body, with adaptation bounded so that contact with an unknown environment remains passive. Its implementation and experimental validation on a bimanual soft platform are the subject of this thesis.

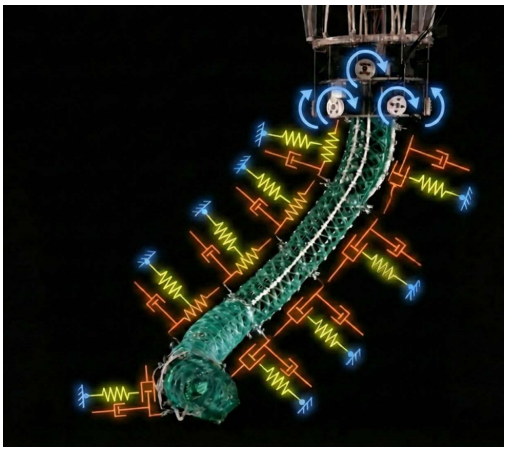


Figure 4.2: Graphical representation of VMC on a soft arm

Part III

System Development

Chapter 5

System Architecture and Project Workflow

This chapter provides a high-level overview of the operative workflow and the system architecture developed in this thesis. The final framework is the result of iterative design choices, hardware optimizations, and control testing, culminating in the integrated closed-loop architecture depicted in Figure 5.1. Each specific sub-module of this scheme will be analyzed in detail in the subsequent chapters.

The primary objective of this work is the implementation of Virtual Model Control (VMC) on a bimanual soft continuum manipulator. To achieve this, the project was divided into three main sequential phases:

1. **Mechatronic Design:** Modeling, 3D printing, and assembling the soft robot's mechanical components, followed by the integration of the hardware architecture (actuators, sensors, and microcontrollers).
2. **State Estimation:** Developing a reliable software pipeline to estimate the robot's configuration in real-time.
3. **Control Implementation:** Formulating and deploying the VMC loop, ensuring stable and intuitive task-space interaction.

The core of the control strategy relies on the Piecewise Constant Curvature (PCC) state parametrization, previously introduced in Chapter 3. The state of each section is fully described by the configuration vector $\boldsymbol{\Delta} = [\Delta_x, \Delta_y, \delta L]^T$. This state must be continuously estimated by fusing exteroceptive and proprioceptive data. The system relies primarily on motor encoders and a Motion Capture system. Inertial units were mounted on the arms but their data is not fed to the estimator; their integration is left to future work. While the sensors measure different physical quantities (joint positions, Cartesian positions, and velocities), their data is mathematically projected into the shared PCC configuration space.

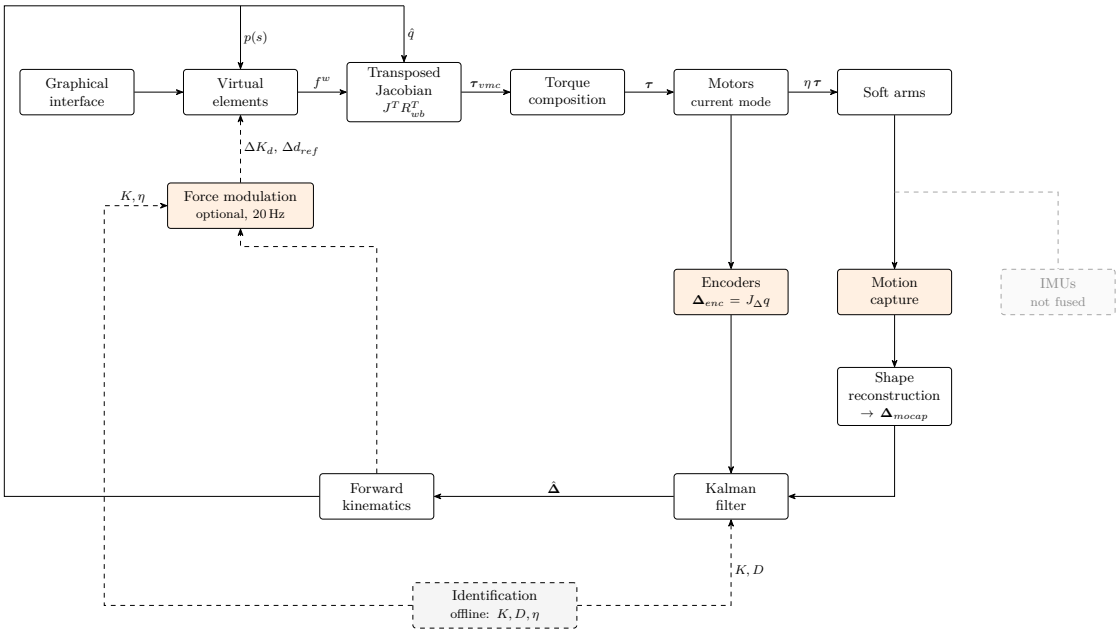


Figure 5.1: Architecture of the control loop. The virtual elements produce a Cartesian force, which the transposed Jacobian turns into motor torques; no inverse kinematics and no inverse dynamics appear anywhere in this path. Only the fraction η of the commanded torque survives the tendon transmission and reaches the structure. Every sensor is mapped into the PCC space before the filter. Solid: the 500 Hz path. Dashed: what is optional, slower, or offline.

To handle sensor noise and different sampling rates, a Kalman Filter is employed to fuse the measurements, dynamically weighting the information based on the statistical reliability (covariance) of each sensor. The prediction step of the filter relies on a dynamic model of the arm, whose natural stiffness and damping parameters were experimentally obtained through a dedicated System Identification procedure.

Once the state $\mathbf{\Delta}$ is robustly estimated, the Forward Kinematics block computes the spatial position and orientation of any arbitrary point along the arm. This spatial awareness is fed into a custom Graphical User Interface (GUI), featuring a real-time digital twin of the manipulator. The GUI facilitates the user's control action, allowing them to intuitively define target coordinates and attach virtual interactive elements (springs and dampers) directly onto the robot's body in real time.

Based on the user's input, the VMC generates a desired Cartesian force. To perform adaptable manipulation, this force can be dynamically regulated: a gradient descent algorithm modulates the virtual parameters to actively track a target contact force, itself inferred from the model rather than measured. Because this adaptation mathematically injects energy into the system, an Energy Tank-based architecture bounds the parameter updates, preserving stable and passive interactions.

Ultimately, the accuracy of the applied Cartesian force depends strictly on the precision of the state estimation and the kinematic mapping. The calculated task-space force is translated into the corresponding joint-space motor torques through the transposed analytical Jacobian ($\boldsymbol{\tau} = J^T F$), effectively driving the soft manipulator to accomplish the desired task.

Chapter 6

Hardware

6.1 Motors

Eighteen DYNAMIXEL XC330-T288-T motors were used in this project. They are frequently employed in academic robotics projects due to their high reliability and optimal balance between weight and performance.



Figure 6.1: Dynamixel XC330-T288-T used for the project

6.1.1 Mechanical characteristics

Weighing only 23 grams, these motors are extremely compact. The "T288" designation indicates a reduction gear ratio of 288.4:1, which limits the maximum velocity to 65 rpm but significantly increases the deliverable torque. The motors are driven by torque commands rather than by position targets, so the figure that matters is the torque ceiling: a stall torque of 0.92 N m leaves a substantial margin for the requested manipulation tasks. Furthermore, the internal gears and bearings are entirely metallic, ensuring high mechanical durability.

To provide state feedback, the motors feature a 12-bit magnetic encoder with 4096 ticks per revolution. This yields a position precision of 0.088°, which is strictly necessary to accurately drive the encoder-based kinematic estimations.

Table 6.1: Main characteristics of the DYNAMIXEL XC330-T288-T and of the bus configuration adopted on this platform.

Quantity	**Value**	**Unit**
Mass	23	g
Gear ratio	288.4:1	—
No-load speed	65	rpm
Stall torque	0.92	N m
Torque constant K_t	1.15	N m/A
Encoder resolution	4096	counts/rev
Angular resolution	0.088	deg
Velocity quantization	0.229	rpm
Communication protocol	TTL half-duplex, Protocol 2.0	—
Bus baud rate	3	Mbps
Motors on the bus	18	—
Operating mode	Current	—

6.1.2 Communication and control modalities

The motors support daisy-chain series connections. The bus cable consists of just three wires: power, ground, and data. This simplifies cable management significantly, which is critical when operating an 18-motor bimanual platform. Communication is handled via a standard TTL (Transistor-Transistor Logic) half-duplex protocol. It relies on a Master-Slave architecture where the master (the main PC) sends command packets containing a header, the receiver ID, and an instruction (write, read, ping). After a brief delay, the targeted motor replies with a Status Packet containing an error byte and the requested data.

The interface between the computing unit and the motor bus is managed by a ROBOTIS U2D2 board, which acts as a signal converter translating the standard USB protocol to the TTL logic levels.

Regarding the control architecture, the motors can be driven using three distinct nested loops:

- **Current (Inner loop):** Regulates the current, which translates directly into torque via the motor torque constant ($K_t = 1.15\,\mathrm{N\,m/A}$).
- **Velocity (Middle loop):** Reads the position time-derivative from the encoders. The internal controller computes the error against a target velocity, and a PI controller nullifies this error to generate a current reference for the inner loop.
- **Position (Outer loop):** Receives a target position and current angular feedback, computing a target velocity profile to feed into the middle loop.

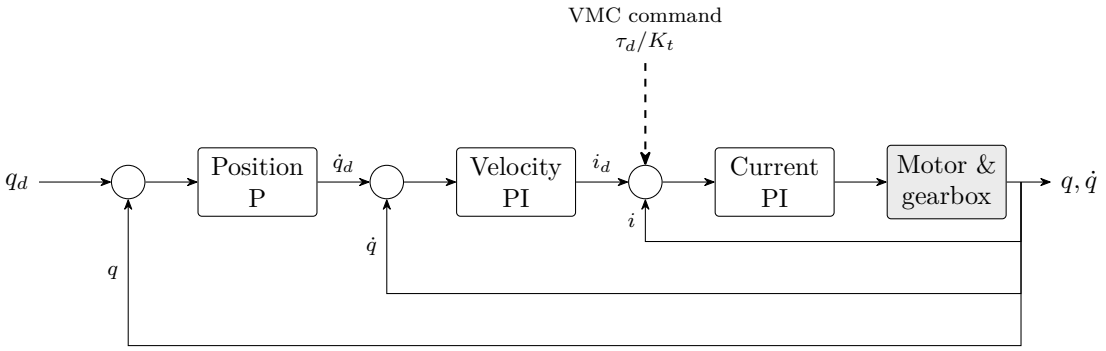


Figure 6.2: The three nested control loops implemented in the Dynamixel firmware. In this work the two outer loops are bypassed: the VMC torque command enters the chain directly as a current reference (dashed).

Because the VMC strategy computes analytical forces translated into torques via the Jacobian, the motors are commanded purely through torque references. These are converted to current references via K_t. Consequently, only the inner current loop is active. Utilizing the position loop would have required solving the exact inverse kinematics of the hyper-redundant soft arm, violating the core premise of the VMC approach.

6.1.3 Dynamixel Wizard

A graphical user interface called Dynamixel Wizard, provided by ROBOTIS, was utilized to test the various control modes, update firmware, and assign unique IDs and baud rates to each motor. Because the VMC control loop executes at 500 Hz, the baud rate of all motors was raised as far as the bus would reliably allow, in order to minimize communication latency. The maximum supported rate of 4 Mbps proved unstable with 18 motors on a single chain, occasionally dropping packets, so the platform settles on 3 Mbps.

6.1.4 C++ Node Implementation

Standard sequential TTL communication presents severe limitations in terms of latency and bandwidth, making it unsuitable for a high-frequency 18-motor control loop. The platform therefore relies on a custom C++ node, developed by L. Vignoli, which integrates the motor control directly into the ROS 2 architecture; the contribution of this thesis to it is the velocity filtering described below. Its main architectural optimizations are:

- **Synchronous Broadcasting:** Instead of addressing each slave in sequence, a single broadcast packet carries the commands for all 18 motors, using the Dynamixel SDK APIs `GroupSyncWrite` and `GroupSyncRead`. The motors are configured not to acknowledge write commands, so only the subsequent read transaction produces replies; a 100 μs delay separates the two to avoid bus collisions.

```cpp
// --- Synchronizing in writing mode (Broadcast) ---
syncWrite_->clearParam();
for (size_t i = 0; i < motor_ids_.size(); i++) {
    // ... [Data conversion and byte-splitting] ...
    syncWrite_->addParam(id, param);
}
// Only one packet sent for all 18 motors
syncWrite_->txPacket();

usleep(100); // Micro-delay before the read transaction

// --- Synchronization in Reading mode (global feedback) ---
int comm_result = syncRead_->txRxPacket();
if (comm_result != COMM_SUCCESS) {
    RCLCPP_WARN_THROTTLE(this->get_logger(), *this->get_clock(), 1000,
                         "SyncRead failed: %d", comm_result);
    return;
}

```

Listing 6.1: GroupSyncWrite and GroupSyncRead utilization.

- **Thread-Safe Memory Management:** The `goal_command_storage_` array is written by the ROS callback whenever a new torque command arrives, and read by the `publishState()` function, which runs on its own 1 ms timer — deliberately faster than the 500 Hz control loop, so that the controller never has to wait on the bus for fresh state. To prevent data tearing or collisions during simultaneous read/write operations, a mutex locks the array during updates. Listing 6.2 demonstrates the asynchronous ROS 2 callback handling the torque targets. The lock guarantees the timer callback never reads a partially overwritten command vector.

```cpp
// Asynchronous reception of torque targets through ROS 2
void torqueCallback(
    const std_msgs::msg::Float64MultiArray::SharedPtr msg)
{
    if (msg->data.size() != motor_ids_.size()) return;

    // The lock prevents the timer from reading until the update is complete
    std::lock_guard<std::mutex> lock(goal_mutex_);
    for (size_t i = 0; i < motor_ids_.size(); i++) {
        // Convert torque (Nm) into the current reference via Kt
        goal_command_storage_[i] = msg->data[i] / KT;
    }
} // The lock is released automatically at the end of the scope

```

Listing 6.2: ROS 2 callback for receiving and safely storing motor torque targets.

- **Velocity Filtering:** Velocity is a critical state variable for the VMC's virtual dampers, and the raw finite-difference velocity derived from the encoders is heavily

quantized. An Exponential Moving Average (EMA) low-pass filter was therefore added to the node in the course of this work, smoothing the signal directly at the source before publication.

```
// Inside the publishState() data extraction loop
for (size_t i = 0; i < motor_ids_.size(); i++) {
    // ... [Position extraction and raw velocity calculation (vel_deg_s)] ...

    // Apply the EMA low-pass filter to the velocity signal
    filtered_velocities_[i] = velocity_filter_alpha_ * vel_deg_s
                            + (1.0 - velocity_filter_alpha_) *
    filtered_velocities_[i];

    vel_msg_.data.push_back(filtered_velocities_[i]);
}

```

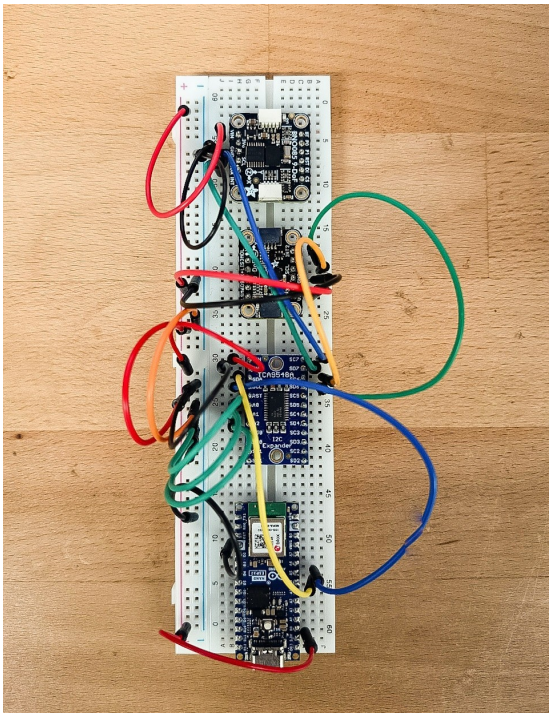
Listing 6.3: EMA filter applied to the raw velocity data.

6.2 IMUs

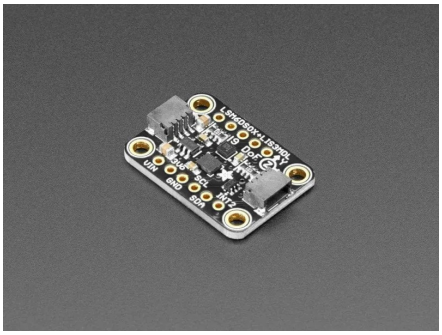
6.2.1 Electronic circuit

The IMUs selected for this platform are the *Adafruit LSM6DSOX + LIS3MDL 9-DoF IMU*. Each board houses an LSM6DSOX (6-DoF accelerometer and gyroscope) and a LIS3MDL (3-axis magnetometer). Communication is established via the I2C (Inter-Integrated Circuit) protocol, requiring only four lines: Power, Ground, Data (SDA), and Clock (SCL).

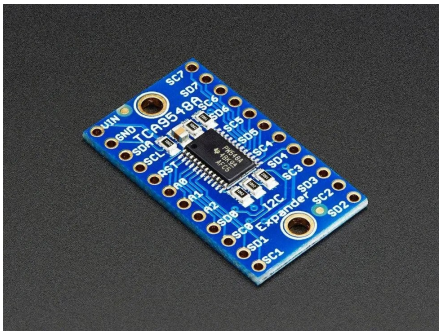
Because all IMUs of this specific model share the same hardcoded hexadecimal I2C address, wiring them to a single bus would cause immediate data collisions. To manage the array, a TCA9548A I2C multiplexer is utilized, allowing an ESP32 microcontroller to address each IMU individually. The hardware logic was initially verified on a breadboard prototype before final integration.



(a) Electronic prototyping setup



(b) Nine-DoF IMU board



(c) I2C multiplexer

Figure 6.3: Electronic prototyping setup and the main components of the IMU acquisition system.

6.2.2 Arduino code

A diagnostic C++ routine was deployed on the microcontroller to validate the hardware integrity, the I2C multiplexing logic, and the raw data acquisition from the IMUs. As the state estimation relies exclusively on spatial kinematic data, the magnetometer readings were deliberately excluded.

The integration of this standalone diagnostic script into a real-time micro-ROS node is left for future work. The fundamental operations of the acquisition logic are broken down below:

- **Bus Routing:** The `tcaSelect` and `tcaDisableAll` functions actively control the TCA9548A multiplexer. It acts as a gatekeeper, isolating a single I2C channel at a time to prevent bus collisions between the identically-addressed IMUs.
- **Sensor Initialization:** During setup, the routine iteratively opens each configured channel, verifies the presence of the LSM6DSOX, and configures its internal registers. The accelerometer is scaled to $\pm 4g$ and the gyroscope to $\pm 500\,\text{dps}$, with both data rates set to 52 Hz to balance responsiveness against electrical noise.

- **Sequential Polling:** Inside the main loop, the microcontroller switches the active channel dynamically. It then queries the target IMU through the Adafruit unified sensor library and clears the bus before the next iteration.

6.2.3 Soldering

Once the sensors were calibrated and the logic validated, all the wiring was soldered directly to the electronic components, which were then mounted on their custom mechanical supports. The wires were twisted to mitigate electromagnetic interference and simplify cable management. Before fully assembling the robotic arms, a final signal test was conducted to verify the integrity of the electronic hardware and ensure no cold joints disrupted the I2C bus.



Figure 6.4: Soldering setup and final hardware verification.

Chapter 7

Mechanical Design

Before implementing the software architecture, the physical robotic platform had to be manufactured and assembled. The procedure followed includes: mechanical design choices, CAD modeling in Autodesk Fusion 360, rapid prototyping via 3D printing (using a Bambu Lab X1-Carbon), and final hardware assembly.

7.1 CAD and 3D Printing

7.1.1 Soft Arms

The concept of independent actuation via Bowden cables was employed. A baseline CAD model of the soft arm module was adapted from previous projects of the CREATE Lab. The primary geometric modification concerned the dimensions of the rings hosting the Bowden cables (which have an external diameter of 3 mm), which were precisely toleranced to ensure proper clearance; otherwise, additional friction and hysteresis would have been introduced into the transmission system. The modules are identical, though the first one includes an integrated base to attach the arm to the robot's chassis.

Although the reference paper suggested TPU 72D, slightly different materials were utilized in the current work: TPU 95A for the right arm and TPU 90A for the left one. Therefore, while the general kineto-static theory behind the TH remains valid, the specific stiffness parameters vary due to the choice of materials. Furthermore, printing the two arms in different grades contributes to an asymmetric stiffness profile between them, a characteristic that can theoretically be exploited in specialized bimanual manipulation tasks. How large that asymmetry actually is, and how much of it the material choice alone accounts for, is measured in Section 13.3.

The 3D printing process was particularly challenging due to the flexible nature of the TPU filament and the suspended geometry of the helicoid rings. Several prototypes were tested iteratively to achieve a satisfactory result. Custom manual supports were specifically designed to hold the rings, and critical slicing parameters were meticulously

tuned. The printing velocity was drastically reduced to ensure consistent extrusion, and residual plastic stringing was safely removed post-print using a heat gun. Table 7.1 reports the primary 3D printing parameters optimized for these soft components.

Parameter	**Value**
Speed Settings	
Outer / Inner Wall Speed	$25 - 30\,\text{mm/s}$ / $30 - 40\,\text{mm/s}$
Sparse Infill / Gap Infill	$35 - 40\,\text{mm/s}$ / $20 - 25\,\text{mm/s}$
Top Surface / Bridge	$20\,\text{mm/s}$ / $15 - 20\,\text{mm/s}$
Travel Speed	$250\,\text{mm/s}$ (reduced to minimize stringing)
Acceleration Settings	
Normal Printing	$500 - 1000\,\text{mm/s}^2$
Outer Wall / Bridge	$300 - 500\,\text{mm/s}^2$ / $500\,\text{mm/s}^2$
Filament & Quality Settings (TPU 95A / 90A)	
Retraction Distance & Speed	$1.5 - 2.0\,\text{mm}$ @ $30 - 40\,\text{mm/s}$
Z-Hop	$0.4\,\text{mm}$
Support Type	Tree (Manual), Slim style
Top Z Distance	$0.25 - 0.30\,\text{mm}$
Part Cooling Fan	100% after initial layers
Avoid Crossing Walls	Enabled

Table 7.1: Optimized 3D printing parameters for TPU soft modules on the Bambu Lab X1-Carbon.

A primary design decision involved defining the general dimensions of the arms, the number of modules per section, and the individual module length. A shorter section is stiffer and its shape is reconstructed more accurately, since the constant-curvature hypothesis has less length over which to drift; a longer one buys reach. The two requirements act at different points along the arm, so the three sections were given different lengths: the proximal section stacks two modules, where reach matters most and the accumulated pose error is smallest, while the second and third sections use a single module each, placing the finest sections where precision is needed — at the end-effector. Each module is 144.6 mm long, so that the resting lengths of the three sections are 289.2 mm, 144.6 mm and 144.6 mm. The external radius is 30 mm, the inner trimming hole has a radius $W = 20\,\text{mm}$, and the thickness of the helicoidal struts is 2.118 mm. Each module features integrated routing holes to guide the tendons.

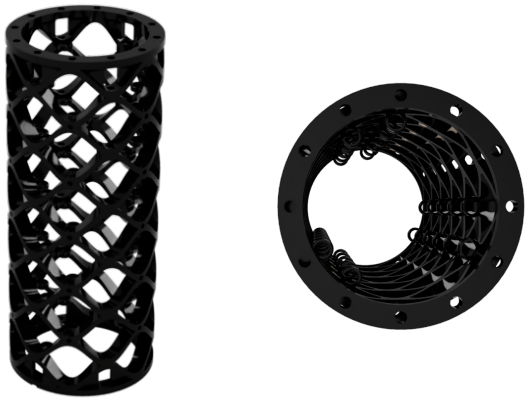


Figure 7.1: Detailed CAD view of a single soft module highlighting the routing holes.

A rigid PLA link was designed to connect the individual soft modules. An alternative version of this link was created specifically to house the IMU breakout boards. Both versions include protruding mounting bumps designed to securely attach the Motion Capture retro-reflective markers. A custom socket was also modeled to anchor the terminal ends of the Bowden cable sheaths. The various components were mechanically joined using standard screws and nuts.

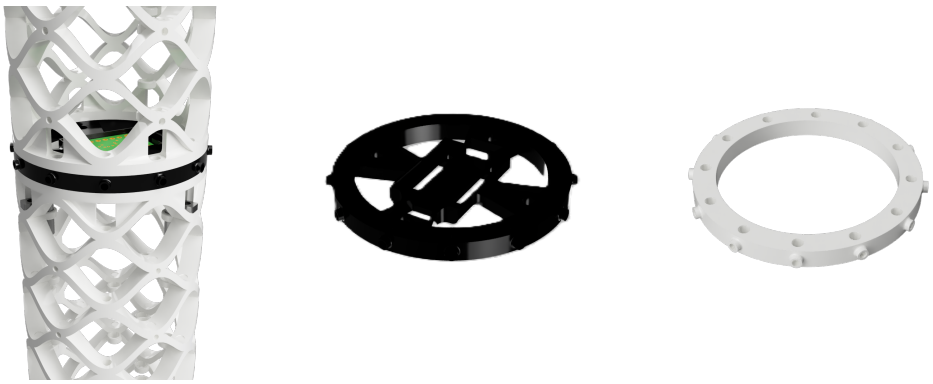


Figure 7.2: CAD models of the rigid connecting links, including the IMU housing version and the Bowden cable sockets.

7.1.2 Chassis

The main structural challenge for the chassis involved the spatial arrangement of the motors and Bowden cables. The tendons are secured to custom-designed 3D-printed motor pulleys, of radius $R_{motor} = 3\,\text{mm}$, using a simple fishing knot; that radius is what converts a motor angle into a tendon displacement in the kinematic mapping. Because the proximal first section is actuated directly via tendons rather than through Bowden

cables, a dual-layer motor layout was implemented: the three motors on the bottom tier directly drive the first section, while the six motors on the upper tier drive the second and third sections through the Bowden cables. A rigid guiding tube protector was integrated to secure the Bowden cables and prevent buckling under compressive loads. These cables pass freely through the lower helicoid rings until they are anchored at their respective targeted sockets.

Finally, a solid PLA structure, composed of multiple interlocking parts and featuring a dedicated mounting bracket for the U2D2 communication board, was designed to create a robust, compact base for the bimanual system. To successfully print these large structural PLA components, the part cooling fan speed was reduced and a wide brim was utilized to mitigate thermal warping.



Figure 7.3: Detailed CAD view of the bimanual chassis, illustrating the dual-layer motor arrangement and Bowden cable routing.

7.1.3 Assembly

Following fabrication, all mechanical and electronic components were assembled and secured using metal fasteners. Figure 7.4 illustrates key stages of the physical assembly process, while Figure 7.5 provides a direct comparison between the final CAD rendering and the fully assembled bimanual robotic platform.

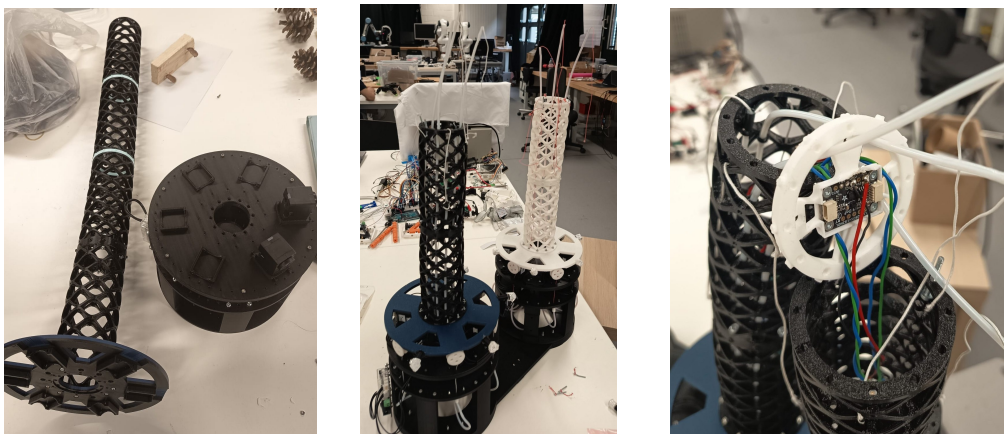


Figure 7.4: Key stages of the physical assembly process.

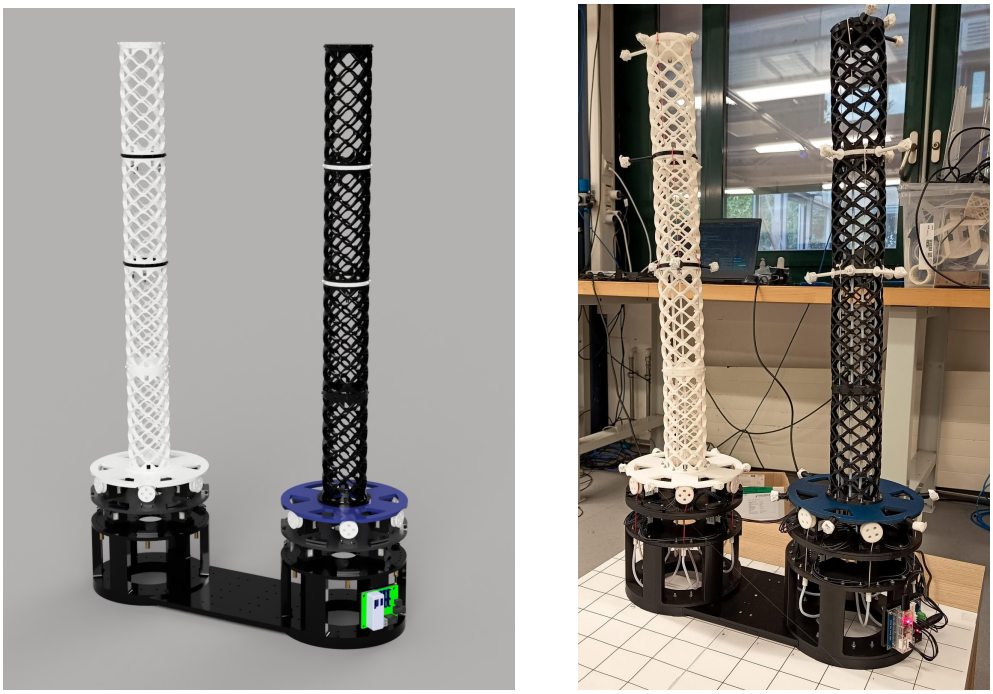


Figure 7.5: Final CAD render (left) and the physically assembled bimanual robot (right)

Chapter 8

Kinematics Implementation

The fundamental theoretical concepts regarding the state parametrization of soft continuous robots were established in Chapter 3. This chapter provides an in-depth analysis of the practical and computational implementation of the kinematics block for the physical platform.

8.1 Tendon-to-Configuration Mapping for a Three-Tendon Section

The singularity-free state parametrization adopted in this work is $\mathbf{\Delta}_i = [\Delta_{x,i}, \Delta_{y,i}, \delta L_i]^T$. It was originally proposed for a soft segment actuated by four orthogonal tendons [Della Santina et al., 2020a]. However, the hardware design developed for this bimanual system employs exactly three tendons per section, arranged radially at 120°. This architectural choice dictates a fundamental shift in the kinematic mapping.

Let the scalar curvature magnitude be defined as $\Delta_i \triangleq \sqrt{\Delta_{x,i}^2 + \Delta_{y,i}^2} = \theta_i d$, which will be extensively utilized in the spatial transformations. Utilizing three independently actuated tendons results in an exactly actuated segment, establishing a one-to-one mapping between the actuator space and the configuration space.[1] Note that a four-tendon design yields a non-square 4×3 mapping requiring a least-squares pseudo-inverse; the three-tendon layout instead admits an exact, square inversion.

While each individual segment is exactly actuated, the complete arm, comprising three segments (9 motors), is highly redundant. Considering that the Virtual Model Control operates in a 3D positional task space (3 DOFs), the arm possesses 6 degrees of kinematic

[1] Kinematically, the mapping is exactly determined. However, dynamically, tendons can only transmit pulling forces (unilateral constraint). The one-to-one mapping holds true as long as all tendons remain in tension, a condition ensured by the pre-tensioning term in the control architecture.

redundancy. This allows the manipulator to reconfigure its internal shape and place virtual interactive elements along its body without altering the end-effector position.

To extract the configuration parameters from the motor displacements, a geometric projection is derived. Let $d = 30$ mm be the constant radial distance of the tendons from the segment's neutral axis. For a generic section, the tendons are routed at angles $\gamma_{1,i}, \gamma_{2,i}, \gamma_{3,i}$ with respect to the local x-axis. Assuming the segment bends along a direction ϕ, the distance of each tendon from the undeformed neutral axis is equivalent to the orthogonal projection of the tendon's position onto the bending direction, as illustrated in Figure 8.1. The shortening of the j-th tendon due to bending alone is given by $\theta_i d \cos(\phi - \gamma_{j,i})$. Taking into account the pure axial elongation of the central axis δL_i, the total change in length for the j-th tendon is:

$$\Delta L_{j,i} = \delta L_i - \theta_i d \cos(\phi - \gamma_{j,i}) \tag{8.1}$$

By expanding the cosine term and substituting the state parameters ($\Delta_{x,i} = \theta_i d \cos\phi$ and $\Delta_{y,i} = \theta_i d \sin\phi$), the change in length becomes:

$$\Delta L_{j,i} = \delta L_i - \Delta_{x,i} \cos(\gamma_{j,i}) - \Delta_{y,i} \sin(\gamma_{j,i}) \tag{8.2}$$

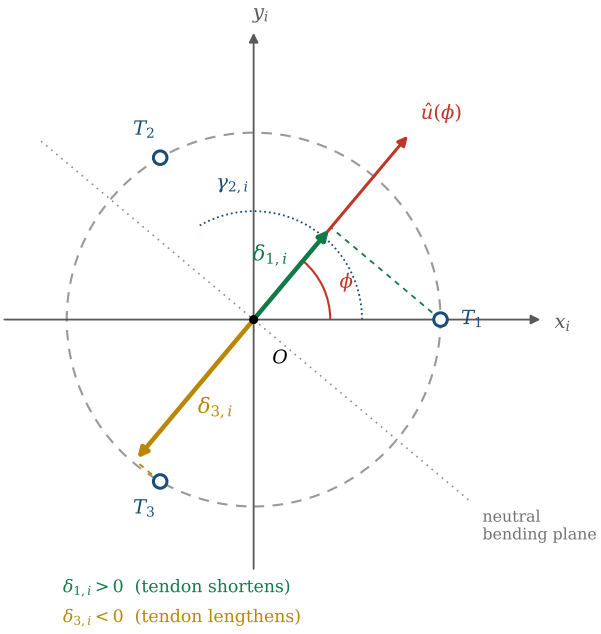


Figure 8.1: Geometric projection of the tendons' positions onto the bending direction. The section is seen from above: the three tendons sit at a radius d from the neutral axis O, at the routing angles $\gamma_{j,i}$.

Applying this relation to all three tendons generates a linear system governed by a 3×3 transformation matrix A_i:

$$\begin{bmatrix} \Delta L_1 \\ \Delta L_2 \\ \Delta L_3 \end{bmatrix}_i = \underbrace{\begin{bmatrix} -\cos(\gamma_{1,i}) & -\sin(\gamma_{1,i}) & 1 \\ -\cos(\gamma_{2,i}) & -\sin(\gamma_{2,i}) & 1 \\ -\cos(\gamma_{3,i}) & -\sin(\gamma_{3,i}) & 1 \end{bmatrix}}_{A_i} \begin{bmatrix} \Delta_x \\ \Delta_y \\ \delta L \end{bmatrix}_i \tag{8.3}$$

Due to physical routing requirements through the upstream modules, the tendons are not aligned identically across the three sections. The routing angles exhibit systematic offsets, detailed in Table 8.1.

Section (i)	$\gamma_{1,i}$	$\gamma_{2,i}$	$\gamma_{3,i}$
1	0°	120°	240°
2	60°	180°	300°
3	30°	150°	270°

Table 8.1: Tendon routing angles for each continuum section.

The offsets are rigid rotations: the relative 120° symmetry is preserved within each section, so the three matrices are equally well conditioned and each is inverted exactly.

Finally, the variation in tendon length is kinematically bound to the motor angle $\boldsymbol{q}_i$ through the pulley radius $R_{motor} = 3$ mm. Adopting the controller's sign convention, in which a positive motor rotation (counter-clockwise in the Dynamixel frame) tightens the tendon, i.e. shortens its free length ($\boldsymbol{\Delta L}_i = -R_{motor}\boldsymbol{q}_i$), the full mapping from actuator to PCC configuration space is:

$$\boldsymbol{\Delta}_i = A_i^{-1}\boldsymbol{\Delta L}_i = -R_{motor}A_i^{-1}\boldsymbol{q}_i \tag{8.4}$$

8.2 Forward Kinematics and Arc-Length Parametrization

Once the state variables $\boldsymbol{\Delta}_i$ are extracted, the Forward Kinematics evaluates the Cartesian pose in 3D space. To evaluate the position at any arbitrary point along the arm, a global arc-length parameter $s \in (0,1]$ is introduced. Based on the resting lengths of the sections ($L_{0,1} = 289.2\,\text{mm}$, while $L_{0,2} = L_{0,3} = 144.6\,\text{mm}$), the arm is normalized into intervals: $s \in (0, 0.5]$ for Section 1, $(0.5, 0.75]$ for Section 2, and $(0.75, 1.0]$ for Section 3. The parameter s coincides with the fractional arc-length *at rest*, which serves as a highly accurate approximation when axial elongations δL_i are small.

The spatial transformation relies on the closed-form matrices R_{i-1}^i and t_{i-1}^i derived in Chapter 3 (Equations 3.2 and 3.3), and the position of any point is obtained by cascading them along the three sections.

For the bimanual architecture, the local coordinates are finally mapped into a shared world frame:

$$p_{world} = t_{wb} + R_{wb} \cdot p_{local} \tag{8.5}$$

The two arms are mounted side by side on the same chassis and with the same orientation, so no relative rotation is involved: $R_{wb} = I$ for both, and the transformation reduces to the $\pm 125\,\text{mm}$ shoulder offset along x that separates the two bases. Figure 8.2 shows the frames on the platform on CAD.

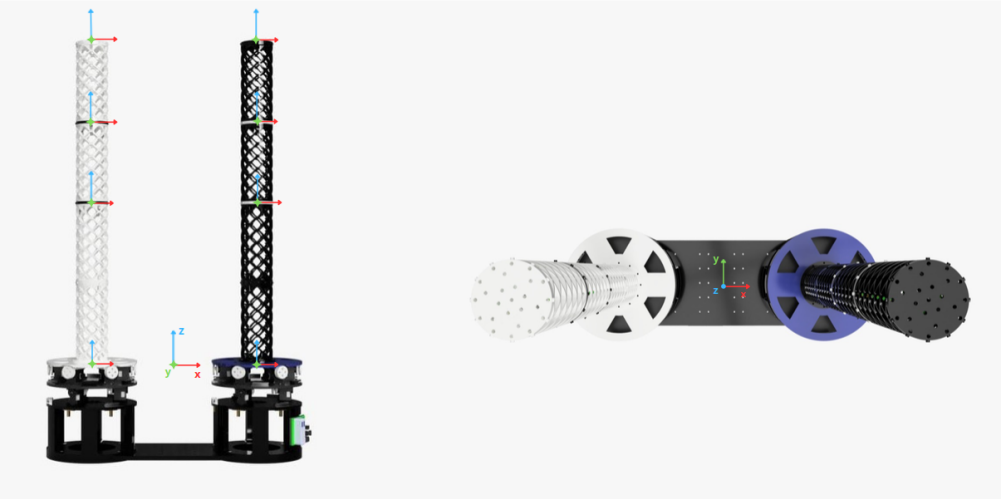


Figure 8.2: The world frame and the two base frames on the platform on CAD. The bases differ only by the $\pm 125\,\text{mm}$ offset along x; their axes are parallel, so $R_{wb} = I$.

8.3 Differential Kinematics and Computational Implementation

The Differential Kinematics translates joint velocities into Cartesian task-space velocities ($\dot{x} = J\dot{\boldsymbol{q}}$), where $J \in \mathbb{R}^{3\times 9}$ is a purely positional Jacobian matrix. To achieve real-time execution within the 500 Hz control loop, the software architecture exploits the chain rule:

$$J = \frac{\partial p}{\partial \boldsymbol{q}} = \underbrace{\frac{\partial p}{\partial \boldsymbol{\Delta}}}_{3\times 9,\ \text{nonlinear}} \underbrace{\frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{q}}}_{J_\Delta,\ 9\times 9,\ \text{constant}} \tag{8.6}$$

8.3.1 Actuator to Configuration Space Mapping

Because the mapping from motor space to PCC configuration space is strictly linear, the partial derivative yields a constant block-diagonal matrix $J_\Delta \in \mathbb{R}^{9\times 9}$:

$$J_\Delta \triangleq \frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{q}} = \text{blkdiag}\left(-R_{motor}A_1^{-1},\ -R_{motor}A_2^{-1},\ -R_{motor}A_3^{-1}\right) \tag{8.7}$$

Crucially, because the map is linear, J_Δ serves simultaneously as the velocity mapping ($\dot{\boldsymbol{\Delta}} = J_\Delta \dot{\boldsymbol{q}}$) and the positional mapping ($\boldsymbol{\Delta} = J_\Delta \boldsymbol{q}$). This ensures that both the configuration and its rate are obtained algebraically: the former requires no integration, the latter no numerical differentiation.

8.3.2 Configuration to Task Space Mapping

To handle the algebraic complexity of differentiating the concatenated spatial equations, a symbolic computation approach was employed utilizing the *SymPy* Python library. The chain rule composition is not evaluated as a runtime matrix product. Instead, the forward kinematics is built symbolically as an explicit function of the actuator variables $\boldsymbol{q}$, and the differentiation is carried out once, with respect to $\boldsymbol{q}$ directly. The symbolic engine applies the chain rule internally, and the resulting routine returns the full $\frac{\partial p}{\partial \boldsymbol{q}}$ matrix in a single evaluation.[2]

Listing 8.1 illustrates the architectural core of this approach. The entire symbolic cost is paid once, when the controller starts: `lambdify` turns the symbolic matrix into a plain numerical function, with common sub-expressions factored out. During active operation each control cycle merely substitutes the current configuration estimate into that function, leaving the 500 Hz cycle budget essentially untouched.

```
def q_motor2pose(s_param):
    # Initialize symbolic variables
    q_vec = Matrix(symbols("q11:14 q21:24 q31:34"))

    # Actuator -> PCC, Eq. (8.4): this is where q enters the chain
    pcc = [Matrix(A_INV[i]) * (-q_vec[3*i:3*i+3, 0] * R_MOTOR)
           for i in range(3)]

    # Cascade the three sections; s_param truncates the last one
    t1 = t_vector(*pcc[0], L0_1)
    t2 = t_vector(*pcc[1], L0_2)
    t3 = t_vector(*pcc[2], L0_3, local_s)
    t  = t1 + R_MATRIX(*pcc[0]) * (t2 + R_MATRIX(*pcc[1]) * t3)

    # Extract exact analytical derivatives in one shot
    J = t.jacobian(q_vec)  # (3, 9) positional Jacobian

    # Turn it into a fast numerical function (built once, at start-up)
    return lambdify(tuple(q_vec), J, modules=['numpy'], cse=True)
```

Listing 8.1: Symbolic generation of the spatial analytical Jacobian.

[2]The generated Jacobian was verified against a finite-difference approximation of the forward kinematics across the whole arc-length range, with a maximum residual of $3.4 \cdot 10^{-9}$ m/rad, i.e., at the truncation limit of the check itself.

Chapter 9

State Estimation

The motor encoders provide actuator positions, which were directly related to the robot's configuration in the previous chapters. Theoretically, Virtual Model Control (VMC) can be driven entirely by the position of any point on the soft arm computed through the Forward Kinematics pipeline. However, a purely proprioceptive approach relies on the assumption of an ideal kinematic model and is susceptible to unmodeled mechanical phenomena, such as tendon friction, material hysteresis, and external disturbances such as obstacles. To obtain a highly accurate and robust state estimate, the system must fuse multiple sensory sources.

9.1 Sensors

To achieve reliable state estimation, exteroceptive sensors such as a Motion Capture (MoCap) system and Inertial Measurement Units (IMUs) were integrated into the design. The software architecture was specifically developed to be sensor-agnostic: every sensor converts its raw data into the PCC curvature space $(\boldsymbol{\Delta}, \dot{\boldsymbol{\Delta}})$ before feeding it to the estimation filter.

Although the mechanical structure accommodates IMUs at each section tip and the estimator architecture exposes the interface for their integration, the final fusion algorithms involving IMU data have been left to future work. The current state estimation relies entirely on sensor fusion between the motor encoders and the Motion Capture system. The integration of IMUs aims to eventually replace the optical MoCap system, enabling autonomous operation outside structured laboratory environments.

9.1.1 Encoders

As detailed in the kinematics implementation chapter, the high-resolution encoders embedded in the Dynamixel motors provide the joint-space variables $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. Through the constant analytical matrix J_Δ, these readings are linearly mapped into the curvature

space without numerical approximation:

$$\boldsymbol{\Delta}_{enc} = J_{\Delta}\boldsymbol{q}, \qquad \dot{\boldsymbol{\Delta}}_{enc} = J_{\Delta}\dot{\boldsymbol{q}} \tag{9.1}$$

The map is exact, the measurement is not: the velocity is differentiated and quantized inside the motor firmware, then smoothed by the acquisition node. Two properties matter for the fusion. The kinematic chain adds no differentiation of its own, unlike the MoCap pipeline; but the encoders remain blind to any discrepancy between the tendon-implied configuration and the arm's actual physical shape.

9.1.2 Motion Capture

The Motion Capture system provides a high-accuracy exteroceptive measurement of the arm's posture in 3D space.

Hardware Setup and Data Streaming

Tracking is performed using an OptiTrack system running the Motive software suite, publishing data at 120 Hz. The setup comprises six cameras, re-calibrated—including ground-plane definition—whenever repositioned. Markers were fabricated by applying infrared retro-reflective tape to 3D-printed spherical supports. Four distinct rigid bodies per arm were defined: one for each section tip, plus one rigidly attached to the base chassis.

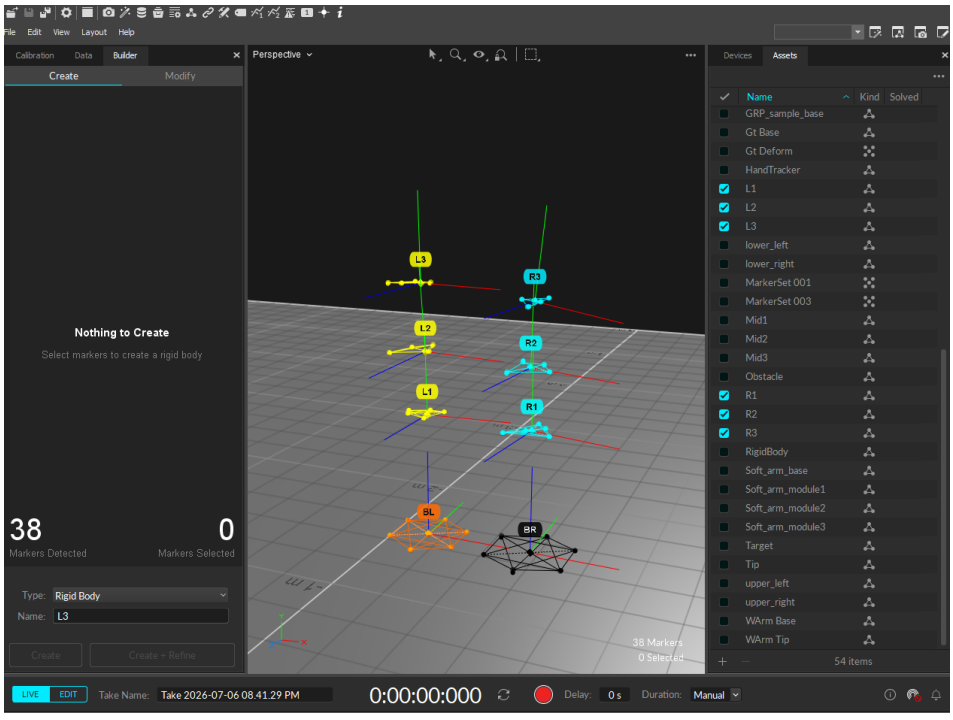


Figure 9.1: The Motive Software during a session

Asymmetry is crucial in optical tracking to prevent the software from confusing or swapping rigid bodies when the soft arm twists or bends upon itself. Therefore, each rigid body is identified by a unique, asymmetric arrangement of markers (utilizing as many markers as physically viable). The pose data of each rigid body is continuously streamed to the main control architecture via a dedicated ROS 2 interface.

Kinematic Inversion and Velocity Estimation

Unlike the encoders, the MoCap system provides Cartesian data. While the VMC requires Cartesian positions, the rigid bodies only track the tips of each section. To evaluate the VMC at any arbitrary point along the arm, the Cartesian data must be inverted to obtain the PCC curvature state $\mathbf{\Delta}_{mocap}$. The software pipeline executes this conversion sequentially:

1. **Reference Frame Transformation:** The raw rigid body poses are captured in the OptiTrack global world frame. Before any kinematic computation, the Cartesian positions of the section markers are roto-translated into the robot's local base frame (p_{local}) using the real-time orientation R_{base} and position p_{base} of the base rigid body:
$$p_{local} = R_{base}^{T} \left(p_{marker} - p_{base} \right) \tag{9.2}$$
Because this transformation relies on a live stream rather than fixed static calibration, it renders the estimation immune to unexpected physical shifts of the entire robotic setup.

2. **Closed-Form Geometric Inversion:** Because each marker is located at the tip of its respective section, the nine-parameter inversion decomposes into three independent three-parameter problems solved in cascade. As demonstrated in Chapter 3, the Cartesian tip position expressed in the section's local base frame allows for the analytical extraction of the section's curvature parameters $(\Delta, \Delta_x, \Delta_y, \delta L)$. By solving this inverse kinematics analytically for the three sections in sequence, evaluating the Forward Kinematics on the recovered state $\mathbf{\Delta}_{mocap}$ exactly reproduces the measured rigid body positions. This closed-form formulation requires no iterations and is completely immune to singularity-induced convergence failures.

3. **Velocity Estimation:** The OptiTrack system only provides positional data. The configuration velocity $\dot{\mathbf{\Delta}}$ is estimated numerically via finite-differences between consecutive observations:
$$\dot{\mathbf{\Delta}}_{raw}^{k} = \frac{\mathbf{\Delta}_{mocap}^{k} - \mathbf{\Delta}_{mocap}^{k-1}}{dt} \tag{9.3}$$
where $dt = 1/500\,\mathrm{s}$ is the period of the control loop. Because optical tracking noise is amplified by numerical differentiation, the signal is smoothed using a first-order Low-Pass (LP) filter:
$$\dot{\mathbf{\Delta}}_{mocap}^{k} = (1 - \beta)\dot{\mathbf{\Delta}}_{raw}^{k} + \beta\dot{\mathbf{\Delta}}_{mocap}^{k-1} \tag{9.4}$$

where $\beta = 0.3$ weights the previous estimate. Note that the difference is taken at the 500 Hz control rate while the optical stream updates at 120 Hz: the raw quotient is therefore zero on most ticks and spikes on the one carrying a new frame. The filter recovers the correct mean rate, but the MoCap velocity remains intrinsically noisier than the encoder-derived one—a fact that directly determines the covariances of Section 9.2.2.

9.2 Kalman Filter

The Kalman Filter, running at 500 Hz, fuses information from the different sensors to provide a robust estimate of the soft arm's state.

9.2.1 General Kalman Filter Algorithm

In probabilistic robotics [Thrun et al., 2005], the estimated state at time t is represented by its mean vector μ_t and its covariance matrix Σ_t. The algorithm operates recursively in two steps:

Prediction: The state is propagated forward in time using the system's dynamic model:

$$\bar{\mu}_t = A\mu_{t-1} + Bu_t \tag{9.5}$$

$$\bar{\Sigma}_t = A\Sigma_{t-1}A^T + Q \tag{9.6}$$

where A and B are the discrete state and input matrices, u_t is the control input, and Q is the process noise covariance matrix.

Correction: The predicted state is updated using a new measurement z_t:

$$K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1} \tag{9.7}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t) \tag{9.8}$$

$$\Sigma_t = (I - K_t C)\bar{\Sigma}_t \tag{9.9}$$

where C is the observation matrix mapping the state to the measurement space, R is the measurement noise covariance, and K_t is the Kalman gain. The term $(z_t - C\bar{\mu}_t)$ is the *innovation*, representing the discrepancy between the actual measurement and the expected one.

9.2.2 Filter Tuning and Covariances

The filter weights each source according to its statistical confidence, defined by the measurement covariance matrix R. The two modalities are complementary in the way just described, and the covariances of Table 9.1 simply follow that asymmetry: in position the MoCap is trusted well above the encoders, since it observes the spatial shape rather

than the tendon lengths, while in velocity the ordering reverses by two orders of magnitude. That reversal is not arbitrary tuning but a consequence of how each signal is constructed.

Matrix	Value	Quantity
Q	$10^{-6}\, I_{18}$	process noise, per step
R_{enc}	$10^{-3}\, I_9$	encoder position
$R_{enc;}$	$5 \cdot 10^{-4}\, I_9$	encoder velocity
R_{mocap}	$5 \cdot 10^{-5}\, I_9$	MoCap position
$R_{mocap;}$	$5 \cdot 10^{-2}\, I_9$	MoCap velocity

Table 9.1: Default covariances of the state estimator.

9.2.3 Prediction Model

To develop the Kalman filter for this architecture, the prediction matrices A and B must be extracted from the physical model. Because all sensors convert their raw readings into the curvature space ($[\boldsymbol{\Delta}, \dot{\boldsymbol{\Delta}}]^T \in \mathbb{R}^{18}$), and because the elastic and dissipative terms of the arm's dynamics are linear with respect to $\boldsymbol{\Delta}$ and $\dot{\boldsymbol{\Delta}}$, the system is implemented as a Linear Time-Variant Kalman Filter (LTV-KF) in that space. The mass matrix $M(\boldsymbol{\Delta})$ is recomputed sequentially, making the state matrices time-dependent.

The dynamic model is derived from a basic Newton-Euler balance. Let $F_{applied}$ be the generalized active force applied to the robot. The motor torques $\boldsymbol{\tau}$ are mapped to this space via the transposed Jacobian:

$$F_{applied} = J_{\Delta}^{-T}(\eta\boldsymbol{\tau}) \tag{9.10}$$

where η accounts for the transmission efficiency. The internal resistance of the soft material opposes motion with a force modeled by the linear stiffness K and damping D:

$$F_{resistive} = K\boldsymbol{\Delta} + D\dot{\boldsymbol{\Delta}} \tag{9.11}$$

According to the dynamic balance, the difference between the applied active force and the resistive force generates the inertial acceleration:

$$M(\boldsymbol{\Delta})\ddot{\boldsymbol{\Delta}} + D\dot{\boldsymbol{\Delta}} + K\boldsymbol{\Delta} = J_{\Delta}^{-T}(\eta\boldsymbol{\tau}) \tag{9.12}$$

The only configuration-dependent term in this linear structure is the inertia matrix $M(\boldsymbol{\Delta})$, which is recomputed at every step using a lumped-mass model (placing one mass at the center of each section's arc length).

To cast Equation 9.12 into the standard continuous state-space form $\dot{x} = A_c x + B_c u$, the state vector (representing the mean μ) is defined as $x = [\boldsymbol{\Delta}, \dot{\boldsymbol{\Delta}}]^T$, and the control input as $u = \eta\boldsymbol{\tau}$. Solving the dynamic balance explicitly for the acceleration yields:

$$\ddot{\boldsymbol{\Delta}} = -M^{-1}K\boldsymbol{\Delta} - M^{-1}D\dot{\boldsymbol{\Delta}} + M^{-1}J_{\Delta}^{-T}u \tag{9.13}$$

The time derivative of the state vector is $\dot{x} = [\dot{\mathbf{\Delta}}, \ddot{\mathbf{\Delta}}]^T$. Grouping the trivial identity relation $\dot{\mathbf{\Delta}} = I\dot{\mathbf{\Delta}}$ with the derived acceleration equation yields the full matrix block representation:

$$\underbrace{\begin{bmatrix}\dot{\mathbf{\Delta}}\\ \ddot{\mathbf{\Delta}}\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}0 & I\\ -M^{-1}K & -M^{-1}D\end{bmatrix}}_{A_c}\underbrace{\begin{bmatrix}\mathbf{\Delta}\\ \dot{\mathbf{\Delta}}\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0\\ M^{-1}J_{\Delta}^{-T}\end{bmatrix}}_{B_c}\underbrace{\eta\boldsymbol{\tau}}_{u} \tag{9.14}$$

The Kalman Filter operates in discrete steps ($dt = 1/500\,\mathrm{s}$). To avoid numerical drift, the software implements an exact analytical discretization. Assuming the motor torque remains constant during a single interval (Zero-Order Hold), the continuous differential equation is solved exactly using the matrix exponential:

$$A = \exp(A_c\,dt), \qquad B = A_c^{-1}\,(A - I)\,B_c \tag{9.15}$$

This step-invariant integration ensures the controller and the physical plant share the exact same dynamic assumptions.

9.2.4 Initialization, Correction, and Innovation Gate

Filter Initialization. The initial state mean μ_0 is set to the first valid sensor reading, and the initial covariance Σ_0 is initialized as a sufficiently large diagonal matrix. This high initial uncertainty allows the Kalman gain to aggressively trust incoming measurements during the first few control cycles, rapidly converging to the true state.

Correction Model. The architectural strength of this state estimator lies in its sensor-agnostic approach. Because every measurement is converted into the exact curvature space format $(\mathbf{\Delta}, \dot{\mathbf{\Delta}})$ *before* the filter processes it, the observation matrix C is simply the 18×18 identity matrix ($C = I_{18}$). The state is *fully observed*, meaning the physical prediction model acts primarily as an advanced noise filter rather than an observer of hidden variables.

Innovation Gate. Before the correction update is applied, each incoming measurement is validated through the Mahalanobis distance of its innovation $v_t = (z_t - C\bar{\mu}_t)$:

$$d^2 = v_t^T \left(C\bar{\Sigma}_t C^T + R\right)^{-1} v_t \tag{9.16}$$

The measurement is discarded as an outlier if d^2 exceeds a threshold of 40, which is slightly tighter than the 99.9% critical value of the chi-squared distribution with 18 degrees of freedom (≈ 42.3).

A rejected measurement does not contribute to that step. Only when no measurement survives—due to a rejection in single-sensor mode, or a physical occlusion of the MoCap markers—does the state advance on the prediction alone.

Chapter 10

Virtual Model Control Implementation

Chapters 8 and 9 leave the controller with everything it needs at every tick: the configuration $\boldsymbol{\Delta}$ and its rate, and an analytical Jacobian at any point along the arm. This chapter assembles them into the active control law.

As established in Chapter 4, no inverse kinematics and no inverse dynamics enter that law. Virtual mechanical elements—springs and dampers—are attached to chosen points of the soft body; they generate a force in the Cartesian space, and the transposed Jacobian converts it into the tendon torques required to reproduce it. The whole law runs as a single ROS 2 node at 500 Hz, driving both arms within the same computational tick so that they stay synchronised.

10.1 From a virtual element to motor torques

A virtual element is defined by three quantities: a force law (the spring), a dissipative law (the damper), and the arc-length coordinate $s \in (0,1]$ at which both are anchored to the backbone.

Let $p(s, \boldsymbol{q}) \in \mathbb{R}^3$ be the position of the anchor point and p_d the position of its goal, both expressed in the shared world frame of the bimanual platform. The virtual element requests the Cartesian force, defined in the world frame:

$$F^w = \underbrace{F_K(p, p_d)}_{\text{spring}} + \underbrace{F_D(\dot{p})}_{\text{damper}} \tag{10.1}$$

This represents the general formulation of a linear spring and damper. Several specialized types of virtual elements can be utilized, which will be briefly described in Section 10.4.

Each arm has its own base frame, related to the world by the constant translation and rotation pair (R_{wb}, t_{wb}) defined in Chapter 8. The velocity must be converted into the world frame before being inserted into Equation (10.1):

$$\dot{p}^w = R_{wb}\, J(s, \boldsymbol{q})\, \dot{\boldsymbol{q}} \tag{10.2}$$

The force generated by the virtual element is described by Equation (10.1) in the world frame. However, since the Jacobian derived in Chapter 8 is a *local* operator $J(s, \boldsymbol{q}) \in \mathbb{R}^{3\times 9}$ defined in the arm's own base frame, the force must be rotated back into the base frame before multiplying the transposed Jacobian:

$$\boxed{\boldsymbol{\tau} = J(s, \boldsymbol{q})^T R_{wb}^T F^w} \tag{10.3}$$

Whole-body virtual elements. Any $s \in (0,1]$ yields a valid Jacobian, meaning a virtual element can be anchored anywhere along the backbone. This is precisely the feature that separates Virtual Model Control from conventional end-effector impedance control. With N elements anchored at different points along the arms, the total control torque is obtained through linear superposition: each element is evaluated independently through Equation (10.3) and the resulting torques are simply added, $\boldsymbol{\tau}_{vmc} = \sum_{k=1}^{N} J(s_k, \boldsymbol{q})^T R_{wb}^T F_k^w$.

10.2 Torque composition

The total virtual torque $\boldsymbol{\tau}_{vmc}$ is not yet the final command sent to the motors. Three further terms are added, and the overall result is safely saturated before the torque leaves the controller:

$$\boldsymbol{\tau} = \mathrm{sat}_{\tau_{lim}}\Big(\boldsymbol{\tau}_{vmc} + \boldsymbol{\tau}_g(\boldsymbol{q}) + \boldsymbol{\tau}_0 - \boldsymbol{\tau}_{pre}(\boldsymbol{q})\Big) \tag{10.4}$$

Each term addresses a distinct physical requirement: $\boldsymbol{\tau}_g$ removes the gravitational disturbance the virtual model should not have to fight, $\boldsymbol{\tau}_0$ and $\boldsymbol{\tau}_{pre}$ pretension the tendons, and the saturation enforces the hardware limits of the robot.

10.2.1 Gravity compensation

The soft arm slightly sags under its own weight, especially when the robot is moving. Gravity is therefore compensated in feedforward. Consistently with the lumped-mass inertia model adopted for the Kalman prediction step (Section 9.2.3), each section is represented by a point mass placed at the center of its arc length, $s = \left(\frac{1}{4}, \frac{5}{8}, \frac{7}{8}\right)$, with masses $(2m, m, m)$, where $m = 0.030\,\mathrm{kg}$. Each lumped mass requires a compensating

force equal and opposite to its weight, expressed in the arm's base frame, and mapped to motor torques by the exact same duality used for the virtual elements:

$$\boldsymbol{\tau}_g(\boldsymbol{q}) = \sum_{i=1}^{3} J(s_i, \boldsymbol{q})^T \left(- R_{wb}^T \, [\, 0,\ 0,\ -m_i g \,]^T \right) \tag{10.5}$$

This term is configuration-dependent—the Jacobian J is re-evaluated every cycle—so the compensation dynamically follows the arm as it bends.

10.2.2 Pretensioning and the unilateral constraint

The kinematic map developed in Chapter 8 assumes a perfect one-to-one correspondence between motor angles and configuration. This correspondence holds only while every tendon remains strictly taut. Otherwise, due to tendon slack, the actuator moves but the tendon does not. Consequently, the encoder reports angles that the kinematics erroneously converts into a curvature the arm does not physically possess, corrupting the state estimation.

Two specific terms keep the tendons in tension. A constant bias $\boldsymbol{\tau}_0 = 0.03\,\mathrm{N\,m}$ is applied to all nine motors for each arm. On top of this, a one-sided soft joint stop heavily resists any motor that unwinds past a safety threshold:

$$\tau_{pre,i} = \begin{cases} W_i \, (q_i + q_{th}) & q_i < -q_{th} \\ 0 & \text{otherwise} \end{cases} \qquad W_i = 0.03\,\mathrm{N\,m\,rad^{-1}}, \quad q_{th} = 30^\circ \approx 0.52\,\mathrm{rad} \tag{10.6}$$

Since a positive motor rotation tightens a tendon ($\Delta L = -R_{motor} q$), the critical regime to be penalized is $q \ll 0$.

10.2.3 Saturation

The final command is clipped component-wise to $\tau_{lim} = 0.5\,\mathrm{N\,m}$, keeping it comfortably below the $0.92\,\mathrm{N\,m}$ stall torque of the XC330 motors (Section 6.1.1). This safety margin was established by evaluating the structural limits of the 3D-printed TPU robot and the tensile strength of the tendons.

10.3 Default placement and gains

Virtual elements are dynamically configured at runtime through the user interface described in Section 10.6. They are initialized with baseline stiffness and damping values that depend on their longitudinal anchor point along the arm (Table 10.1). The values are empirical. A proximal anchor takes the largest gain ($K = 100$, $D = 4$): it sits close to the base, so a given force there displaces it far less than the same force applied distally,

and the element must pull harder to move its anchor at all. Both distal sections use lower gains; the third-section stiffness was raised from 35 to 45 because tuning showed the lower value insufficient for distal-point tracking, while both retain the same damping ($D = 2$).

Arc-length range	K [N/m]	D [N·s/m]
$s \leq 0.50$ (Section 1)	100	4
$0.50 < s \leq 0.75$ (Section 2)	35	2
$s > 0.75$ (Section 3)	45	2

Table 10.1: Default stiffness and damping assigned to a virtual element according to its anchor point.

10.4 Library of virtual elements

A comprehensive library of different types of virtual elements was implemented. Each element possesses a `compute_force` function that calculates the localized virtual force before it is plugged into Equation (10.3). The virtual elements defined in this library serve two primary functions: changing the *shape* of the force as a function of the positional error, and restricting the *geometric subspace* in which that force is allowed to act.

10.4.1 Force laws

Let $e = p_d - p$ be the position error and $d = \|e\|$ its absolute magnitude. The implemented laws are summarized in Table 10.2. Each gain may be supplied as a scalar, a vector (diagonal matrix), or a full matrix. This allows a single element to be made highly anisotropic—for instance, stiff along an approach vector but entirely compliant laterally—without changing its underlying force law.

Law	Force	Purpose
Linear	$F = Ke$	Baseline attraction
Tanh	$F = F_{max}\tanh\big(Ke/F_{max}\big)$	Bounded force, unchanged near $e = 0$
Sigmoid	$F = k(d)\,e, \quad k(d) = k_{min} + \dfrac{k_{max} - k_{min}}{1 + e^{-\alpha(d-d_{th})}}$	Soft near the goal, firm far away
Polynomial	$F = K\big(d/d_0\big)^n e$	Progressive stiffening
Gaussian	$F = A\,e^{-d^2/2\sigma^2}\,\delta, \quad \delta = p - p_d$	Repulsion / obstacle avoidance

Table 10.2: Implemented spring and damper laws. Dampers follow the identical pattern operating on velocity: $F = -D\dot{p}$ for the linear damper and $F = -F_{max}\tanh(D\dot{p}/F_{max})$ for its saturated counterpart.

Two of these laws deserve an explicit technical comment:

The **tanh** law is the natural answer to a critical hardware constraint. A linear spring produces a force strictly proportional to the error without bound; a goal placed far outside the workspace generates a torque command the motors cannot physically deliver, which the torque saturation of Section 10.2.3 would eventually truncate in an uncontrolled, non-linear way. The tanh law elegantly caps the force at F_{max} per component while preserving the local slope K at the origin. Thus, the behavior near the goal—where tracking accuracy is decided—is identical to the linear case, and only the far-field forces are safely limited.

The **Gaussian** law is the only actively repulsive element of the library, specifically utilized to turn a spatial coordinate into an obstacle. Its magnitude is $\|F\| = A\,d\,e^{-d^2/2\sigma^2}$: identically zero exactly *at* the marked point, reaching a maximal repulsion at a distance σ where it peaks at $A\sigma e^{-1/2} \approx 0.607\,A\sigma$, and becoming negligible beyond 3σ.

10.4.2 Constrained elements

Virtual elements that geometrically restrict the subspace in which a force may act are referred to as constrained elements. Given a unit normal vector n and the associated projector matrix $P = nn^T$, the constrained variant of a law computes:

$$F = P\,K\,P\,e \tag{10.7}$$

The construction is worth unpacking, since the same projector returns in Chapter 11.

For any vector v, $Pv = n\,(n^T v)$: the inner product extracts the component of v along n, and multiplying by n puts it back as a vector in that direction alone. Everything orthogonal to n is discarded. Applying it on both sides of K in Equation (10.7) therefore restricts both what the element reads and what it applies to the line spanned by n.

This methodology is applied analogously to the tanh, Gaussian, and damping laws. The element then behaves as a cart on a rail: it exerts force strictly along n and leaves the orthogonal plane completely free, guaranteeing the arm's compliance in the unconstrained directions is perfectly preserved.

10.5 Smooth transitions

A virtual model is mathematically a static map: given a goal and a state, it outputs a force. Every discontinuity in its inputs (e.g., a newly dragged goal, an updated gain, a change of feedback source, or a reconfiguration of the control points) appears instantaneously at the output as a violent step in torque. On a soft, lightly damped structure, such discrete steps excite the exact resonant oscillations the inherent compliance is meant to absorb. To mitigate this, three filtering mechanisms were integrated into the architecture.

10.5.1 Moving goal filter

The commanded target coordinate is never fed to the virtual spring directly. Instead, each element actively chases an internal *moving goal* g, which is driven towards the user's commanded position g^* by a first-order discrete filter. Without this filter, an instantaneously displaced goal of $\Delta d = 5\,\mathrm{cm}$ with a stiffness of $K = 45\,\mathrm{N\,m^{-1}}$ injects $2.25\,\mathrm{N}$ in a single $2\,\mathrm{ms}$ tick, instantly saturating the torque limit.

$$g_{k+1} = g_k + \alpha\,(g^* - g_k), \qquad \tau_{goal} = \frac{\Delta t}{\alpha} = 0.2\,\mathrm{s} \quad \text{for } \alpha = 0.01,\ \Delta t = 2\,\mathrm{ms}. \tag{10.8}$$

The filter rate is exposed as a live tunable parameter. At the default $\alpha = 0.01$, the internal goal smoothly covers a stepped displacement in roughly $0.5\,\mathrm{s}$.

10.5.2 Gain ramping

The stiffness and damping matrices commanded from the interface are treated identically. Each element holds a smoothed active pair (K, D), which ramps towards the newly commanded targets using the same first-order exponential law with a rate of $\alpha_{gain} = 0.01$. The physical motivation is identical in kind: a massive gain applied while a non-zero positional error is present changes the resulting force by $\Delta K \cdot e$ instantaneously. A step from $K = 5$ to $K = 30$ with a few centimeters of error is enough to produce an underdamped mechanical jolt.

10.5.3 Bumpless transfer

Two distinct events can change the entire physical context of the control law from one tick to the next: a switch of the active feedback source, or a runtime geometric reconfiguration of the control points. In both cases, the arm's estimated position may jump discontinuously—the encoder-implied pose and the MoCap-measured pose differ by the exact tendon slack the estimator exists to reconcile—or the very set of virtual springs may be entirely replaced.

The controller handles both scenarios with a unified safety policy: every internal moving goal is instantly snapped onto the *current* estimated position of its anchor, and every stiffness is reset to a safe baseline of $K = 5\,\mathrm{N\,m^{-1}}$. The spring error is thus identically zeroed at the exact instant of the contextual change, and the gains then ramp safely back up under Equation (10.8) as the goals travel smoothly back to their commanded positions. This represents a classical bumpless transfer methodology, successfully adapted to a task-space virtual model instead of a conventional PID's integral state.

10.6 Graphical user interface

Operating a highly redundant bimanual soft robot exclusively through configuration files is highly impractical. A virtual model is defined by where its elements sit, what laws they follow, and how stiff they are; all three are dynamic quantities an operator needs to interactively tune while watching the physical arm respond. A dedicated Graphical User Interface (GUI) was therefore developed (Figure 10.1), built entirely on the ROS 2 topics summarized in Table 10.3.

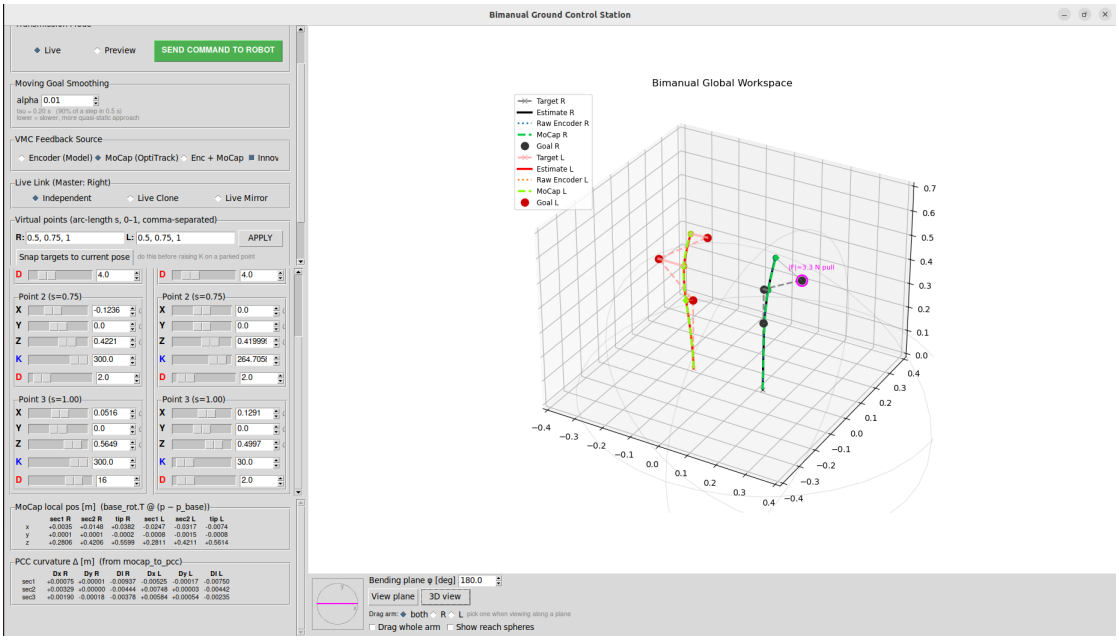


Figure 10.1: Graphical user interface featuring a real-time digital twin of the bimanual system, per-element gain sliders, a control-point spatial editor, and a feedback-source selector.

Topic	Direction	Content
`/goal_simulation`	GUI → ctrl	Goal positions, all elements
`/vmc_gains`	GUI → ctrl	(K, D) per element
`/vmc_set_points`	GUI → ctrl	Arc-lengths s_k (reconfiguration)
`/vmc_spring_types`	GUI → ctrl	Per element: attractive / repulsive σ
`/change_feedback_source`	GUI → ctrl	Encoder / MoCap / both
`/use_moving_goal`, `/alpha_goal`	GUI → ctrl	Reference filter toggles and rate
`/vmc_config`	ctrl → GUI	Current active element layout
`/debug_vmc_*`	ctrl → GUI	Measured and target positions

Table 10.3: Main interface topics. The main controller constantly republishes its active configuration so that the GUI dynamically rebuilds itself around the running layout.

The central widget is a real-time interactive digital twin. The backbones of both arms (continuous lines) are continually reconstructed by evaluating the forward kinematics derived in Chapter 8 on the actively estimated state. They are rendered in the 3D space alongside the control points (dots), their moving goals, and the raw optical MoCap markers (green dashed lines). Displaying the physically measured markers adjacent to the reconstructed kinematic shape makes any discrepancy between the mathematical

model and physical reality immediately visible, serving as a critical diagnostic tool during development.

Goals are manipulated by dragging them directly within the 3D view. Because a 2D mouse provides only two coordinates while the physical goal exists in a 3D space, the drag action is constrained to a vertical plane whose orientation the operator selects, perfectly matching how a soft section actually deforms: a specific bending plane coupled with an angle. The remaining panels expose the control-point editor (which triggers the reconfiguration and bumpless reset of Section 10.5), per-element sliders for K and D, the attractive/repulsive toggles, the feedback-source selector, and the moving-goal rate.

10.6.1 Force modulation panel

A dedicated panel for managing the energy-tank limits and force-modulation was developed. Launched with the `--force_mod` flag, the interface docks a further strip on the right of the 3D view (Figure 10.2). It exposes the parameters of the force-regulation layer developed in Chapter 11, and it is the panel through which every experiment of Chapter 14 was driven. What follows describes what it controls; the meaning of the quantities is the subject of the next chapter.

Contact and objective. The first two groups select which arm is in contact and which parameter the descent is allowed to move — the reference d_{ref}, the stiffness K_d, or both, the three strategies compared in Section 11.2. The third sets the objective itself: a scalar force level and the three components of the contact normal $\hat{n}$, normalised automatically to a unit vector. The force being regulated is $f = \text{level} \cdot \hat{n}$, so the panel commands a magnitude and a direction separately rather than a Cartesian vector.

Step size. This group exposes the three bounds that limit each descent step: the cap on the predicted force change per iteration, the divisor applied to the passivity bound, and the two geometric caps on the reference and stiffness increments. The panel displays the rule combining them, $\alpha = \min(\alpha_{pass}/s_f,\ \alpha_F,\ \alpha_{geom})$, so that the operator can see which of the three is binding at any moment; all three are also logged.

Calibration. The transmission factor η is set here, together with the two actions that establish the common zero required by the calibration protocol of Section 14.1: *Zero force* tares the model estimate at the current pose, and *Align ref to actual, then tare* first snaps the reference onto the measured pose and then tares, which is the sequence to use before enabling the descent.

Two further controls belong to closed-loop operation. A toggle activates closed-loop mode and a deadband sets the band within which the descent holds still instead of chasing.

Automatic sweep. The static calibration of Section 14.1 is driven from this group rather than by hand. It steps a chosen axis — one of the three position components, or the stiffness — by a fixed increment on a fixed timer, and stops itself if the measured force reaches a configurable limit.

Energy tank and status. The last two groups set the tank's initial charge and capacity and report the live state of the loop: the estimated force along $\hat{n}$, the load-cell reading when one is connected, the step size currently granted, and the tank level as a bar. A single *ACTIVE* switch at the bottom freezes both arms and starts the descent; until it is enabled the panel only observes, which is what allows a calibration sweep to run with the adaptation off.

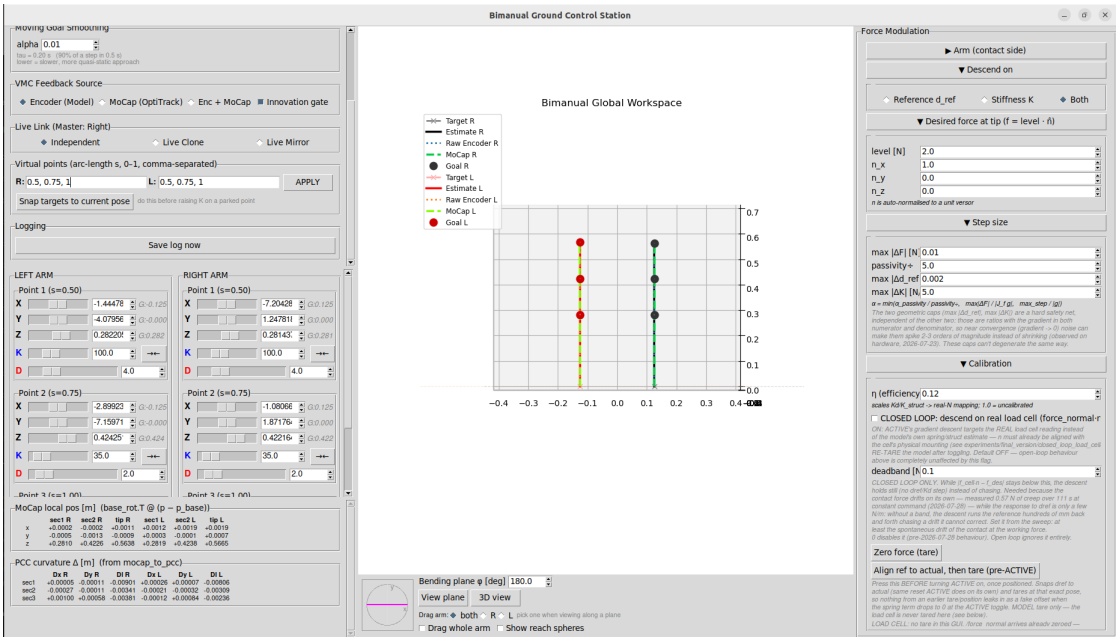


Figure 10.2: The interface with the force-modulation panel enabled. Left: the standard control column and per-element gains. Centre: the digital twin of both arms. Right: the force-modulation strip, here showing the contact arm, the descent mode, the commanded force and normal, and the calibration group.

10.7 Passivity of the fixed-gain controller

With constant user-defined gains, the control law of Section 10.1 is mathematically passive. Defining the storage function as the sum of the potential energies of the virtual elements, $V = \frac{1}{2}\sum_k e_k^T K_k\, e_k$, the active power injected by the static controller is zero. Every force law implemented in Table 10.2 is conservative (e.g., the hyperbolic tangent is the gradient of a separable ln(cosh) potential), and the dampers are strictly dissipative ($\dot{V}_{diss} = -\dot{p}^T D \dot{p} \leq 0$). The static pretension $\boldsymbol{\tau}_0$ derives from a potential linear in $\boldsymbol{q}$,

bounded below on the motor range that the joint stop of Section 10.2.2 enforces, and gravity compensation merely cancels the physical potential. Therefore, energy never actively enters the system through the VMC controller itself.

However, this theoretical guarantee breaks the moment the gains stop being constant. Modulating K or updating the rest position of a virtual spring in order to actively regulate a contact force—which forms the core subject of Chapter 11—physically injects energy into the system. The very mechanism that makes dynamic force regulation possible concurrently makes it capable of destabilizing the interaction. The discrete energy tank formulation developed in the next chapter exists to bound this injected energy, effectively restoring passivity with respect to the storage function while retaining the required adaptability.

Chapter 11

Force modulation and passivity

The controller presented in Chapter 10 acts primarily as a position regulator. A virtual spring pulls its anchor towards a goal; if an obstacle stands in the way, the arm pushes against it with whatever force the residual deflection happens to produce. In this scenario, the contact force is a passive *outcome* of the goal placement, spring stiffness, and structural bending. However, manipulating delicate objects requires the opposite: commanding a specific force, achieved through whatever deflection is necessary.

Estimating that force, inverting the estimate online by gradient descent, and paying for the energy the adaptation injects are the three problems this chapter solves, in that order.[1]

11.1 Task-space force under contact

11.1.1 The problem, and why it is not a sensing problem

The natural way to know a contact force is to measure it, typically placing a sensor at the end-effector. This approach does not apply here. The arms carry no force sensors and lack a dedicated end-effector: contact can occur anywhere along the soft bodies, which is the core principle of whole-body manipulation.

Instead, the system leverages its validated internal model. The structural stiffness K is identified in Chapter 13, the shape $\boldsymbol{\Delta}$ is estimated in real-time, and the virtual forces are precisely commanded by the controller. Under static equilibrium, the forces applied to the environment perfectly balance the virtual and structural forces. Estimating the

[1] The theory regarding the gradient descent on the virtual parameters and the energy-tank argument that bounds it, is the work of L. Vignoli, and is reproduced here because the experimental campaign of Chapter 14 rests on it.

contact force is therefore an exercise in kinematic bookkeeping rather than direct sensing of an environment that is never explicitly modeled.

11.1.2 Three power flows

Consider the arm in contact with the environment at an arc-length s_c, with a unit contact normal n and the associated projector $P = nn^T$ introduced in Section 10.4.2. At any instant, three distinct power flows interact:

- **Virtual elements** inject power through the actuators. Each element applies its force f_d along the velocity $\dot{d}$ of its anchor point, injecting a mechanical power of $\dot{d}^T f_d$.
- **The soft structure** exchanges power as it deforms. The material opposes the motion with the restoring force f_Δ, contributing $\dot{\mathbf{\Delta}}^T f_\Delta$ to the balance.
- **The environment** exchanges power at the contact point. The interaction force f_x acts along the permitted velocity component, expressed as $(P\dot{x})^T f_x$.

The principle of virtual work states that, for any kinematically admissible virtual motion, these powers must perfectly balance:

$$\dot{d}^T f_d \; + \; \dot{\mathbf{\Delta}}^T f_\Delta \; = \; (P\dot{x})^T f_x \tag{11.1}$$

11.1.3 From power balance to force balance

These velocities can be mapped back to the actuator velocities $\dot{q}$ through their respective Jacobians:

$$\dot{d} = J_d\,\dot{q}, \qquad \dot{\mathbf{\Delta}} = J_\Delta\,\dot{q}, \qquad \dot{x} = J_x\,\dot{q}, \tag{11.2}$$

where J_Δ is the constant block-diagonal map and J_d, J_x are the positional Jacobians evaluated at the anchor points and the contact point. Substituting these into Equation (11.1) yields:

$$\dot{q}^T J_d^T f_d + \dot{q}^T J_\Delta^T f_\Delta = \dot{q}^T (PJ_x)^T f_x \tag{11.3}$$

Crucially, the structure receives only the fraction of the virtual torque that survives the transmission, denoted by η. Factoring out the common arbitrary velocity $\dot{q}^T$, the power balance translates into a static force balance in the actuator space:

$$\eta\, J_d^T f_d \; + \; J_\Delta^T f_\Delta \; = \; (PJ_x)^T f_x \tag{11.4}$$

11.1.4 Specializing to static contact

At steady state, the anchors are stationary ($\dot{d} = 0$), meaning the virtual and structural damping terms vanish. Only the elastic contributions remain:

$$f_d = K_d\,(d_{ref} - d) \;\equiv\; K_d\,\delta d, \qquad f_\Delta = -K\,\boldsymbol{\Delta} \tag{11.5}$$

Substituting both into Equation (11.4) defines the static equilibrium:

$$\eta \sum_{i=1}^{N} J_{d,i}^T K_{d,i}\,\delta d_i \;-\; J_\Delta^T K \boldsymbol{\Delta} \;=\; (P J_x)^T f_x \tag{11.6}$$

This equation is physically transparent: of the torque delivered by the virtual springs, part is spent deforming the arm's structure, and the remainder is transferred to the environment.

Note that η scales the virtual term only. The system identification of Chapter 13 was fitted against the delivered torque $\eta\boldsymbol{\tau}$, so the identified K is already the physical stiffness of the structure and the reaction $J_\Delta^T K \boldsymbol{\Delta}$ needs no further downscaling; the virtual forces, by contrast, originate as raw commanded torques and are attenuated before reaching it. The calibration of η against an external sensor is treated in Section 14.1.

11.1.5 Inverting a non-square map

Equation (11.6) determines f_x implicitly through $(PJ_x)^T$. This matrix is not square and is rank-deficient by construction, reflecting a physical reality: a redundant arm can hold the same contact force with infinitely many internal configurations. Actuator torques lying in the null space of $(PJ_x)^T$ simply produce no contact force.

Applying the Moore–Penrose pseudo-inverse returns the least-squares solution, isolating the contact wrench from the internal structural forces:

$$\boxed{f_x = \left((PJ_x)^*\right)^T \left[\, \eta \sum_{i=1}^{N} J_{d,i}^T K_{d,i}\,\delta d_i \;-\; J_\Delta^T K \boldsymbol{\Delta} \,\right]} \tag{11.7}$$

11.2 Adapting the virtual model

11.2.1 Gradient descent on the virtual parameters

Equation (11.7) expresses the interaction as an analytical, differentiable function of the virtual parameters, which allows them to be inverted online by gradient descent rather than by solving an optimization problem in closed form.

Given a scalar objective $L(\theta)$, gradient descent iterates $\theta_{k+1} = \theta_k - \alpha\, \partial L/\partial\theta$, moving each parameter along the direction of steepest decrease. The step size α governs the usual trade-off between convergence speed and overshoot; here it carries a second and more stringent role, since every step also injects energy into a system in contact with an unknown environment, and determining it is the subject of Section 11.4.

The objective is the quadratic loss on the force error, $L = \frac{1}{2}\|f_x - f_{x,ref}\|_2^2$, whose gradient with respect to any virtual parameter θ follows from the chain rule:

$$\frac{\partial L}{\partial \theta} = \left(\frac{\partial f_x}{\partial \theta}\right)^T (f_x - f_{x,ref}) \tag{11.8}$$

Everything that follows is Equation (11.8) specialized to a particular θ. The choice is not a matter of taste: the two parameters produce physically different behaviours.

11.2.2 Reference modulation

The first strategy holds the stiffnesses fixed and moves the virtual rest positions. Since $d_{ref,i}$ enters Equation (11.7) linearly, through $\delta d_i = d_{ref,i} - d_i$ alone, the gradient is immediate and the update reads

$$\Delta d_{ref,i} = -\,K_{d,i}^T\, J_{d,i}\, \eta\, (PJ_x)^*\, (f_x - f_{x,ref}), \qquad d_{ref,i} \leftarrow d_{ref,i} + \alpha\, \Delta d_{ref,i}. \tag{11.9}$$

Because increments accumulate, the reference integrates the force error over successive iterations: the scheme acts as a discrete integral controller and drives the steady-state error to zero.

11.2.3 Stiffness modulation

The dual strategy holds the references fixed and adapts the stiffnesses. Here $K_{d,i}$ multiplies the fixed vector δd_i, so the gradient of a quadratic loss with respect to it is an outer product and the update has rank one:

$$\Delta K_{d,i} = -\left[J_{d,i}\, \eta\, (PJ_x)^*\right] (f_x - f_{x,ref})\, \delta d_i^T, \qquad K_{d,i} \leftarrow K_{d,i} + \alpha\, \Delta K_{d,i}. \tag{11.10}$$

The implementation restricts this update to be isotropic, $\Delta K_{d,i} = \Delta k_i\, I_3$: a full 3×3 update exposes nine free parameters per element driven by a three-dimensional force error, and the descent would spend energy — charged to the tank of Section 11.3 — sculpting anisotropy the task never requested. The updated stiffnesses are clipped at zero: a negative virtual stiffness would describe an element that pushes harder the further it is compressed, which is no longer a passive spring.

11.2.4 Choosing between the strategies

- **Reference modulation** changes the force while leaving the arm's compliance untouched, so an unexpected disturbance is still absorbed by the structure.
- **Stiffness modulation** reaches the force by stiffening the arm instead, spending the compliance the platform exists to provide.
- **Combined modulation** adapts both channels at once. The two can work against each other, since a change in $K_{d,i}$ alters the very sensitivity from which the d_{ref} gradient was computed.

All three were exercised on hardware; Section 14.2 reports what separates them. The implementation provides two further strategies, not exercised in this work: direct stiffness tracking, in which a desired task-space stiffness is requested directly, and position regulation in free space.

11.3 Energy accounting and the tank

Modulating K_d or d_{ref} alters the stored potential energy *even if the arm is perfectly still*. This represents a mathematically active energy injection inside a feedback loop [Califano et al., 2023], and it is exactly what forfeits the passivity established in Section 10.7. To maintain that property, the system must account for what it dissipates and what it injects.

Differentiating the stored energy $E = \frac{1}{2}\,\delta d^T K_d\,\delta d$ with all quantities allowed to vary isolates the two active terms from the ordinary mechanical work: the injected power is

$$P^{OUT} = \tfrac{1}{2}\,\delta d^T \dot{K}_d\,\delta d \;+\; \delta d^T K_d^{sym}\,\dot{d}_{ref}, \qquad K_d^{sym} = \tfrac{1}{2}(K_d + K_d^T), \tag{11.11}$$

the symmetric part appearing because the rank-one update of Equation (11.10) is not symmetric in general and only the symmetric part of a matrix contributes to a quadratic form. Its counterpart is what the arm gives up through the virtual dampers and the identified structural damping,

$$P^{IN} = \sum_{i=1}^{N} \dot{d}_i^T D_{d,i}\,\dot{d}_i \;+\; \dot{\boldsymbol{\Delta}}^T D\,\dot{\boldsymbol{\Delta}} \;\geq\; 0, \tag{11.12}$$

which is unconditionally non-negative but requires motion: a perfectly still arm dissipates nothing.

A scalar reservoir $T_i \geq 0$ (the energy tank) accumulates the dissipated energy and pays for these injections [Califano et al., 2023]. Over a discrete timestep the budget available is $Z_i = T_i + \Delta t\, P^{IN}$, and the tank state updates by debiting the energy spent by the adaptation (ΔE), capped by a maximum capacity T_{max}:

$$T_{i+1} = \min\left(\, Z_i - \Delta E,\; T_{max} \,\right), \qquad T_{i+1} \geq 0 \tag{11.13}$$

Both saturations carry meaning. The upper limit prevents a long quiet phase — during which the arm dissipates without adapting — from accumulating a budget large enough to authorize an arbitrarily violent update much later. The lower limit is what makes the guarantee meaningful at all, since a tank allowed to go negative would describe energy the system never had.

The tank is initialized with a finite charge T_{init} rather than empty. This is a deliberate and necessary choice: with an empty tank the first adaptation could not be funded at all, because dissipation requires the motion that the adaptation itself is meant to produce. It should however be stated plainly what it costs. The guarantee established here is passivity *up to the initial energy*: the controller never creates more energy than T_{init} plus what the system has demonstrably dissipated. Choosing T_{init} is consequently not a formality but the decision of how much budget must carry the descent through the phase in which the arm is barely moving.

11.4 Bounding the step size

11.4.1 The passivity bound

To ensure passivity, the energy demanded by the discrete parameter update step must never exceed the budget available for it:

$$\Delta E \;\leq\; Z_i. \tag{11.14}$$

The energy demanded follows from Equation (11.11) by replacing the continuous rates with the discrete increments actually applied, $\dot{K}_d \Delta t = \alpha\, \Delta K_d$ and $\dot{d}_{ref} \Delta t = \alpha\, \Delta d_{ref}$:

$$\Delta E_K = \tfrac{1}{2}\, \alpha \sum_i \delta d_i^T\, \Delta K_{d,i}\, \delta d_i, \tag{11.15}$$

$$\Delta E_{ref} = \alpha \sum_i \delta d_i^T\, K_{d,i}^{sym}\, \Delta d_{ref,i}. \tag{11.16}$$

Both are *linear* in α, which is what makes Equation (11.14) solvable in closed form rather than by search. Substituting and solving for the largest admissible step gives

$$\alpha_{pass}^{K} = \frac{2 Z_i}{\left| \sum_i \delta d_i^T\, \Delta K_{d,i}\, \delta d_i \right|}, \qquad \alpha_{pass}^{ref} = \frac{Z_i}{\left| \sum_i \delta d_i^T\, K_{d,i}^{sym}\, \Delta d_{ref,i} \right|}. \tag{11.17}$$

The interpretation is direct: the step is as large as the accumulated dissipation can afford, and no larger. When the arm is moving and dissipating freely the adaptation is quick; when it is nearly still it slows down, because there is nothing to spend.

The absolute value in the denominators is a deliberate conservatism. Without it, a step with $\Delta E < 0$ — one that refills the tank — would satisfy Equation (11.14) for any α whatsoever and be granted an unbounded rate. Such a step is energetically safe by construction, but an unbounded one is not physically safe: it is still a discontinuous jump of the virtual model.

11.4.2 Two further bounds

The passivity bound alone proved insufficient on hardware, for two distinct reasons.

Force cap. Passivity constrains energy, not force. A step taken where the arm is stiff can be energetically cheap and still produce a large jump in contact force — and on a delicate object it is the force, not the energy, that does the damage. The step is therefore also limited so that the predicted force change stays below ΔF_{max} per iteration.

Geometric cap. Both bounds above are ratios carrying the gradient in the numerator and the denominator alike. As the descent converges the gradient tends to zero, which is the desired outcome; but the force driving it is the model-based estimate of Equation (11.7), which inherits the noise of the MoCap-based pose. Near convergence the true gradient becomes comparable to that noise, and on an unlucky iteration its norm dips spuriously close to zero, inflating α by orders of magnitude. The bound is thus pathological exactly where the descent is succeeding. The remedy is a cap that does not divide by anything gradient-dependent, limiting the reference step and the stiffness step to fixed geometric thresholds δ_{max} and Δk_{max}; set above ordinary motion and far below the observed spikes, it is inactive during normal descent and binds only on pathological iterations.

11.4.3 The applied step

The step actually applied is the most restrictive of the three, with the passivity bound divided by a safety factor s_f that prevents any single iteration from consuming the entire budget:

$$\boxed{\alpha = \min\left(\frac{\alpha_{pass}}{s_f},\ \alpha_F,\ \alpha_{geom}\right)} \tag{11.18}$$

Each term guards a different failure mode: α_{pass} the stability of the interaction, α_F the integrity of the object, α_{geom} the numerical health of the descent itself. Which one binds is itself diagnostic information, and all three are logged at every iteration. The values used throughout the experiments are collected in Table 11.1.

Parameter	Meaning	Value
Δt	descent loop period	0.05 s (20 Hz)
T_{init}	initial tank charge	2 J
T_{max}	tank capacity	10 J
s_f	passivity-bound divisor	5
ΔF_{max}	max predicted force change per step	0.01 N
δ_{max}	max reference step	2 mm
Δk_{max}	max stiffness change per step	5 N m^{-1}
η	lumped transmission factor	0.12

Table 11.1: Force-modulation parameters. All are adjustable at runtime from the interface; η is the value identified against the load cell in Section 14.1.

11.5 Implementation

The descent runs at 20 Hz, an order of magnitude below the 500 Hz control loop, and the separation is structural rather than computational. The fast loop holds the virtual model against the arm's dynamics; the slow one adapts the model's parameters. Running the adaptation at the control rate would couple it to the very transient the impedance is meant to absorb: the descent would read a force that is still ringing, attribute the ringing to a parameter error, and correct against its own oscillation. The 20 Hz rate is thus a compromise between adaptation responsiveness and the validity of the static equilibrium assumption of Equation (11.6) on which every expression in this chapter rests, and the bounds on α absorb the residual transients between iterations.

Each iteration evaluates the gradient, computes the three bounds, applies the step, credits the tank with $\Delta t\, P^{IN}$ and debits it by ΔE. The updated references or stiffnesses are then published to the controller, which consumes them as ordinary goals and gains: the fast loop contains no adaptation logic whatsoever, and would run identically if the descent were switched off.

The force driving the descent can be either the model estimate (open loop) or an external measurement from the load cell (closed loop). In open loop, the descent minimizes the model's own predicted force: the objective has an exact solution, which the descent reaches and holds as long as the bounds of Section 11.4 grant it a non-zero step, and the accuracy of the result is entirely the accuracy of the model. In closed loop, the residual is computed against the load cell, closing the modeling gap but introducing the physical limitations discussed next. Both modes, together with the tank state, the three bounds and the resulting α, are exposed in the interface and logged at every iteration.

11.6 What the guarantee covers

The bounds of Equation (11.18) buy stability at a direct cost in convergence speed, and one case deserves stating because it is not incidental: since the tank recharges only through motion, the regime in which it runs dry fastest is a slow, quasi-static push — exactly the regime delicate manipulation calls for.

Two final clarifications bound what the guarantee covers. Passivity here is a statement about the *controller*: the adaptation creates no energy beyond the initial charge and what the system has demonstrably dissipated. That bounds the interaction with any passive environment without requiring a model of it, which is the property that makes contact with an unknown object safe. It says nothing, however, about whether the commanded force is the right one for the object at hand — that is the operator's judgement — nor about the accuracy with which Equation (11.7) estimates the force actually applied. The latter is a question of model fidelity rather than of passivity, and it is settled experimentally, against the load cell, in Chapter 14.

Part IV

Experimental Validation

Chapter 12

Experimental Overview

The preceding parts developed a model of the platform and a controller built on top of it. Both rest on quantities that were assumed rather than measured: that the identified stiffness describes the real structure, that the transmission factor η is a constant, that the force expression of Chapter 11 reports what the arm is actually applying, and that the arm reaches where the kinematics says it does. This part measures each of them on hardware.

The experiments are ordered so that every one of them can be read without trusting the result of a later one. Each answers a question that the next takes as given.

- **Is the identified structure right?** Chapter 13 identifies K and D from a step-and-settle protocol, on the encoder and the MoCap datasets independently, and settles which of the two the controller loads where.
- **Does the model measure force correctly?** Section 14.1 calibrates η against a load cell in static contact, which is the first time an external force sensor enters the work.
- **Can it command a force?** Section 14.2 runs the gradient descent of Chapter 11 in open loop: the controller drives its own estimate to a target force and the load cell, never fed back, records what actually arrives. Section 14.3 then closes the loop on the cell.
- **Can it position?** Chapter 15 measures the workspace directly rather than deriving it, then characterises the step response and the tracking of closed planar trajectories.

12.1 Common setup

For contact force experiments, the measurements were taken on the **right** arm with a single virtual element anchored at the tip ($s = 1.0$). Contact forces are measured by

two single-axis load cells mounted on orthogonal linear guides, which resolve the normal and the tangential (shear) component of the contact force. Each is read through an HX711 amplifier by an Arduino, which publishes both components and their magnitude in newtons, zeroed at boot. Unless stated otherwise, the force reported against the model is the normal component, the one the estimator of Chapter 11 predicts. The pose is measured by the OptiTrack system described in Section 9.1.2. The controller logs every tick at 500 Hz — state, gains, goals, torques, tank state and the three step-size bounds — and each run is archived immediately after recording into its own timestamped folder. The analysis is performed post-hoc from those archives rather than from live inspection, so that every figure in this part can be regenerated from the saved data.

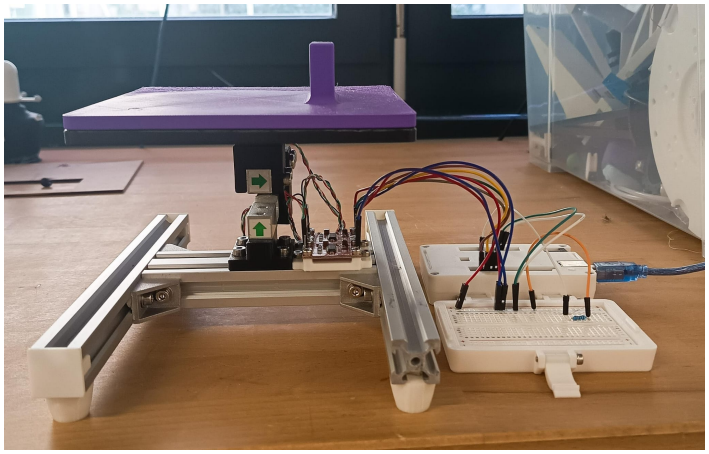


Figure 12.1: The load-cell rig used throughout this part, shown without the arm: two single-axis cells on orthogonal guides resolve the normal and the tangential component of the contact force. During the experiments the tip of the right arm presses on the plate, and the rig serves as ground truth for every force reported in the following chapters.

Chapter 13

System Identification

The soft arm's material inherently resists motion. Each section can be modeled as a continuum system characterized by stiffness (K) and damping (D) matrices. Identifying these matrices is a prerequisite for the prediction model. The adopted procedure is a constrained linear least-squares regression.

13.1 Experimental Protocol

The dataset was gathered by applying a step-and-settle torque sequence. To maintain tendon tension, a constant baseline torque of 0.03 N m was applied to all 18 motors throughout the experiment.

The excitation routine was structured into two main categories:

- **Bending phase:** Step torques were applied to one motor at a time to identify the structural bending stiffness.
- **Compression phase:** Three motors of a single section were pulled simultaneously, compressing the entire section to identify the axial stiffness K_l and damping D_l.

The commanded torque amplitudes are summarized in Table 13.1.

Excitation type	**Section**	**Training torques** [N·m]	**Testing torques** [N·m]
Bending (1 motor)	1st	0.025, 0.040, 0.060	0.035, 0.050
	2nd & 3rd	0.045, 0.060, 0.090	0.045, 0.100
Compression (3 motors)	1st	0.010, 0.015	0.012
	2nd & 3rd	0.015, 0.020	0.018

Table 13.1: Commanded step torque amplitudes applied above the 0.03 N m baseline.

Each cycle consisted of a 4 s resting period, followed by a 10 s step torque maintained until the arm fully settled. Finally, a joint-space PID controller, properly tuned, safely returned the arm to the home configuration. These recovery phases were intentionally not recorded.

13.2 Dynamic Model and Least-Squares Regression

Given the commanded torques, the transmission factor η, the mass matrix $M(\boldsymbol{\Delta})$, and the state $\boldsymbol{\Delta}$, the K and D matrices must be identified through a least-squares regression:

$$J_{\Delta}^{T}\left(M(\boldsymbol{\Delta})\ddot{\boldsymbol{\Delta}} + D\dot{\boldsymbol{\Delta}} + K\,\delta\boldsymbol{\Delta}\right) = \eta\,\delta\boldsymbol{\tau} \tag{13.1}$$

Since the mass matrix $M(\boldsymbol{\Delta})$ is completely known, the small inertial contribution is evaluated sample by sample and moved to the right-hand side as a known vector $b = \eta\,\delta\boldsymbol{\tau} - \boldsymbol{\tau}_{inertia}$. Because the motors of a specific section only dictate the curvature of that section, the Jacobian contains non-zero rows only for the corresponding motors, causing the matrices to assume a block-diagonal structure.

Tendon pretension and the initial resting state would bias the system identification. Therefore, the model was evaluated on variations from the resting pose ($\delta\boldsymbol{\Delta} = \boldsymbol{\Delta} - \boldsymbol{\Delta}_{base}$ and $\delta\boldsymbol{\tau} = \boldsymbol{\tau} - \boldsymbol{\tau}_{base}$).

Placing η on the right-hand side of Equation (13.1) is what makes the identified K the *physical* stiffness of the structure rather than a command-referred one: the regression is fitted against the torque actually delivered, not against the torque requested.

The problem is solved using a bounded linear least-squares algorithm (`lsq_linear`), constrained to non-negative solutions ($K, D \geq 0$). The temporal first half of the active samples was used for training, while validation was performed by running a forward simulation over the entire active window.

Two parallel identifications were executed: one using the encoders' forward kinematics, and one using the raw MoCap positions.

13.3 Results and Model Selection

The identified values for the right arm are reported in Table 13.2. The stiffnesses exhibit two physical consistencies: the axial stiffness K_l is approximately double the bending stiffness across all sections, and the shorter distal sections (Sections 2 and 3) correctly exhibit a higher structural stiffness than the longer proximal base section.

	Encoder			MoCap		
Section	x	y	l	x	y	l
Stiffness K [N/m]						
1st	234	199	410	142	131	335
2nd	392	475	832	296	323	589
3rd	445	518	938	322	334	552
Damping D [N·s/m]						
1st	125	105	205	34	26	27
2nd	139	148	284	45	17	15
3rd	135	141	280	17	12	15

Table 13.2: Identified stiffness and damping parameters for the right arm, comparing encoder and MoCap datasets.

Figures 13.1 and 13.2 illustrate the tracking performance of the identified model. During bending, the simulated K and D parameters accurately predict the measured PCC curvature strain. However, during high-value compression testing, the fit diverges slightly. This discrepancy likely arises because deep compression pushes the TPU structure into its non-linear deformation regime, making it difficult to estimate a unique linear K and D value.

The two datasets disagree far more on damping than on stiffness: the stiffnesses differ by 25 to 40 %, the damping by factors of three to nearly twenty. This is the sensing asymmetry of Section 9.2.2 seen from the identification side. Damping is identified from velocity, which the MoCap obtains by numerical differentiation, whereas its position is the more trustworthy of the two measurements.

Which identification is used where. The controller does not use the same identification everywhere, and the split is deliberate. The force estimator of Chapter 11 loads the **MoCap** identification. Damping is not what decides it: the contact force is evaluated in quasi-static conditions, where the damping term is absent by construction, so the disagreement of Table 13.2 on D is irrelevant there. What decides it is K, which enters the force estimate directly through the structural reaction $K\boldsymbol{\Delta}$. The encoders report the length the tendons were *commanded* to take, and tendon slack means the shape they imply is not the shape the arm has, whereas the MoCap observes that shape directly.

The state estimator of Chapter 9 loads the **encoder** identification instead. There the observation matrix is the identity and the full state is measured, so the physical model acts as a noise filter rather than as an observer of hidden variables, and an error in K has a second-order effect on the fused estimate.

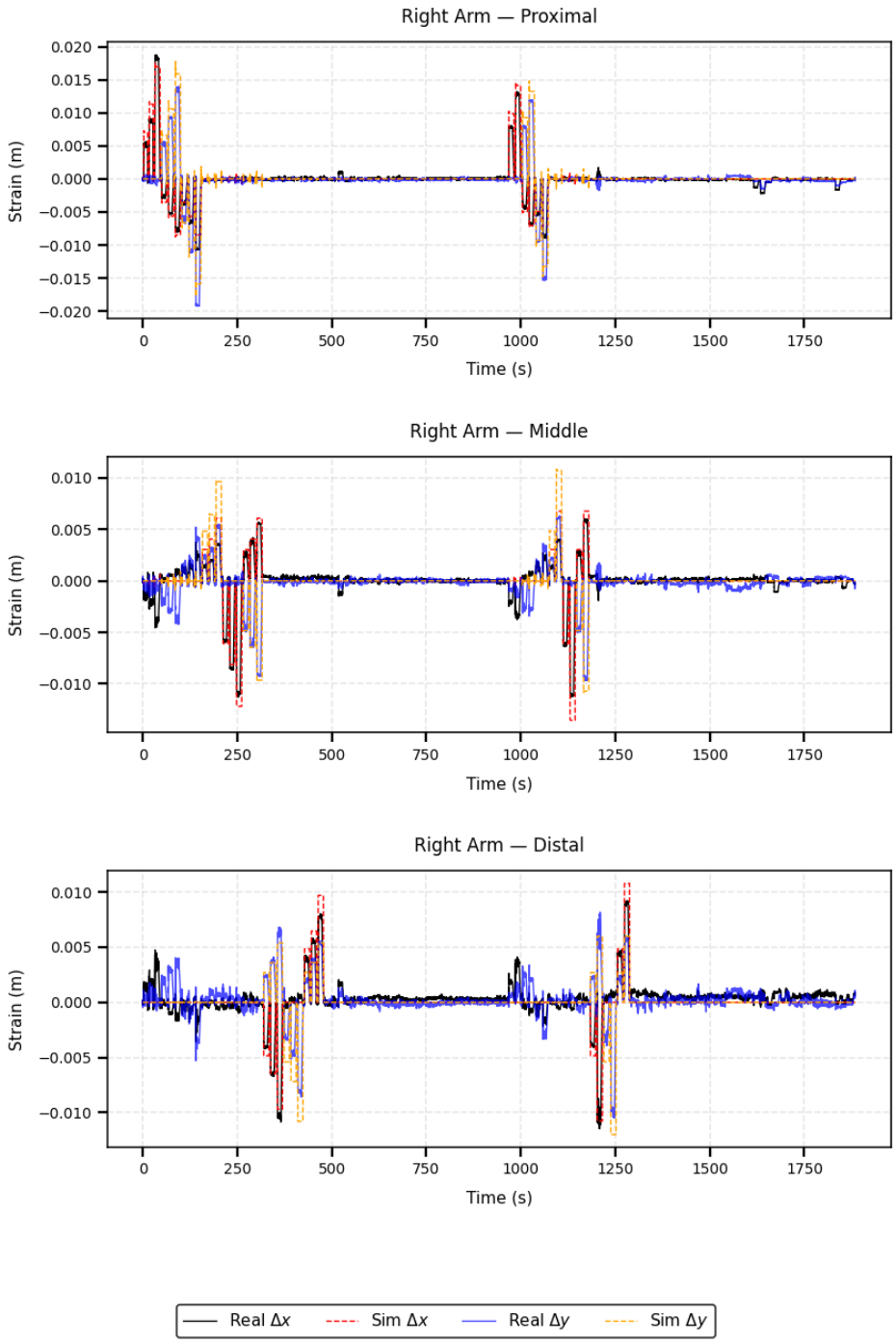


Figure 13.1: Measured and simulated PCC curvature strain during the bending excitation, right arm. Section 1 is the proximal one.

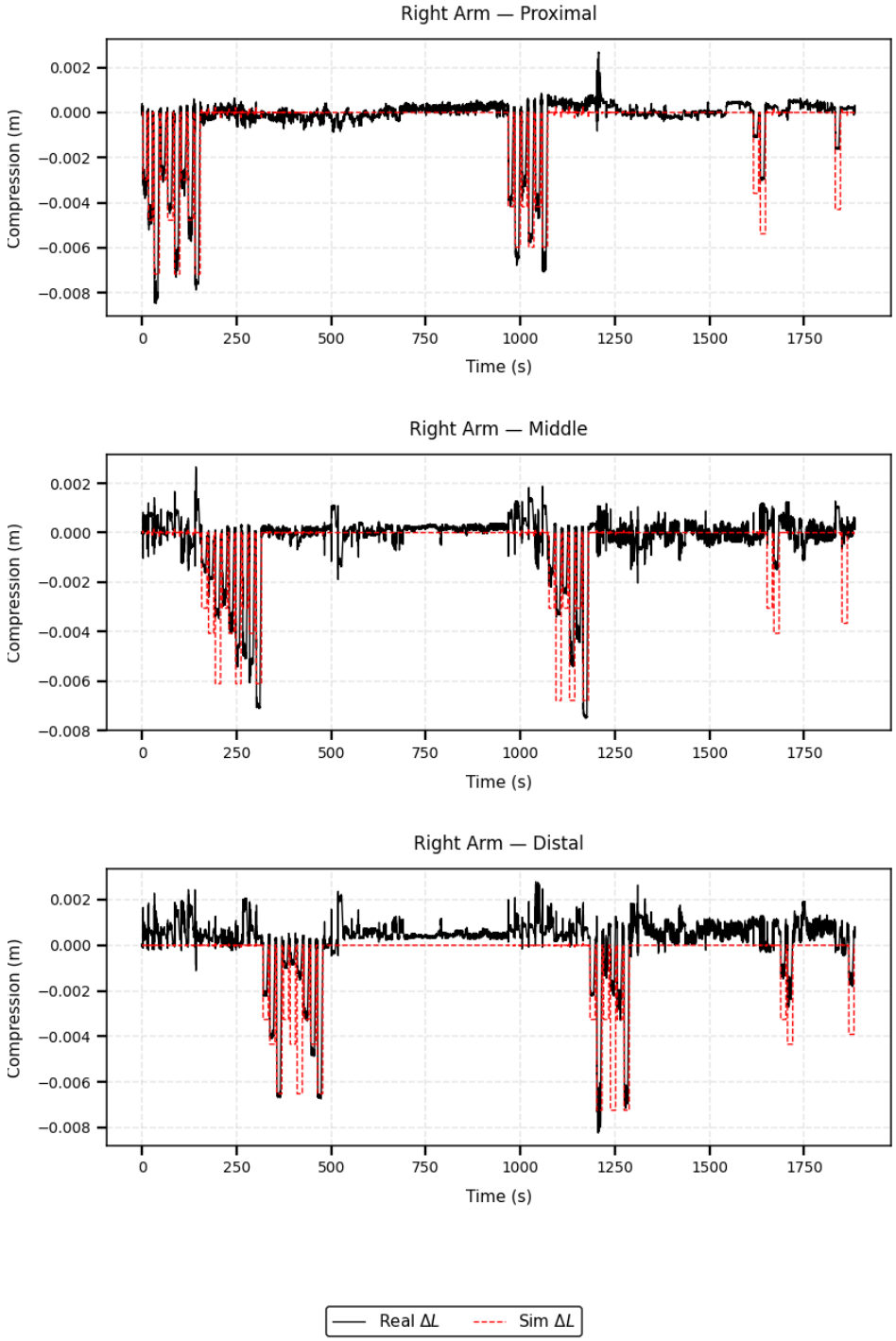


Figure 13.2: Measured and simulated axial compression δL during the compression excitation, right arm.

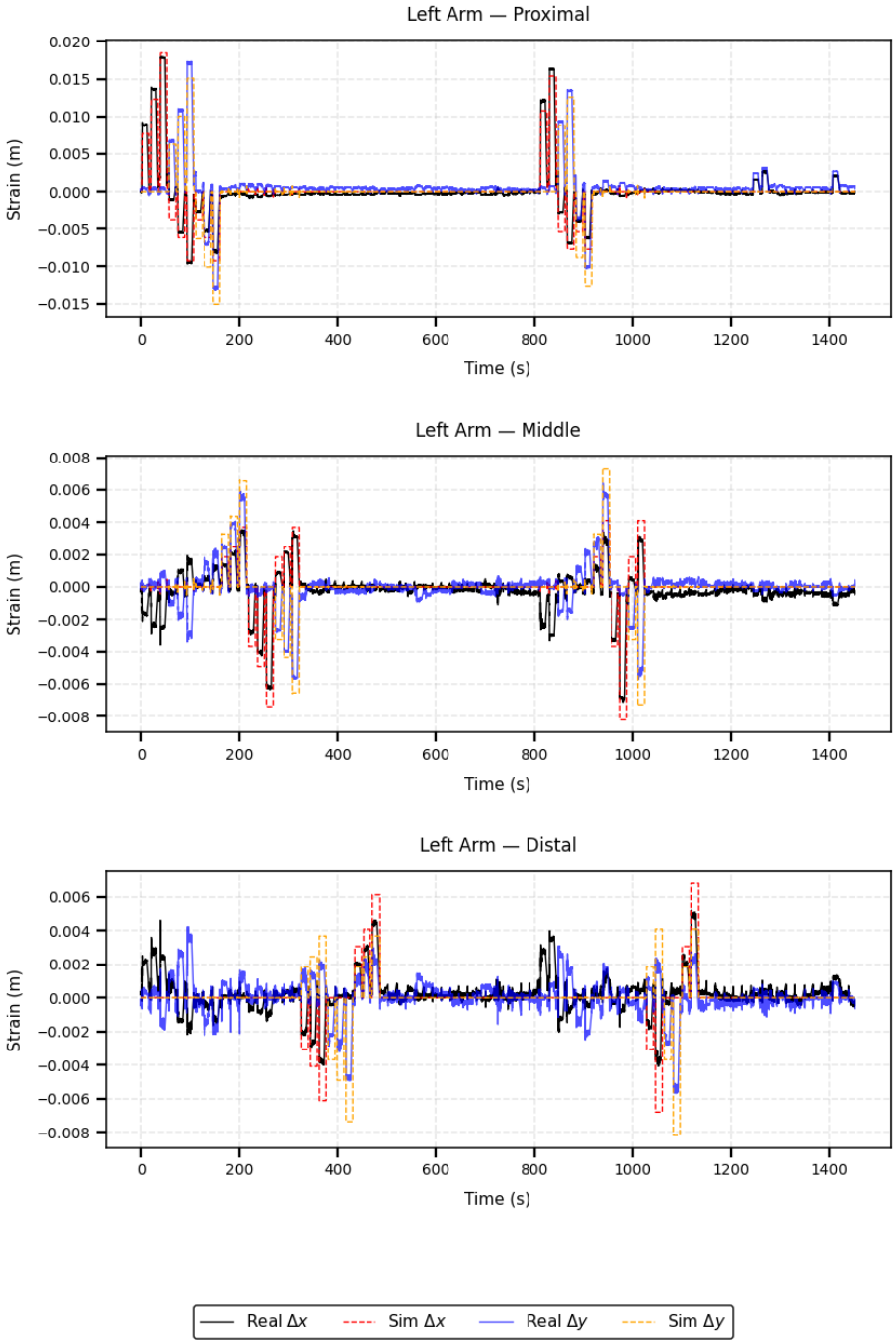


Figure 13.3: Measured and simulated PCC curvature strain during the bending excitation, left arm.

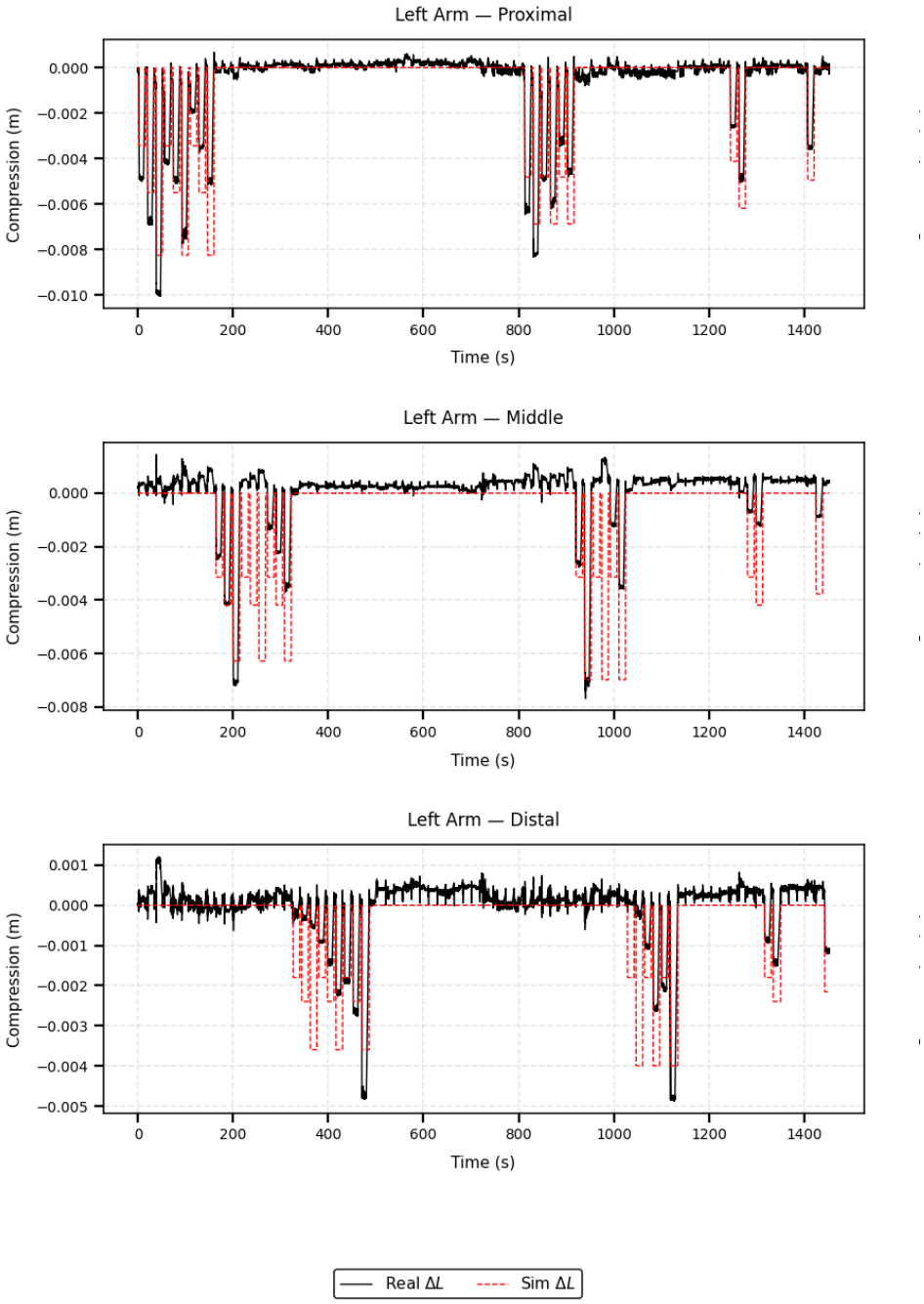


Figure 13.4: Measured and simulated axial compression δL during the compression excitation, left arm.

The structural damping entering the energy tank of Section 11.3 is taken from the MoCap identification, which is the one estimating the *lower* dissipation — so the passivity bound errs on the conservative side, authorizing smaller adaptation steps than a larger damping estimate would.

The two arms. The left arm underwent the identical protocol; Table 13.3 compares the two on the MoCap identification. The arms are measurably different, but not in a uniform way: the proximal section is slightly softer on the left, while the two distal sections are markedly stiffer, by some 50 % in bending and, for the third section, by nearly a factor of two axially. The asymmetry is therefore real and exploitable in principle — one arm can support a load while the other manipulates — but it is not a single global stiffness offset, and it cannot be attributed to the choice of TPU grade alone. Print quality, assembly tension and tendon routing all differ between the two builds.

	Right arm			**Left arm**		
Section	x	y	l	x	y	l
Stiffness K [N/m]						
1st	142	131	335	130	138	291
2nd	296	323	589	487	476	573
3rd	322	334	552	509	489	1001
Damping D [N·s/m]						
1st	34	26	27	43	47	56
2nd	45	17	15	45	27	74
3rd	17	12	15	16	7	51

Table 13.3: The two arms compared on the MoCap identification.

Free-space equilibrium and unmodelled friction. One qualitative observation is worth recording. In free space the arm is in static equilibrium between the virtual springs and its own elasticity, so the force estimator of Chapter 11 must read zero; with the MoCap identification loaded it was instead observed to reach approximately 0.2 N, and to change sign with the direction of travel. A term that reverses with direction is not elasticity but friction, and the model of Chapter 11 contains no such term. Its magnitude is comparable to the 0.2 N accumulated during the free-space approach of Section 14.2, which suggests that the two are the same effect and that the accuracy of the force estimate is bounded by unmodelled friction rather than by the identification of K.

This was observed during operation and not recorded as a dedicated experiment, so it is reported here as an observation rather than as a measurement. Quantifying it properly is straightforward and requires no force sensor: a slow sweep out to full reach and back, in free space.

Chapter 14

Force control

Chapter 13 established that the structural model is accurate within the limits of Section 13.3. This chapter takes the force estimator built on that model to the load cell and asks three successive questions: whether it *measures* contact force correctly, whether it can *command* one, and what changes when the sensor is placed inside the loop.

The distinction between the second and third is the organising idea of the chapter. In open loop, the descent drives the model's own estimate to the target and the load cell is a silent witness; the error it records is the model's error. In closed loop, the residual is computed directly from the cell itself, assessing closed-loop tracking performance.

The force experiments described in this chapter were conducted through the GUI using the `force_mod` mode. The interface and its control panels are described in Section 10.6.

14.1 η calibration

14.1.1 What is being fitted

The transmission factor η scales the virtual channel of the equilibrium Equation (11.6). It is the only quantity in the force estimator that cannot be obtained from geometry or from the structural identification, so it must be measured against a known external force.

The relation to be fitted follows directly from that equilibrium. Of the force the virtual springs command, only the fraction η reaches the structure; once there, it is spent in two places and two only—deforming the arm itself, and pushing on whatever the arm is touching. The two must therefore add up to what arrived:

$$\underbrace{f_{cell}}_{\text{measured}} + \underbrace{f_{struct}}_{\text{model}} = \eta \cdot \underbrace{f_{spring}}_{\text{commanded}} . \tag{14.1}$$

Every term is known: the load cell reads the first, the structural model of Chapter 13 supplies the second, and the controller generates the third. Plotting the left-hand side against f_{spring} gives a line through the origin whose slope is η, which constitutes the entire measurement.

14.1.2 Protocol

The measurement must be taken *in contact and at rest*, since Equation (14.1) is a static balance. Because the fit is constrained through the origin, both sides of Equation (14.1) must be zero at the same physical state. Establishing that common zero requires two actions:

1. The arm is brought into light contact with the load cell and the cell is **tared**, so that it reads zero in that state.
2. The rest position is **aligned to the estimated pose**, so that $\delta d = 0$ and hence $f_{spring} = 0$ in that same state.

Taring without aligning leaves a virtual force already standing while the cell reads zero; aligning without taring does the reverse. Either action alone displaces the origin and biases the fitted slope. The virtual stiffness is then held fixed at $K_d = 60\,\mathrm{N\,m^{-1}}$ and the rest position is stepped outward in increments of 10 mm, with each step held until the contact force has settled. The viscoelastic compliance of the TPU creates an initial force spike followed by a 40–90 second relaxation transient, until a nearly stable steady-state value is reached. To ensure objectivity regarding when to advance to the next goal, a programmatic procedure was adopted: the settled plateaus are detected post-hoc from the recorded log by requiring d_{ref} and K_d to be constant and the force variance to be strictly low over a sufficient window, rather than being marked manually during the run.

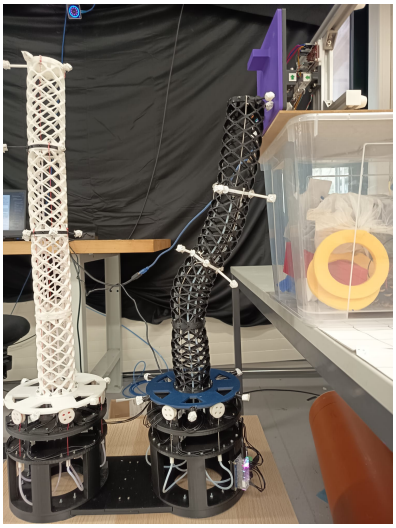


Figure 14.1: Experimental setup

The procedure was repeated across three genuinely different resting poses to verify that the η parameter does not depend on the arm's configuration (i.e., its bending state).

14.1.3 Results

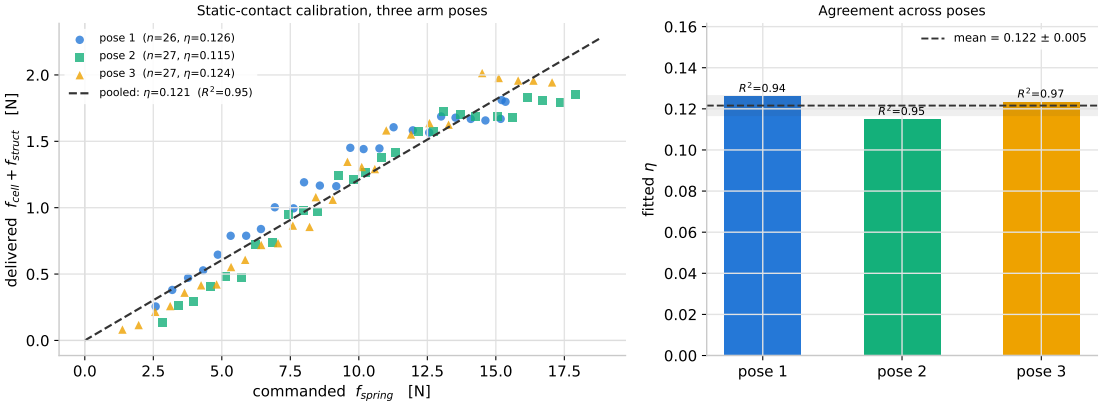


Figure 14.2: Left: the regression of Equation (14.1). Each point is one settled plateau; the dashed line is the fit pooled over all three poses. Right: the factor fitted independently for each pose, displaying the mean and its spread.

The three poses return $\eta = 0.126$, 0.115, and 0.124, each with R^2 between 0.94 and 0.97, yielding:

$$\eta = 0.122 \pm 0.005, \tag{14.2}$$

a spread of about four percent. Pooling all eighty settled plateaus into a single regression gives $\eta = 0.121$ with $R^2 = 0.945$. The value adopted throughout this work is $\eta = 0.12$.

The agreement across the three poses is the substantive result, and it is exactly what the protocol was designed to produce. Using three genuinely different configurations of the arm validates the core assumption the force estimator relies on: a single scalar can accurately describe the transmission efficiency regardless of how the arm is shaped.

14.2 Open-loop force tracking

With η calibrated, the model is a validated force meter. This section tests it as a force *actuator*: a target force is commanded, the gradient descent of Section 11.2 moves d_{ref} until the model's own estimate reaches it, and the load cell — never fed back — records what actually arrives.

14.2.1 Protocol

The setup is the one used for the η calibration, with one difference: the load cell is not re-tared before the run, but zeroed once at boot. The arm consequently accumulates roughly 0.2 N of estimated force during the free-space approach — the same residual identified as unmodelled friction in Section 13.3. The run is left this way deliberately, so that it reproduces a real contact rather than one preceded by a dedicated calibration of the whole system.

Once contact is established and a fixed settling time has elapsed, the descent is switched on. Its parameters — target force, initial tank energy, and the bounds on the step size — are set from the force panel of the interface shown in Figure 10.2. Each target is held until the force reaches steady state, then the next level is commanded.

14.2.2 Results

The result is good, and it is worth stating plainly what it means: the arm applies a commanded contact force using nothing but a model of itself — no force sensor of any kind in the control path.

All three adaptation strategies of Section 11.2 were exercised in open loop, from different initial arm poses. In every run the descent is switched on once per target and switched off in between, while the tip is repositioned by hand; the shaded bands in Figures 14.3 to 14.5 mark the active windows, and only those are part of the measurement.

One protocol difference is worth stating. When the stiffness is the adapted channel, the reference is deliberately *not* aligned to the measured pose but left with a standing deflection: the virtual force is $K_d\,\delta d$, so with $\delta d = 0$ the stiffness channel has no lever and any update to K_d produces exactly nothing.

In several runs, for forces above 2 N, non-ideal behavior caused the force on the load cell to decrease significantly. The interpretation of this behaviour was done by observation of the system. As the commanded force grows, the tangential load at the contact eventually exceeds what friction between the soft tip and the cell face can hold, and the tip begins to slide down the face instead of pressing into it. Part of the commanded push is then converted into tangential load, and the rig measures it directly: the ratio between the tangential and the normal component rises from below 0.1 at 0.5 N to 0.46 at 2.5 N, the same value in both runs of Figures 14.3 and 14.5, which is where a Coulomb contact starts to slide. The normal reading stalls while the controller pushes harder.

Open-loop force control is therefore usable on this platform below about 2.0 N and unreliable above it — not because the controller misbehaves, but because it is blind to a disturbance that a sensor would catch immediately. That is the direct motivation for the next section.

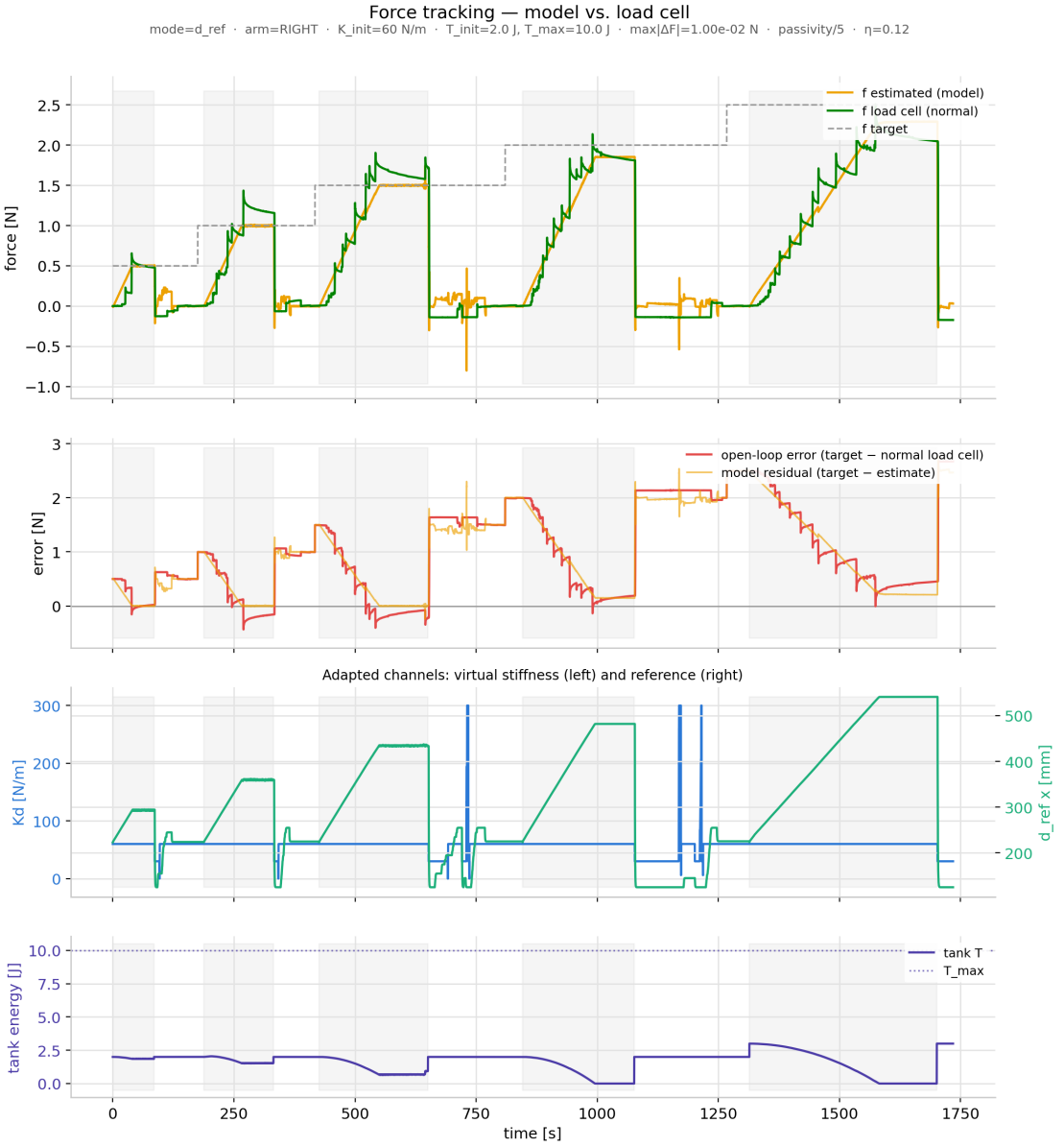


Figure 14.3: Open-loop force tracking, reference modulation. From top: the model estimate, the load-cell reading and the commanded target; the error against the cell, which the controller never sees, and the model residual, which the descent drives to zero whenever the step-size bounds of Section 11.4 grant it a non-zero step; the adapted channels, with K_d held at $60\,\mathrm{N\,m^{-1}}$; the energy tank. The load-cell trace is the normal component. Shaded bands mark the intervals in which the descent is active; between them it is switched off and the tip is brought back to the initial contact by hand. The same conventions hold for Figures 14.4 and 14.5.

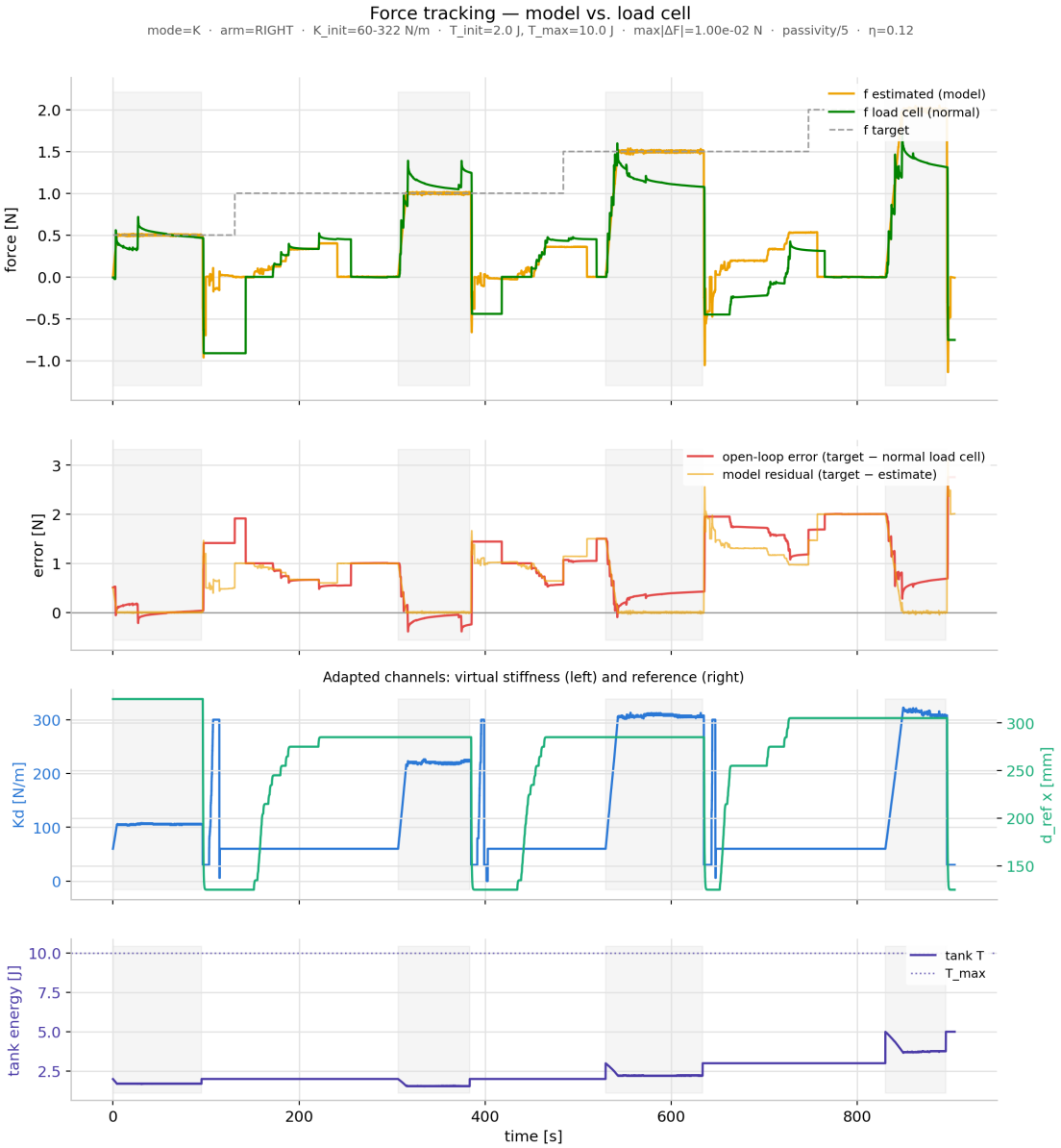


Figure 14.4: Stiffness modulation. The run starts from a standing deflection, so that the stiffness channel has a lever, and K_d alone carries the adaptation: K_d climbs from 60 to 322 $\mathrm{N\,m^{-1}}$. Shaded bands as in Figure 14.3.

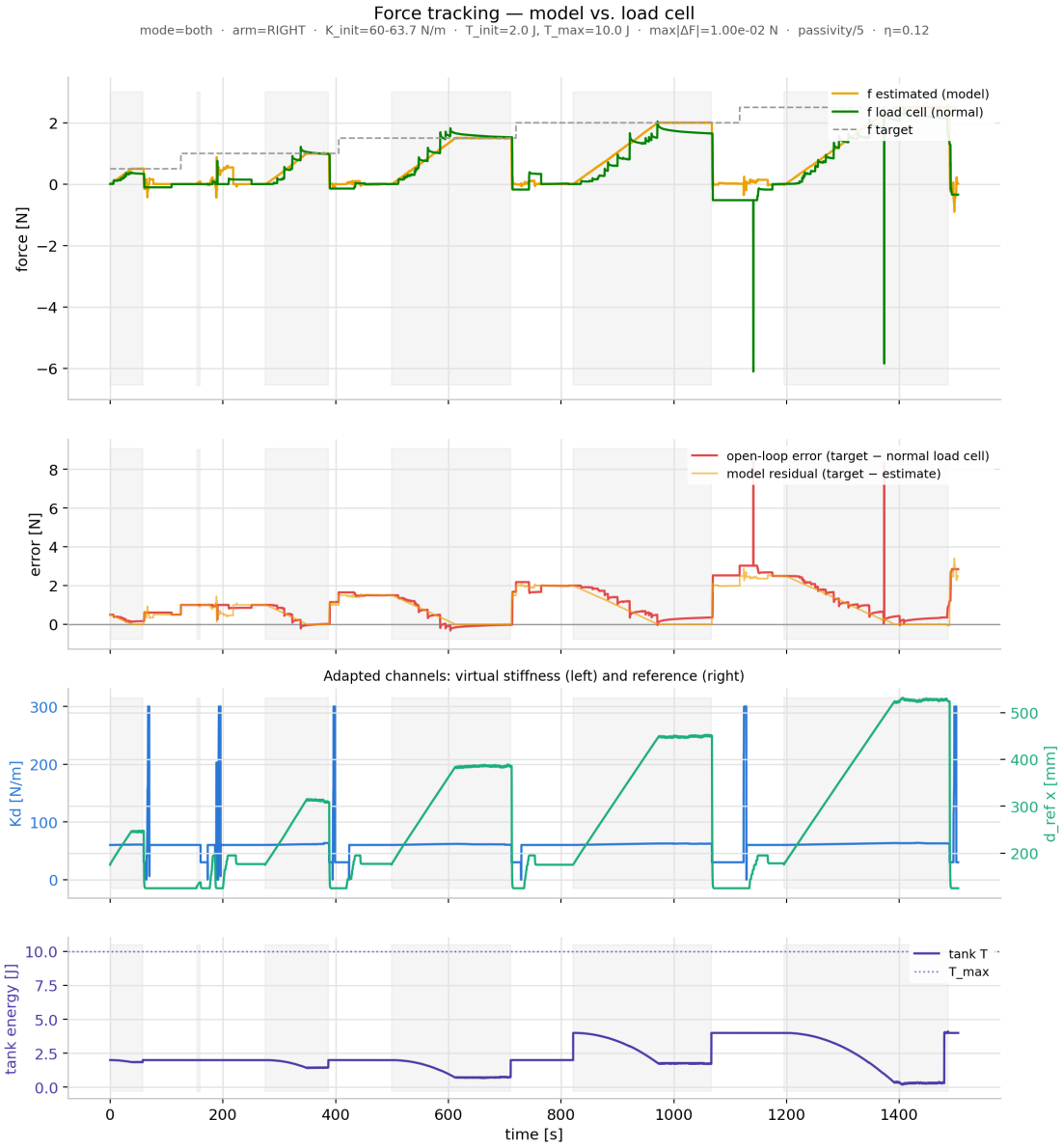


Figure 14.5: Both channels free to adapt. The descent leaves the stiffness untouched and does the work with the reference, reaching 0.5 N, 1.0 N, 1.5 N, 2.0 N and 2.5 N in turn. Shaded bands as in Figure 14.3.

14.3 Closed-loop force tracking

14.3.1 What changes

Only one thing changes: where the error comes from. The descent still computes its *direction* from the model, since only the model knows how a displacement of d_{ref} propagates to the contact point. But the residual it minimises is now $f_{target} - f_{cell}$, measured, instead of $f_{target} - f_{estimate}$, believed.

14.3.2 Results



Figure 14.6: Closed-loop tracking of three force levels. Top: load cell against target, with the detected slip event shaded. Middle: the reference the descent is moving. Bottom: the virtual stiffness, held constant — this mode adapts the reference only.

The loop tracks 1.0, 1.5 and 2.0 N with a steady-state error below 0.1 N at every level (Table 14.1), where the open-loop runs of Section 14.2 stayed within 0.10 N below 2 N and drifted to 0.38 N above it.

f_{target} [N]	steady-state error [N]	RMS, last 30 s [N]	Δd_{ref} [mm]
1.0	−0.029	0.025	15
1.5	−0.050	0.042	156
2.0	−0.034	0.023	28

Table 14.1: Closed-loop force tracking (Figure 14.6), normal component. The error is the mean over the last two seconds of each level; the small negative bias is the tangential share of the magnitude the loop regulates. The reference travel at 1.5 N is the recovery from the slip event.

	mean $\lvert e \rvert$ [N]	max $\lvert e \rvert$ [N]	RMS [N]
open loop, $f_{\text{target}} \leq 1.5$ N	0.103	0.233	0.134
open loop, $f_{\text{target}} \geq 2.0$ N	0.380	0.532	0.401
closed loop, 1.0–2.0 N	0.038	0.050	0.039

Table 14.2: Steady-state error against the load cell, open loop (the two runs of Figures 14.3 and 14.5) and closed loop (Figure 14.6). Normal component. The open-loop error nearly quadruples above 2 N.

Two observations matter more than the tracking itself.

- **The event in the shaded region.** The arm lost contact with the cell entirely, and a marker on its body came into contact with it instead, producing a collapse below zero and a recovery. The gradient descent, still active, drove the new contact until the commanded force was met again. Nothing of this would have happened in open loop, because nothing of it would have been noticed — the model would have reported a steady 1.5 N throughout. What the loop demonstrates here is not accuracy but robustness: it regulates whatever contact actually exists, including one the model has no representation of.
- 2.0 N **is reached and held at all**, since that is the level at which the open-loop runs of the previous section became unreliable. The mechanical limit itself is not removed — the slips still occur — but the loop now observes them and corrects for them.

14.3.3 Why the descent needs a deadband

Closing the loop introduces a failure mode of its own. The contact force creeps by itself at a constant command, while the sensitivity of that force to the reference is only a few newtons per metre, so chasing the creep by moving d_{ref} would demand

hundreds of millimetres of travel. A first run recorded without protection did exactly that: the reference accumulated 518 mm without ever converging. The step-size bounds of Section 11.4 all scale as the inverse of the gradient norm, so the step actually applied has a fixed magnitude regardless of the force error: chasing a target that keeps moving, the descent degenerates into a fixed-step relay and never settles. The deadband exposed in the interface holds the descent still inside a band around the target, which converts an unattainable objective into a bounded steady-state error.

Chapter 15

Position control

The force experiments held the arm against an obstacle. This chapter characterises it in free motion: how it responds to a commanded displacement, and how well it follows a continuous path.

Both required measuring the workspace first. Indeed, the first tracking runs showed steady-state errors of several centimetres that turned out not to be a control problem at all: the arm was simply being sent where it could not physically go. Lateral reach and tip height are strictly coupled on a continuum arm. Since the whole chapter depends on knowing where the arm can actually reach, the envelope is measured rather than derived.

15.1 The measured workspace

The envelope of a continuum arm is a complex curved surface, not a cylinder. The scan was therefore teleoperated: both arms were driven around via the interface while the controller logged spatial data at 500 Hz, holding each pose worth recording for about two seconds. To ensure accuracy, only stationary samples were kept.

Figure 15.1 shows a top view and a lateral view of the teleoperated workspace test, where the arms were driven across their positive and negative x and y boundaries. The measurements confirm a remarkably large operational volume, demonstrating that the arm can even curl inward to safely reach positions well below its own base.

From this plot, a first physical discrepancy was observed: the left arm extends slightly further than the right arm. The printed geometry is identical and only the TPU grade differs, so the difference is not one of design; it follows the structural asymmetry measured in Section 13.3, where the left arm's distal sections were found to be markedly stiffer and therefore less compressible under the same load.

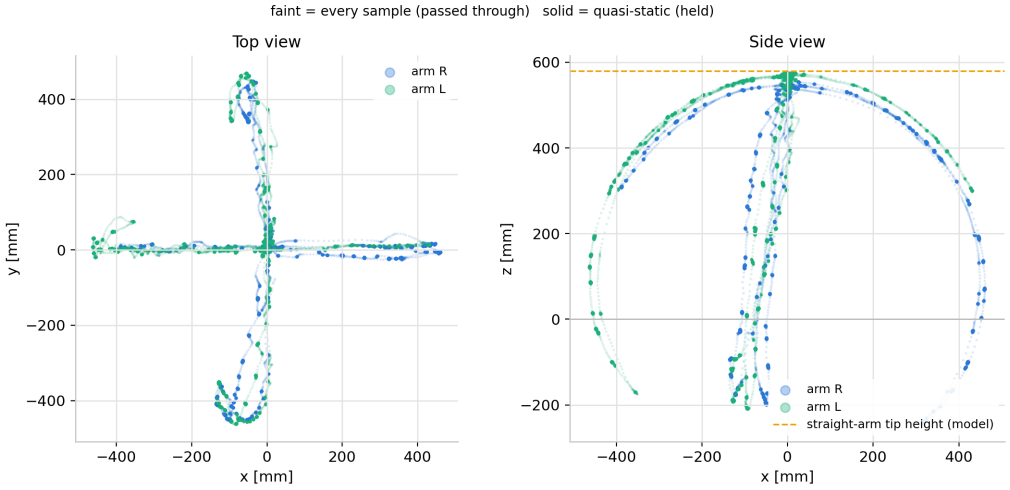


Figure 15.1: Measured workspace of both arms. Left: top view. Right: side view, with the dashed line marking the tip height the kinematic model predicts for the straight arm. Faint points are every logged sample; solid points are the quasi-static ones retained by the speed filter.

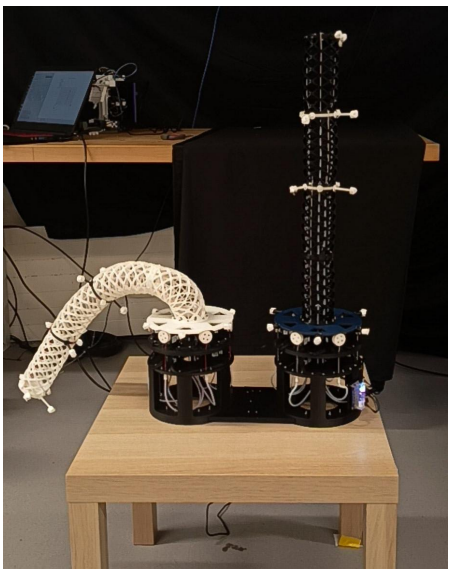


Figure 15.2: Workspace test - bending of the left arm.

The practical consequence is the one that motivated the measurement: the target heights for the experiments below are taken strictly from this empirical envelope.

15.2 Step response

Steps are commanded by a script that drives a fixed sequence of displacements along x and y and holds each until the arm settles. The same sequence of 24 steps was repeated under seven combinations of virtual stiffness and damping.

Overshoot and rise time are measured against the level the arm actually settles at, not the commanded one; the gap between the two is reported separately as the steady-state error.

K [N/m]	D [N·s/m]	steady-state error [mm]	overshoot [%]	rise time [s]
30	2.5	26.7	4.2	0.55
50	3.5	11.3	12.1	0.38
70	3.0	11.6	26.8	0.27
70	4.0	8.9	20.8	0.29
70	6.0	10.1	13.0	0.34
70	8.0	12.7	6.5	0.46
100	5.0	8.1	27.4	0.27

Table 15.1: Step response averaged over 24 steps per gain setting. The steady-state error is taken on the most demanding targets, at 200 and 300 mm from the rest pose.

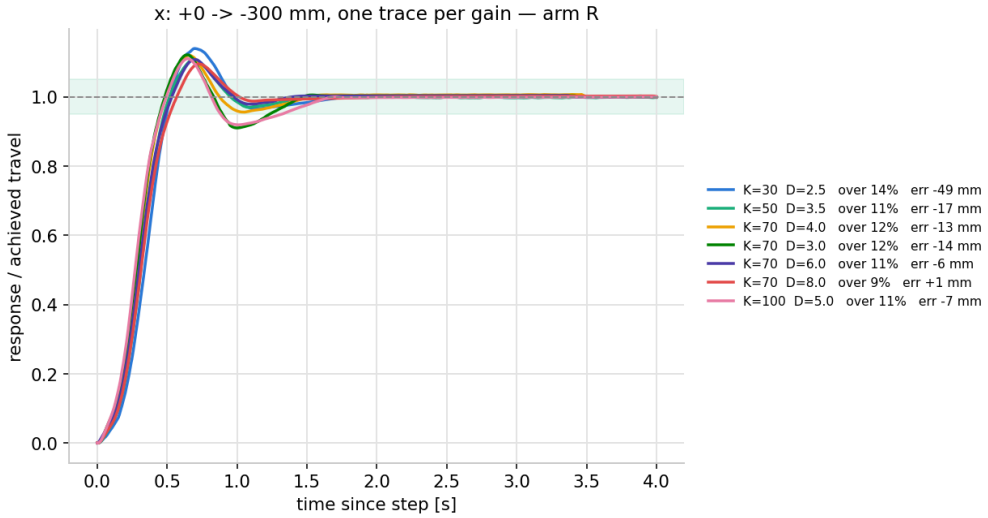


Figure 15.3: Step response of Table 15.1

The middle block of Table 15.1 holds K fixed and varies D alone. The overshoot falls from 27 % to 6.5 % and the rise time lengthens from 0.27 to 0.46 s, while the steady-state error stays between 9 and 13 mm throughout.

Raising K from 30 to 100 $\mathrm{N\,m^{-1}}$ — a factor of 3.3 — cuts the steady-state error from 26.7 to 8.1 mm, also a factor of 3.3. The residual is a compliance, not a tracking failure: the arm stops where the virtual spring balances the structure's restoring force, so it scales as $1/K$.

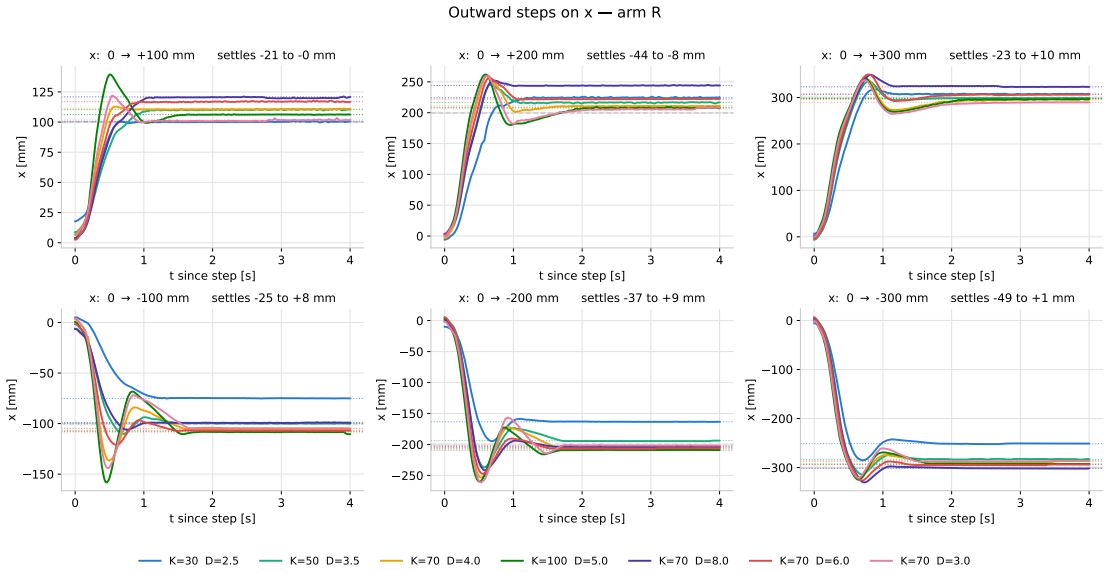


Figure 15.4: Outward steps on the x axis, every gain setting overlaid. Top row: steps away from the rest pose in the positive direction; bottom row: the negative direction. Amplitude grows left to right. Dotted lines mark the level each run settles at. The returns to the rest pose, omitted here, all settle within a few millimetres.

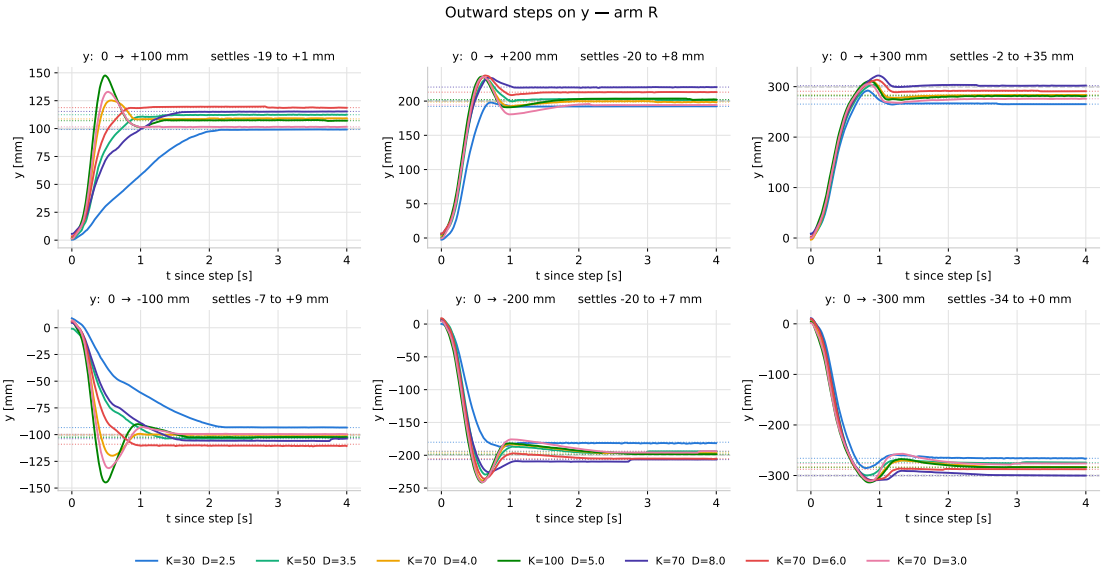


Figure 15.5: Outward steps on the y axis, every gain setting overlaid. Top row: steps away from the rest pose in the positive direction; bottom row: the negative direction. Amplitude grows left to right. Dotted lines mark the level each run settles at. The returns to the rest pose, omitted here, all settle within a few millimetres.

15.3 Trajectory tracking

Three closed planar shapes were tracked — a circle, a figure-eight and a square — each of 0.1 m characteristic size, executed with a period of 13 s at $K = 70\,\mathrm{N\,m^{-1}}$ and $D = 4\,\mathrm{N\,s\,m^{-1}}$, at a height taken from the measured envelope. The square is the polygonal case: it adds corners, where the reference direction changes discontinuously and the arm must reverse a compliant structure.

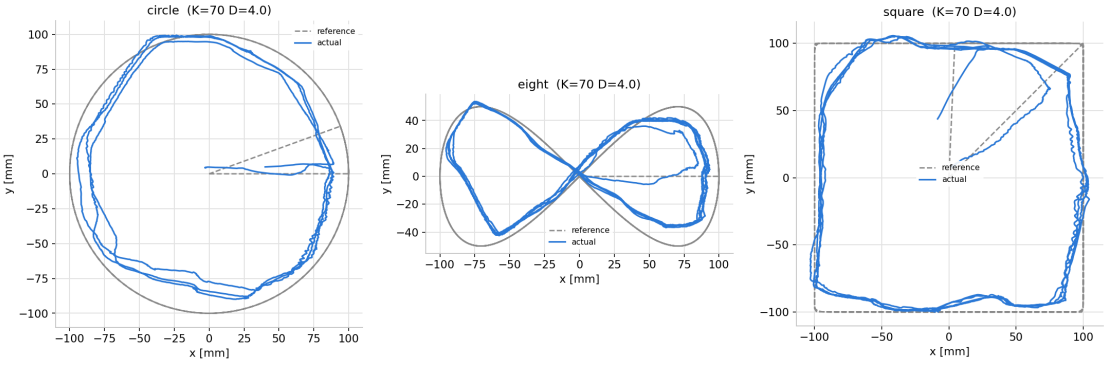
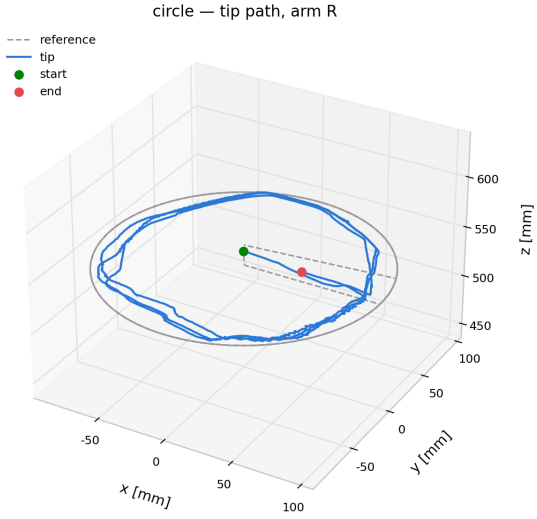
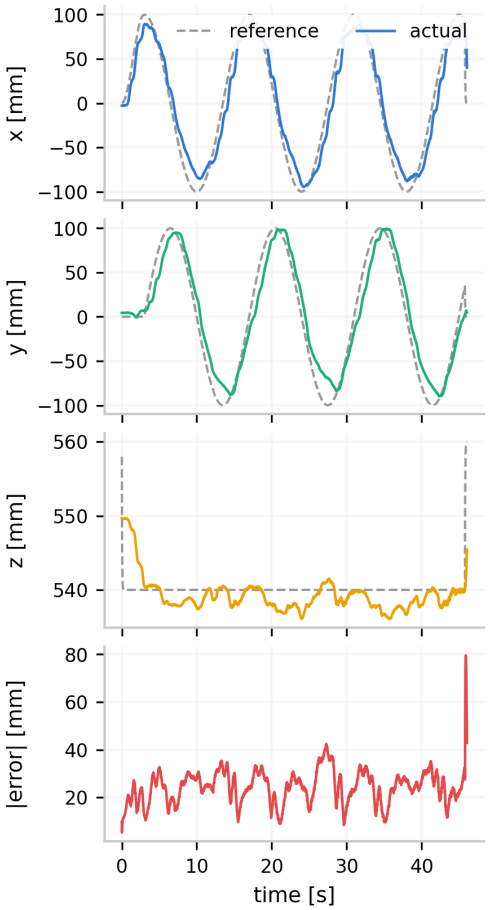


Figure 15.6: Reference and measured tip path for the three shapes.

Figures 15.7 to 15.9 pair, for each shape, the tip path in space with the same run resolved per axis against time.

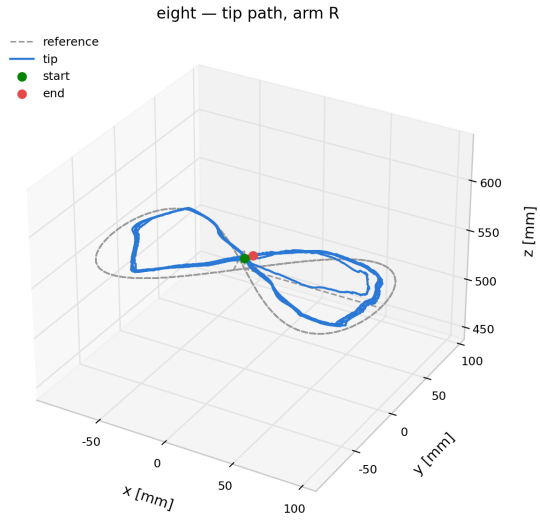


(a) Tip path in space.

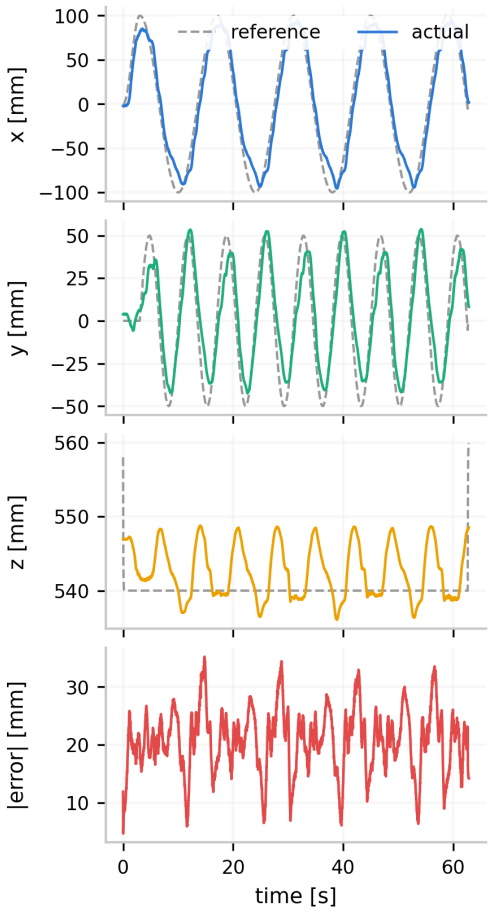


(b) Per-axis tracking and error.

Figure 15.7: Circle.

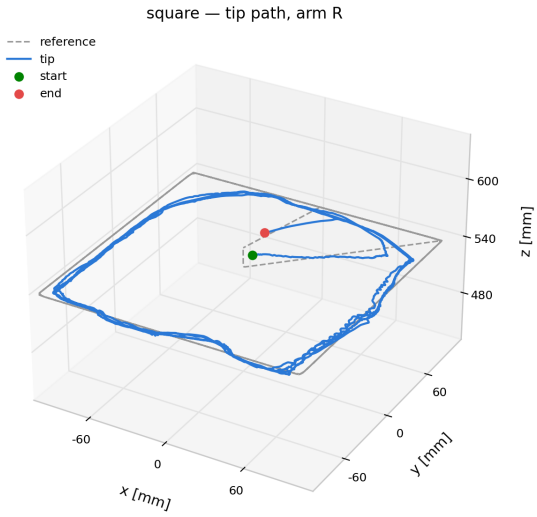


(a) Tip path in space.

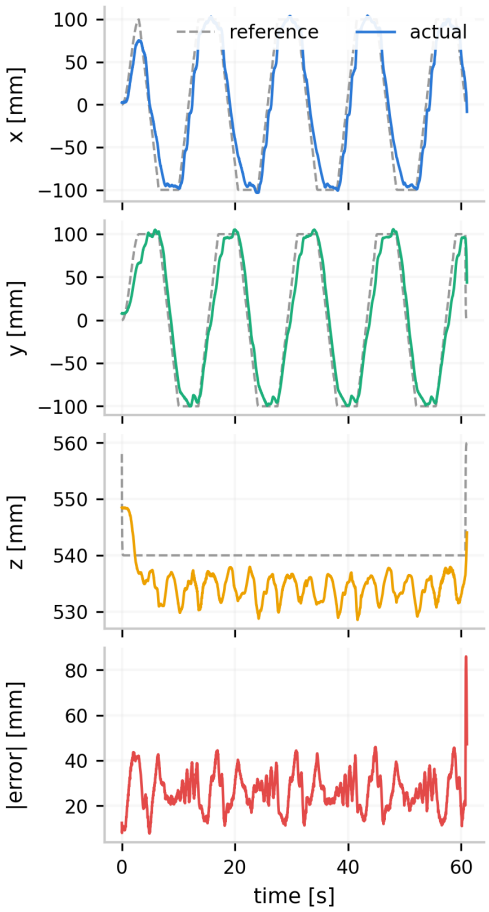


(b) Per-axis tracking and error.

Figure 15.8: Figure-eight.



(a) Tip path in space.



(b) Per-axis tracking and error.

Figure 15.9: Square.

The shapes were run at a single size, period and gain pair, on both arms; the data shown are from the right one, the left tracking the same reference with a comparable error (RMS 23 and 25 mm on the square). They establish that the controller follows a continuous reference with the expected behaviour and quantify the error at that operating point: in planar motion the arm lags the reference by a roughly constant phase, while the offset that remains once that lag is accounted for is the compliance measured in Section 15.2 rather than a failure of the control law.

Chapter 16

Conclusion

This work set out to bring Virtual Model Control to a bimanual soft continuum platform (designed, 3D-printed and assembled as part of this thesis) and to make it regulate contact force without a force sensor. The experimental part answered, in order, the questions Chapter 12 posed:

- the structural stiffness and damping were identified from two independent sensing sources;
- the transmission factor was calibrated against a load cell and found to be configuration independent across three different arm poses, which licenses treating it as a single scalar;
- with that factor in place the arm commands a contact force using nothing but a model of itself, accurately below about 2 N and unreliably above it, where a friction limit at the contact takes over;
- closing the loop on the cell does not remove that ceiling but makes the controller aware of it, to the point of recovering a contact the model had no representation of;
- in free motion the workspace was measured rather than derived and some points of it were used as target points for step responses, demonstrating how varying K and D independently shapes the transient and steady-state behavior.

The chapter closes with a QR code linking to videos recorded during the work: system identification, the workspace scan, step responses, force modulation and a bimanual pick and place. The pick and place experiment is offered as a qualitative result; what the demonstration does establish is feasibility: that the components validated separately in Chapters 13 to 15 compose into a working task without additional machinery.

Three aspects of the pick-and-place demonstration are worth noting:

- The virtual elements were anchored at the tips, while the object sat lower down the bodies, at roughly $s = 0.8$: the arms therefore held it with a stretch of structure the controller was never told about. Nothing in the control law needs the contact location — the force estimator of Chapter 11 takes the contact normal as given and never identifies where along the body the contact sits — and the grasp works for the same reason.
- The compliance is doing work that no model supplies. Neither arm has a model of the object, and none is needed: the structure conforms to whatever it meets, and the interaction force is whatever the virtual stiffness and the residual deflection produce. This is the property the platform exists to exploit, and it is why the force estimator of Chapter 11 needs no model of the environment.
- The grasp succeeded only after repeated attempts: the same compliance that lets the arms conform to the object also lets them slide along it, and raising the virtual stiffness did not remove the effect. The mechanism is the one already measured in Section 14.2, where the tip begins to slide down the load cell once the tangential load at the contact exceeds what friction can hold. The run was also the least explored: a single linear spring, with the geometry hard-coded and no force regulation at all. The gradient descent of Chapter 11 was not enabled, so the grasp force was whatever the fixed virtual stiffness and the residual deflection produced, exactly as in the position regulator of Chapter 10.

Scope and continuation. Coordinated bimanual control, as named in the title, is the problem this platform is now equipped to pose. What has been delivered is the calibrated single-arm layer on which it rests: the experiments stop at single-arm interaction against a rigid, instrumented obstacle. Extending them to manipulation proper — grasping compliant objects of different stiffness, characterising when a whole-body grasp holds and when it slips, and regulating the internal force between two arms rather than the force against ground — is the natural continuation of this work.

Varying the virtual model itself belongs to that continuation as well, and the pick and place is the clearest case for it: the library of force laws of Section 10.4, the anisotropic and constrained variants of Section 10.4.2, and the number and placement of the control points were all left at their defaults. Identifying the object itself, rather than treating it as an unknown passive environment, is the further step beyond that.

That continuation is now supported by a platform that is calibrated rather than merely assembled: the structural parameters are identified from two independent sensing sources, the transmission factor is calibrated against an external sensor, the force estimator's accuracy and its failure mode are both characterised, and the workspace is measured. The measurement infrastructure is part of that inheritance — every run is archived with its full parameter context and analysed from the archive rather than from live inspection, so results recorded a year apart remain comparable.

The limits established here are equally part of it, and they define where the next work

should begin: the unmodelled frictional term of Section 13.3, the mechanical ceiling near 2 N of Section 14.2, and the viscoelastic creep that forced a deadband into the closed loop. None of these is a defect of the control architecture; all three are properties of the plant that a controller must be designed around, and knowing them in advance is worth more than a demonstration that avoids them.



Figure 16.1: Video of some parts of the experiments.

Bibliography

Federico Califano, Ramy Rashad, Cristian Secchi, and Stefano Stramigioli. On the Use of Energy Tanks for Robotic Systems. In *Human-Friendly Robotics 2022*, volume 26 of *Springer Proceedings in Advanced Robotics*, pages 174–188. Springer, 2023. doi: 10.1007/978-3-031-22731-8_13.

Cosimo Della Santina, Antonio Bicchi, and Daniela Rus. On an Improved State Parametrization for Soft Robots With Piecewise Constant Curvature and Its Use in Model Based Control. *IEEE Robotics and Automation Letters*, 5(2):1001–1008, 2020a. doi: 10.1109/LRA.2020.2967269.

Cosimo Della Santina, Robert K. Katzschmann, Antonio Bicchi, and Daniela Rus. Model-Based Dynamic Feedback Control of a Planar Soft Robot: Trajectory Tracking and Interaction with the Environment. *The International Journal of Robotics Research*, 39(4):490–513, 2020b. doi: 10.1177/0278364919897292.

Cosimo Della Santina, Christian Duriez, and Daniela Rus. Model-Based Control of Soft Robots: A Survey of the State of the Art and Open Challenges. *IEEE Control Systems*, 43(3):30–65, 2023. doi: 10.1109/MCS.2023.3253419.

Huxin Gao, Xiaoxiao Yang, Xiao Xiao, Xiaolong Zhu, Tao Zhang, Cheng Hou, Huicong Liu, Max Q.-H. Meng, Lining Sun, Xiuli Zuo, Yanqing Li, and Hongliang Ren. Transendoscopic Flexible Parallel Continuum Robotic Mechanism for Bimanual Endoscopic Submucosal Dissection. *The International Journal of Robotics Research*, 43 (3):281–304, 2024. doi: 10.1177/02783649231209338.

Qinghua Guan, Federico Stella, Cosimo Della Santina, Jinsong Leng, and Josie Hughes. Trimmed Helicoids: An Architectured Soft Structure Yielding Soft Robots with High Precision, Large Workspace, and Compliant Interactions. *npj Robotics*, 1(1):4, 2023. doi: 10.1038/s44182-023-00004-7.

Neville Hogan. Impedance Control: An Approach to Manipulation: Part I—Theory. *Journal of Dynamic Systems, Measurement, and Control*, 107(1):1–7, 1985. doi: 10.1115/1.3140702.

Bryan A. Jones and Ian D. Walker. Kinematics for Multisection Continuum Robots. *IEEE Transactions on Robotics*, 22(1):43–55, 2006. doi: 10.1109/TRO.2005.861458.

Daniel Larby and Fulvio Forni. A Generalized Approach to Impedance Control Design for Robotic Minimally Invasive Surgery. *IFAC-PapersOnLine*, 56(2):8548–8555, 2023. doi: 10.1016/j.ifacol.2023.10.015.

Daniel Larby and Fulvio Forni. Optimal Virtual Model Control for Robotics: Design and Tuning of Passivity-Based Controllers. *IEEE Transactions on Robotics*, 42:439–454, 2026. doi: 10.1109/TRO.2025.3645886.

MIT Leg Laboratory. Spring Flamingo Robot. MIT Leg Laboratory project archive, 2000. Online project archive, accessed 1 September 2026.

Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual Model Control: An Intuitive Approach for Bipedal Locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001. doi: 10.1177/02783640122067309.

Enoch Quaicoe, Ayman Nada, Hiroyuki Ishii, and Haitham El-Hussieny. Cooperative Control of Bimanual Continuum Robots for Automated Knot-Tying in Robot-Assisted Surgical Suturing. *Journal of Robotics and Control (JRC)*, 5(4):1149–1165, 2024. doi: 10.18196/jrc.v5i4.21617.

Yi Shen, Ruochen Tai, Jinghao Zhang, Linan Deng, Ye Yuan, Rong Su, Fumin Zhang, and Han Ding. Planning and Motion Control for Underwater Bimanual Soft Manipulator in Underwater Grasping Task. *IEEE/ASME Transactions on Mechatronics*, 29 (4):2487–2498, 2024. doi: 10.1109/TMECH.2023.3333317.

Francesco Stella, Qinghua Guan, Cosimo Della Santina, and Josie Hughes. Piecewise Affine Curvature Model: A Reduced-Order Model for Soft Robot-Environment Interaction Beyond PCC. In *2023 IEEE International Conference on Soft Robotics (RoboSoft)*, pages 1–7. IEEE, 2023a. doi: 10.1109/RoboSoft55895.2023.10121939.

Francesco Stella, Josie Hughes, Daniela Rus, and Cosimo Della Santina. Prescribing Cartesian Stiffness of Soft Robots by Co-Optimization of Shape and Segment-Level Stiffness. *Soft Robotics*, 10(4):701–712, 2023b. doi: 10.1089/soro.2022.0025.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

Congnan Yang, Jianwen Liu, Tong Cai, Yijie Zhao, Wenhao Wang, Bolong Liu, and Xiaojun Xu. Design of Multi-Legged Locomotion Control System for Quadruped Robots Based on Virtual Model Control with BP Neural Network. *Machines*, 14(2): 184, 2026. doi: 10.3390/machines14020184.

Yi Zhang, Daniel Larby, Fumiya Iida, and Fulvio Forni. Virtual Model Control for Compliant Reaching Under Uncertainties. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 795–801. IEEE, 2024. doi: 10.1109/IROS58592.2024.10801592.