

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
DEPARTMENT OF ENGINEERING “ENZO FERRARI”

INTERNATIONAL DOCTORATE SCHOOL IN ICT - XXXVII CYCLE

**Methods for a Novel Collaborative
Robotics: From Programming to Human
Skills Transfer**

Candidate:

Giovanni Braglia

Supervisor:

Prof. Luigi Biagiotti

Coordinator:

Prof. Luigi Rovati

ACADEMIC YEAR 2023-2024



Doctoral research thesis co-funded under the National Operational Programme for Research and Innovation 2014-2020 (CCI 2014IT16M2OP005), REACT-EU ESF resources, Action IV.4 "Doctorates and research contracts on innovation topics" and Action IV.5 "Doctorates on Green topics."

Index

Index	iii
1 Introduction	1
1.1 Contributions	2
1.2 Robotic Learning from Demonstration	2
1.3 Human-Robot Interaction	5
1.3.1 No contact interaction	5
1.3.2 Contact interaction	7
1.4 Organization of the Thesis	9
2 Learning from Demonstration	11
2.1 Demonstration	12
2.2 Motion Representation and Encoding	13
2.2.1 Potential Fields	14
2.2.2 Dynamical Systems	15
2.2.3 Probabilistic	17
2.3 Robot control for Human-Robot Interaction	18
2.4 Proposed framework	21
3 Demonstrations alignment for consistent skill extrapolation	24
3.1 Dynamic time warping	25
3.1.1 Limitations of DTW	27
3.2 The Spatial Sampling algorithm	29
3.2.1 From time to arc length domain	29
3.2.2 Sampling in space	31
3.2.3 Arc length and its analogies	34
3.2.4 Optimization of Parameter Δ	35
3.3 Experiments and evaluation	37
3.3.1 Alignment Algorithms Comparisons	37
3.3.2 Skill Synthesis from Demonstrations	40
3.3.3 Discussion	42

4	Motion Representation	45
4.1	Skill Expression	45
4.1.1	Radial Basis Functions	48
4.1.2	Bernstein Polynomials	48
4.1.3	Fourier Series	49
4.2	Skill encoding with GMM	50
4.2.1	Gaussian Mixture Models	51
4.2.2	Gaussian Mixture Regression	52
4.3	Skill Execution:DMP	54
4.3.1	Quaternion-based representation for DMP	56
4.3.2	A reversible DMP formulation	57
4.4	Skill Execution:LQR	62
4.4.1	Batch formulation of two-step LQR	63
4.4.2	Batch formulation of LQR with CP	66
4.4.3	Recursive formulation of two-step LQR	67
4.5	An optimal control formulation of DMP	68
4.5.1	Recursive LQT with CP	71
4.5.2	Discussion	73
5	HRI with Geometric Dynamic Movement Primitives	75
5.1	Geometric Dynamic Movement Primitives	76
5.2	Phase Optimization in Geometric DMP	77
5.2.1	Comparison with other DMPs	80
5.3	Human-in-the-loop Geometric DMP	83
5.3.1	Passivity Analysis	84
5.4	Experiments and Results	86
5.4.1	Experimental Setup	87
5.4.2	Practical Stability Analysis	87
5.4.3	Insertion Task	92
5.4.4	Discussion	94
6	HRI with a minimum-jerk approach	97
6.1	Preliminaries	99
6.1.1	Definition of the virtual constraint	99
6.1.2	Tracking problem statement	100
6.2	The Gauss-Newton Algorithm	101
6.3	A minimum-jerk approach to handle EDS	104
6.4	Experiments and Results	107

6.4.1	Experimental Setup	107
6.4.2	Center-reaching task	108
6.4.3	Target following task	109
6.4.4	Discussion	114
7	Representing tasks on different domains	116
7.1	Task Encoding	119
7.1.1	Spatial Encoding based on B-Splines	119
7.1.2	Temporal and Sequential Encoding Based on HMM	120
7.2	Proposed Method	121
7.2.1	Spatial Modulation with Dynamical Control Points	122
7.2.2	Temporal Modulation with HMM	124
7.2.3	Switching between Trajectory and Symbolic Domains	127
7.3	Experiments and Results	128
7.4	Discussion of limitations and challenges	133
7.5	Conclusions	134
8	Conclusion	137
8.1	Discussion and limitations	137
8.2	Challenges in human-robot interaction	140
8.3	Future directions of research	142
	Appendices	144
A	Geometry of curves	145
A.1	Preliminaries	145
A.2	Tangent line and osculating plane	147
A.3	Length of a curve and arc length parametrization	148
B	Orientation	151
B.1	Introduction	151
B.2	Algebra of Quaternion	152
B.3	Quaternion as rotation operator	152
C	Signal alignment metrics	154
C.1	Alignment Metrics for Comparisons	154
C.1.1	Rho (ρ)	154
C.1.2	Entropy (H)	155
C.1.3	Sum-Normalized Cross-Spectral Density (snCSD)	156

C.1.4	ℓ_2 norm error	157
C.2	Skill Synthesis from Demonstrations	157
C.2.1	Hausdorff distance d_H	157
C.2.2	Dynamic Time Warping (DTW) score d_{DTW}	157
D	Additional Material to Chapter 6	158
D.1	Transfer matrices computation	158
D.2	Inverse LQT	159
	List of the author's publications	162
	References	163

List of Figures

2.1	Learning from Demonstration general architecture.	11
2.2	LfD steps of the proposed framework.	23
3.1	Learning from Demonstration (LfD) framework.	24
3.2	Skill extrapolation with GMM	25
3.3	Aligning two time series with DTW.	26
3.4	DTW cost matrix	28
3.5	Changing DTW reference	29
3.6	Curve representation with time and arc length	31
3.7	SS Algorithm	32
3.8	Choosing Δ for SS	36
3.9	Time vs arc-length on LASA dataset	38
3.10	Box plot LASA dataset	39
3.11	Robot experiments snapshots	41
3.12	Barycenter computation	41
3.13	Box plots for approximation metrics result.	43
3.14	Autolab Co-Manipulation dataset	44
4.1	Approximation with basis functions	48
4.2	Varying the number of basis functions	50
4.3	Varying number of GMM components	54
4.4	DMP block scheme	60
4.5	DMP properties	61
4.6	Performing 2-step LQR on Letters' dataset	65
4.7	LQT-CP with via-points	70
4.8	LQT-CP vs DMP	74
5.1	Proposed GDMP framework	77
5.2	Demonstrated curve used for optimization.	79
5.3	Optimization phase profiles	80
5.4	Optimization problem results	81
5.5	GDMP vs DMPs in minimum time problem	82

5.6	Human-robot interaction framework	83
5.7	Scaling matrix $\mathbf{E}$	84
5.8	Co-manipulation experiments	86
5.9	Trend surfaces	88
5.10	Phase vs exchanged power	89
5.11	Variation of θ with respect to different forces $\mathbf{F}_h$	89
5.12	Framework linearization	90
5.13	Nyquist diagram	91
5.14	Phase margin	91
5.15	Insertion task snapshots	92
5.16	Trajectory recordings for insertion task	94
5.17	GDMP vs other DMPs in insertion task	95
6.1	Euclidean distance singularity	98
6.2	EDS visualization	103
6.3	Trend of the cost in (6.2) for different path deviations.	103
6.4	Snapshots of experiments	104
6.5	Robot experimental results	104
6.6	Arc length velocity profile	106
6.7	Controller framework	108
6.8	Experimental results of the center-reaching task.	110
6.9	Box plots for the center-reaching task.	111
6.10	Experimental results of the target following task.	112
6.11	Phase trajectories	113
7.1	Movement primitives example	116
7.2	B-spline trajectory	120
7.3	Forward-chained HMM	121
7.4	Proposed framework	123
7.5	B-spline with associated dynamical systems	124
7.6	B-spline and HMM combined together	125
7.7	Control at both trajectory and symbolical level	127
7.8	Snapshots from experiments	130
7.9	Box plots of experiments	131
7.10	Three experiments using the same obstacle trajectory	136
A.1	Osculating plane α visualization for Definition A.2.2.	148
A.2	Characteristic versors of a curve $\boldsymbol{\mu}$ at a point $\mathbf{p}(t)$	149

A.3	Inscribed polygonal line γ in a curve μ	149
B.1	Rotation matrices	151

List of Tables

2.1	Adopted notation during the thesis.	23
3.1	Synchrony metrics	40
3.2	Approximation metrics	42
6.1	Errors between EE and target position and DSJ.	114
7.1	Validation tests results	131
8.1	NASA Task Load Index	140

*Dedicated to my parents
and to my dog Zuppa.*

Abstract

Learning from Demonstration (LfD), a key area in robotics, allows robots to learn complex tasks through demonstrations, aiming to make robot programming more accessible to non-expert users. This thesis addresses the fundamental challenges of skill extraction, time-decoupling, and motion adaptation, developing new algorithms that improve the robot's ability to learn, adapt, and interact safely with humans in dynamic environments. One of the central contributions of this work is the development of a novel approach to handle temporal misalignments in demonstration data, which offer accurate skill extraction. Traditional techniques like Dynamic Time Warping (DTW) fall short in fully decoupling timing from skill representation. To address this, in the thesis a Spatial Sampling (SS) algorithm is proposed, which extracts the geometric information of demonstrated tasks independently of timing variations. This allows for time-invariant skill representation, improving the robot's capacity to learn more accurately. The SS method is integrated with Gaussian Mixture Regression (GMR) to generate robust trajectories that can adapt to various execution speeds, providing a flexible framework for skill transfer. This system was validated in rehabilitation tasks, where accurate and adaptable motion reproduction is critical. In collaborative robotics, the thesis advances the application of Virtual Fixtures (VFs) to enhance co-manipulation tasks, where robots assist humans by constraining their motion along specific paths. While VFs are widely used to improve task precision and reduce cognitive load, they face challenges in managing singularities, where multiple valid paths exist. In the thesis, a novel approach to handle these issues is introduced, ensuring smoother and more intuitive human-robot interaction in complex task settings. This contribution enhances the robot's ability to assist in tasks that require both precision and adaptability. Another key aspect of the research focuses on extending the capabilities of Dynamic Movement Primitives (DMPs), a widely-used method for encoding robot motion. DMPs are attractive for their stability and flexibility, but their application in environments with dynamic changes had limitations. In the thesis an extension to DMPs has been provided, that allow robots to adjust their trajectories in real time, reacting to user inputs in Human-Robot Interaction (HRI) architecture. This enables robots

to perform complex tasks, such as co-manipulation, while maintaining robustness, adaptability and, most of all, safety. Safety in human-robot collaboration is a central concern, especially in shared workspaces where humans and robots must work together without compromising performance or safety. To address this, the thesis additionally introduces a real-time obstacle avoidance algorithm that allow robots to modify their paths in response to human movements, ensuring safe interaction without interrupting task execution. The proposed method uses vision-based monitoring to detect potential collisions and dynamically adjusts the robot's speed or position to avoid them. Once the workspace is clear, the robot resumes its original task path, balancing safety with operational efficiency. In summary, this thesis contributes to the fields of robot learning and human-robot interaction by developing novel algorithms for skill extraction, motion adaptation, and safety in collaborative environments. These advancements enable robots to learn from limited demonstrations, adapt to dynamic environments, and interact safely with humans, making them more versatile and accessible for applications such as rehabilitation, industrial automation, and collaborative tasks.

Introduction

In recent years, collaborative robotics has gained significant attention as robots are beginning to work alongside humans rather than performing tasks in isolation. Traditionally, industrial robots have been confined to separate environments, often operating within cages to ensure the safety of human workers (Deshpande, 2022; L. Zhang et al., 2020b). Collaborative robotics breaks this paradigm, enabling robots and humans to share the same workspace and perform tasks together in a more integrated and dynamic manner.

The concept of collaborative robotics is closely tied to Human-Robot Interaction (HRI), a field dedicated to developing algorithms and technologies that enhance cooperation between robots and their human counterparts. Within this domain, a particularly promising approach is Learning from Demonstration (LfD), which offers significant potential for improving collaborative performance, especially in therapeutic and assistive applications (Hogan & Sternad, 2013; Lauretti et al., 2017).

LfD presents two primary advantages (Calinon, 2009). First, through imitation learning, robots can achieve task flexibility while accommodating user-specific requirements. This means that a robot can learn to perform a task while accounting for the unique needs or limitations of an individual, such as a patient undergoing therapy. Second, LfD emphasizes user-friendliness, making it an accessible method for integrating robots into everyday human environments. By eliminating the need for extensive programming skills or specialized effort, LfD allows users to teach robots tasks in an intuitive and straightforward way, fostering broader adoption and usability of robotic systems in diverse contexts.

In this sense, the title *"Methods for a Novel Collaborative Robotics: From Programming to Human Skills Transfer"* takes inspiration from the recent literature in LfD and robotics scenarios where the human is involved actively or passively within the robot workspace while it is accomplishing a certain task. As this cooperation setup is becoming increasingly common, the purpose of this

thesis is to study recent LfD literature and combine it with the integration of the human as a fundamental building block of such techniques. With specific study cases and experiments where the human participates in the analyzed algorithms, the subsequent chapters address the problem of having a human in the control loop. Special considerations are made about the safety and robustness of the frameworks developed throughout this work.

1.1 Contributions

This work provides insights into the integration of humans within HRI frameworks. In collaborative applications, the human can either be in contact with the robot or not, requiring different types of analysis and considerations. In this sense, the main contribution of this work is to provide a generalized framework for implementing and evaluating human-in-the-loop applications. This is achieved by exploiting LfD as an initial step to transfer human knowledge to the robot, utilizing task encoding techniques to memorize and reproduce the target task, and conducting real-world experimental tests of collaborative tasks. These tasks leverage proper metrics to provide a clearer description and evaluation of cooperative performance. Each of these steps is analyzed throughout the work, enriched with novel considerations and algorithms designed to improve the addressed research topics.

1.2 Robotic Learning from Demonstration

Robot Learning from Demonstration (LfD) is a framework for teaching robots tasks through a few demonstrations (Pastor et al., 2009). The primary goal of LfD is to enable intuitive skill transfer, reduce the need for extensive data, and make robots more accessible to individuals without programming expertise (Calinon et al., 2010). In a typical LfD architecture, a set of demonstrations is analyzed to extract the most informative content, commonly referred to as a skill in the literature (Calinon, 2009; Gaspar et al., 2018). This extracted skill is then transmitted to the robot, which uses a controller to interact with its environment and adapt its behavior to accommodate internal or external constraints (Bianco & Ghilardelli, 2017).

LfD facilitates the transfer of skills and knowledge to robots via an external operator, whose goal is to teach the robot to execute a specific task and enable

autonomous repetition of the learned behavior (Billard et al., 2008). HRI benefits from LfD for two main reasons:

- **Task Flexibility and User-Specific Requirements:** Imitation learning combines task adaptability with user-specific needs, enhancing both human and robot awareness of limitations.
- **User-Friendly Interfaces:** Demonstration-based approaches provide an intuitive way of controlling robots, eliminating the need for advanced technical knowledge in task programming.

A central challenge in LfD is the paradigm of interpretation, which addresses *how* and *what* to imitate (Calinon et al., 2007). *How to imitate* encompasses methodologies, control algorithms, and constraints, considering that robots often have different kinematic structures from their teachers and operate in dynamic environments. *What to imitate* focuses on grasping the intent behind demonstrations (the skill) and building internal knowledge that can be rearranged or combined to perform a broader range of movements.

Humans naturally learn and generalize skills from a small number of observations. LfD draws inspiration from this philosophy, aiming to impart knowledge to robots with minimal demonstrations. This requires algorithms capable of extracting meaningful knowledge components, storing them efficiently, and leveraging this prior expertise for future tasks. Skill representation plays a critical role, with motion primitives serving as a common approach (Ijspeert et al., 2013; Ruckert & d’Avella, 2013). Motion primitives are fundamental building blocks, adaptive units of movement that can be combined in series or parallel to create complex behaviors (Fisac et al., 2018). For example, this can involve combining primitives using mixing coefficients to determine their relative influence (Paraschos et al., 2013) or prioritizing learned knowledge (Pignat et al., 2022).

Collaborative robots designed for HRI must be aware of events in their dynamic environment and respond appropriately. This requires robot controllers to feature online awareness of their workspace, enabling motion control and planning that respects task constraints, goals, motion priors, and external events (Kanazawa et al., 2019; H. Liu et al., 2022; Toussaint & Goerick, 2010). For instance, trajectory modulation may involve adding desired via-points or adapting to new final positions or velocities. Furthermore, robots should generalize their learned skills to a broader range of tasks beyond the specific demonstrations. This challenge can be framed as an information fusion problem, where multiple information sources are integrated (Calinon et al., 2007).

The first step in LfD involves an operator demonstrating a desired task to the robot. Demonstrations can be provided through various methods, including simulations (Quigley et al., 2015a), tele-operation (Si et al., 2021), visual systems (Rotithor et al., 2022), or kinesthetic guidance (Simonic et al., 2021). The underlying principle is that learning algorithms should extract useful information from a few demonstrations to reproduce the taught behavior (Billard et al., 2008). This can be conceptualized as identifying a finite and minimal set of movements that form the basis for recreating new skills through combinations of acquired knowledge (Pastor et al., 2009). Achieving this requires addressing two key challenges:

- **Data Alignment:** Demonstrations of the same task may vary in time and geometry. Ensuring semantic alignment of samples is crucial for extracting reliable knowledge from the data (Brunton & Kutz, 2022).
- **Dimensionality Reduction:** Data collected from multiple internal sensors (e.g., end-effector poses, joint angles, forces) and external sources (e.g., IMU/EMG sensors on the teacher) can be of large size. Effective dimensionality reduction is necessary to extract skills from diverse datasets (Huang et al., 2019).

Addressing the data alignment problem enables the training of learning algorithms on coherent and reliable data, while solving the dimensionality reduction problem filters out noisy signals, revealing the underlying hierarchy and priorities in motion patterns (Ficuciello et al., 2018).

The ability to identify and exploit similarities across related tasks is a characteristic of human motor skill learning. Studies have shown that biological systems reproduce complex motor skills by combining a small number of muscle activations (Ruckert & d’Avella, 2013). In robotics, the process of gathering and encoding previously acquired knowledge for actuator interpretation is known as the representation problem (Pastor et al., 2009).

Movement primitives have become a popular method for encoding robot tasks and representing knowledge acquired during demonstrations. These primitives are parameterized representations of elementary movements, typically described as weighted combinations of basis functions to approximate desired behaviors (Calinon et al., 2007; Ijspeert et al., 2013). Proper task execution involves combining a set of primitives selected from a compact dictionary of movements, conditioned by mixing parameters, activation factors, and external events. These param-

eters are integrated through control strategies to achieve the robot's preferred configurations and robustly handle perturbations.

When a robot operates in a dynamic environment, it is challenging to infer new goals from limited initial demonstrations. To enhance efficiency, reliability, and flexibility in task reproduction, additional techniques are required. Robust controllers should incorporate:

- **Motion Planning Algorithms:** For real-time, reactive adaptation to changing environments.
- **Optimal Control Techniques:** To balance execution parameters effectively.
- **Online Learning:** Enabling the robot to learn from trial-and-error experiences and expand its knowledge base.

1.3 Human-Robot Interaction

In a Human-Robot Interaction (HRI) application, the human performs tasks by directly interacting with the robot, utilizing the robot as a specific tool. In modern industries, for example, the demand for collaborative robots is constantly increasing because of their versatility and precision in a wide variety of applications. In a collaborative cell, humans and robots work together, sharing the same workspace. Therefore, the robot must adapt to external events while complying with safety regulations (Khansari-Zadeh & Khatib, 2017). Safe human-robot cooperation can be achieved if the control system can gather information about the state of the robot as well as the state of the worker inside the collaborative workspace. Indeed, when human and robot shares the same workspace, the type of interaction can vary whether the two are in contact or not.

1.3.1 No contact interaction

When no physical contact occurs, HRI emphasizes the concept of awareness, meaning that the robot must continuously monitor human behavior within its workspace. A typical application of this interaction is obstacle avoidance, where controller algorithms are designed to prevent the robot from contacting surrounding objects. For safety reasons, humans are often treated as dynamic obstacles to ensure the avoidance of any dangerous collisions.

In obstacle avoidance scenarios, the robot adjusts its trajectory based on the detected position of target objects to prevent collisions (H. Liu et al., 2022). Safety margins are usually incorporated to maintain a buffer between the robot and the object; however, sudden or unexpected movements can still result in collisions if the algorithm is not reactive enough (Merckaert et al., 2022). Furthermore, once the operator exits the workspace, the robot should restore its original path to resume operations under nominal conditions (Chiaravalli et al., 2018). This dynamic interplay between collision avoidance and task execution highlights the importance of robust planning and control strategies.

Motion planning for collision avoidance often takes the form of an optimal control problem (Kanazawa et al., 2019). These approaches are highly valued for their ability to minimize cost functions while adhering to constraints such as safety requirements, temporal considerations, and task-specific metrics (Merckaert et al., 2022; Zanchettin et al., 2015). For example, Covariant Hamiltonian Optimization for Motion Planning (CHOMP) (Zucker et al., 2013) exemplifies this principle by generating collision-free and locally optimal trajectories through a covariant gradient update mechanism. Similarly, parallel optimization schemes have been successfully applied in collaborative scenarios, such as the manipulation of cable-towed loads by multiple robots (Yang et al., 2022).

Probabilistic inference offers another effective framework for collision avoidance, replacing traditional cost functions with probabilistic models that capture task constraints, motion priors, and goals as conditional dependencies (Fisac et al., 2018; Mukadam et al., 2018; Toussaint & Goerick, 2010). In this context, techniques such as Gaussian Mixture Models (GMMs) have been utilized to encode and retrieve tasks through regression methods, as demonstrated in LfD frameworks (Calinon et al., 2007). These methods provide a statistical perspective to trajectory optimization, enabling flexibility and adaptability in dynamic environments.

An alternative approach to collision avoidance is the use of sample-based planning algorithms, which rely on random sampling of the environment to identify feasible paths (Kavraki & LaValle, 2016). Techniques such as probabilistic roadmaps and rapidly exploring random trees enable efficient exploration of high-dimensional spaces. For instance, sampled via-points can be used to generate candidate trajectories that adhere to kinodynamic constraints (Jankowski et al., 2023). Moreover, local replanning methods, such as those based on Bi-directional Rapidly-exploring Random Trees (Bi-RRT), allow the robot to dynamically adapt its trajectory when a moving obstacle is detected, maintaining continuity with

the original task (Han et al., 2018).

Reactive control methods offer a more direct approach to collision avoidance, leveraging potential field techniques originally proposed by (Khatib, 1986). In this framework, virtual repulsive fields are associated with obstacles to prevent collisions, while attractive fields guide the robot toward its target. These fields create a dynamic system where the robot’s trajectory evolves under the influence of virtual forces. This concept has been further refined to treat trajectories as elastic bands that deform in response to external forces, enabling real-time trajectory adjustments (Quinlan & Khatib, 1993). Over time, potential field methods have become a standard tool for real-time trajectory planning, effectively balancing reactivity and computational efficiency (Flacco et al., 2012; H. Liu et al., 2022).

1.3.2 Contact interaction

Unlike scenarios where there is no physical contact, HRI involving physical contact presents distinct challenges and considerations. In such cases, the robot is inherently aware of the human’s position due to the direct interaction, eliminating the need for external awareness mechanisms. However, safety becomes a more intricate issue to address. In non-contact scenarios like obstacle avoidance, safety can often be reduced to a binary assessment of collision detection. When physical contact occurs, safety must be evaluated from multiple perspectives, incorporating both objective and subjective metrics (Braglia et al., 2025; Ferraguti et al., 2019; Hart, 1988).

A central concept in physical HRI is the notion of Virtual Fixtures (VF), first introduced in (Rosenberg et al., 1993). Virtual Fixtures are collaborative control solutions designed to enhance or assist human manipulation tasks by constraining or regulating motion within specific regions of the robot’s workspace. These constraints often involve attaching tools to a robotic arm, which is predominantly controlled by the human operator. The robot controller actively enforces these constraints by evaluating tool motion against predefined paths and restricted areas. Commands that would cause the manipulator to deviate from the specified path are attenuated or canceled by the controller.

It is important to distinguish between geometric and motion constraints within VF systems. Geometric constraints define spatial limitations, allowing the user to control task execution timing freely. In contrast, motion constraints impose both spatial and temporal regulations, requiring the user to adhere to a predefined

timing along with the spatial restrictions.

Direct physical interaction with the robot end effector allows humans to provide immediate feedback, fostering intuitive task execution. Such applications span various domains, including healthcare and industrial settings. In healthcare, VF techniques have been utilized in robotic surgery and rehabilitation tasks (Savarimuthu et al., 2017; Selvaggio et al., 2018), while in industrial applications, they assist in precision operations like assembly and drilling. The direct physical interaction places stringent demands on ensuring human safety, which becomes a central concern in these applications.

Several studies have explored the use of VF in diverse HRI applications. In the domain of minimally invasive robotic surgery (MIRS), reference points are registered to define VFs that guide surgical instruments (Selvaggio et al., 2018). Similarly, in co-manipulation tasks, VF systems facilitate precise positioning of the robot’s end effector for specific tasks (Cacace et al., 2019). Upper-limb rehabilitation often employs VFs with assistive forces, such as springs, to guide patients along predetermined paths (L. Zhang et al., 2020a). In robotics-assisted minimally invasive surgery (RAMIS), a combination of Remote Center of Motion (RCM) constraints and Forbidden Region VF enhances task execution (Kastritsi & Doulgeri, 2022). Applications such as retinal vein cannulation have also benefited from VF-based control strategies (Bettini et al., 2004).

Dynamic Movement Primitives (DMPs) have been applied to various physical HRI scenarios where human-robot collaboration is integral. For instance, in (Lauretti et al., 2017), users’ arms were actively guided by the robot during daily living activities. Passive assistance has also been explored, as demonstrated in (Braglia et al., 2025; Sidiropoulos & Doulgeri, 2021), where force measurements are used to guide the robot along virtual constraints derived from DMPs.

Probabilistic models have also been integrated into VF frameworks to allow seamless switching between multiple virtual guides. In (Raiola et al., 2018), a co-manipulation framework was developed where a library of virtual guides coexisted during task execution. The robot’s end effector was attracted to the guide with the highest likelihood based on its position, enabling adaptive and efficient task execution.

In the rehabilitation context, impedance controllers have been adapted for VF-based velocity tracking. For example, in (Asl et al., 2020), a controller was used to assist patients in rehabilitation tasks by guiding them along predefined paths.

In this work, the integration of VF within a LfD framework has proven effec-

tive in guiding co-manipulation tasks. Initially, a VF is demonstrated by guiding the robot to execute a target exercise. Once these demonstrations are provided, the VF is extracted and encoded in the robot controller. This encoding ensures that an end-user in physical contact with the robot experiences a constraint in the robot's workspace, facilitating correct and intuitive execution of the target exercise. By combining physical guidance with intelligent control, VF-based HRI systems offer robust solutions for enhancing safety and performance in collaborative tasks.

1.4 Organization of the Thesis

The aim of this chapter was to provide a brief literature review of the topics that will be addressed in the following pages. In particular, the content of each upcoming chapter is summarized as follows:

- **Chapter 2:** Provides an overview of the collaborative framework adopted in subsequent chapters. It analyzes LfD with a particular focus on encoding techniques, offers an overview of these methods, and examines robot interaction with the environment to introduce the proposed HRI framework.
- **Chapter 3:** Delves into LfD from the perspective of data collection and skill extrapolation. It addresses the problem of trajectory time synchronization and introduces a novel algorithm designed to improve the consistency of extrapolated skills.
- **Chapter 4:** Analyzes common encoding techniques such as Gaussian Mixture Models (GMM), Dynamic Movement Primitives (DMP), and Linear Quadratic Tracking (LQT). These techniques are introduced in detail and serve as the foundation for applications developed in subsequent chapters.
- **Chapter 5:** Presents results on a novel DMP architecture within a human-in-the-loop framework. It evaluates previous state-of-the-art DMP solutions and provides safety guarantees for human interaction.
- **Chapter 6:** Explores guidance applications from an optimization perspective. It highlights the limitations of existing techniques and proposes a solution to address these limitations using the LQT algorithm.
- **Chapter 7:** Summarizes preliminary results from a cooperative scenario where the human is treated as an obstacle to avoid, rather than being in

contact with the robot. Additionally, it discusses a novel architecture and its implications, merging different task representation domains.

- **Chapter 8:** Summarizes the key insights from the previous chapters and concludes this work.

Learning from Demonstration

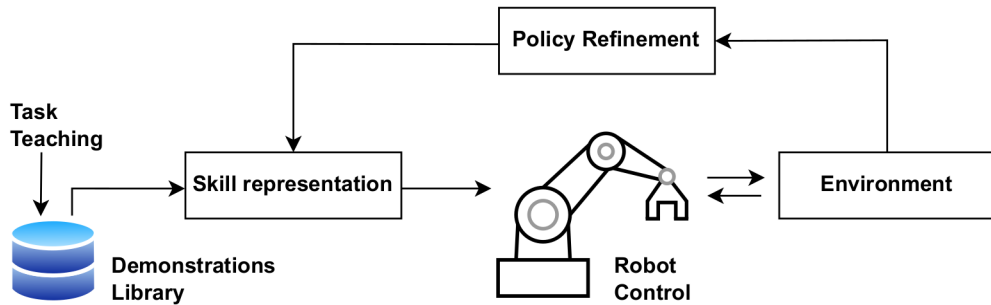


Figure 2.1: Learning from Demonstration general architecture.

The schematic reported in Fig. 2.1 briefly summarizes the methodology adopted throughout this work, which actually reflect a common framework in LfD. Depending on the final desired task, a subject (or master) is taken as an expert to teach the robot what it will be expected to do. Every information is then stored into a demonstration library, which may contains single or multiple demonstrations of the task.

A key aspect of LfD come from its similarity in the way human learns, most of the time by observing just few trials before acquire complete control over the desired action. Indeed this not just involve observation, but also a cognitive effort to extrapolate the essential information to reproduce what the master is teaching. In robotics, this often comes under the name of skill representation, where the term *skill* refer to the core information content which helps reproduced the target task.

Once these information are extrapolated and opportunely processed to elaborate commands for the robot, a controller ensures correct reproduction of the task. Additionally, the controller is responsible to handle both the robot kinematic and internal/external constraints, ensuring robustness and stability against possible disturbances.

While this is sufficient for most of the task where the robot is asked to perform continuously the same actions, there might be situations in which disturbances are intentionally provided to the robot. For example, a target task demonstrated in a certain circumstance might not be sufficient to extend the relative skill to a different scenario. Even more, the same skill could be totally inappropriate to other contexts. In all these cases, the disturbance measurements can be used to update or modify the current skill using specific refinement techniques.

This chapter aims at giving a wide panoramic over the general framework of figure 2.1. In particular, each block of the figure will be further detail, providing detailed discussions whenever the topic encounters some areas explored during the thesis.

2.1 Demonstration

As the LfD term itself suggest, the former step of learning a certain task is to provide a demonstration of it. At the early stages of LfD, this was considered as an enormous step forward in the programming of robots. The reason is simple, if you create more natural or intuitive way of sending task to the robot, their use could potentially be extended also to non-expert user. For example, a master could record him/her self with a camera while performing a certain exercise, the ask to the robot to repeat the same exercise without even write a line of code.

The ways demonstrations are provided to the robot are worth to be analyzed. Among different type of demonstrations, three main categories have been identified, i.e. vision-based, teleoperation based, and by kinesthetic teaching.

In vision-based demonstrations, data acquisition is performed using cameras. Depending on the target, cameras can track the pose of an object (Du et al., 2021; Ten Pas et al., 2017), the object itself (Fang et al., 2020), or a marker (Garrido-Jurado et al., 2014), among other possibilities (I.-C. A. Liu et al., 2022; Savarimuthu et al., 2017). Camera-based tracking is straightforward and does not require robotic equipment for data recording. However, when the goal is to transfer knowledge to a robot, this approach may pose limitations as having access to the robot's state often facilitates the transfer of skills to the robotic system (Eppner et al., 2021).

The teleoperation category includes architectures where a master device sends task commands, and a slave robot executes them, often at a certain distance from the master (Si et al., 2021). Commands can be provided through interfaces such as haptic devices or robot mirroring. While less common than camera-

based demonstrations (Zeestraten et al., 2018), teleoperation offers advantages: (i) direct access to the robot states through transformations between master and slave, and (ii) the ability to operate the robot remotely, which is useful in hazardous environments (Jha & Chiddarwar, 2017; Pervez et al., 2017).

Finally, another common method for providing demonstrations is kinesthetic teaching (Nemec et al., 2014; Samant et al., 2016). This involves physically moving the robot by hand to execute the desired task, recording relevant states such as pose, joint angles, and torques (Nemec et al., 2019). Kinesthetic teaching is widely used because it allows direct interaction with the robot, typically equipped with a gravity compensation controller during the demonstration. This hands-on approach helps the demonstrator understand the robot’s limitations for the target task (Sakr et al., 2020). In guiding applications like rehabilitation, expert therapists can guide the robot to ensure demonstrations align with patient limitations (Braglia et al., 2025; Lauretti et al., 2017). Since the target tasks in this work were predominantly of this nature, kinesthetic teaching was chosen over other techniques.

Although not explicitly mentioned among the main demonstration categories, prompt commands have recently emerged as an additional method for providing demonstrations to robots (X. Wang et al., 2024). With advancements in technologies like Large Language Models (LLMs), it has become increasingly feasible to create libraries of basic robot movements, which can be combined and executed as complex tasks in response to high-level prompts (Barmann et al., 2024; L. Chen et al., 2024). This approach is gaining consideration due to its natural interaction style and high interpretability for humans.

Still, despite its promise, there are concerns about translating high-level commands into low-level robot actions. Misinterpretation of prompts could lead to incorrect task execution, and the variability in how high-level commands are understood raises questions about achieving task precision and accuracy. Nonetheless, this field is progressing rapidly, and it is likely that these challenges will be addressed in the near future.

2.2 Motion Representation and Encoding

Once demonstration data are collected, the next step is to generate robot commands to reproduce the taught task. This process involves processing the data to extrapolate movement commands and translating them into sequences interpretable by the robot. Ideally, the encoding should be compact and adaptable to

more complex robotic systems.

The choice of encoding technique depends on the framework being used. In this section, various encoding frameworks are introduced, categorized by the nature of their motion generators. As suggested in (Khansari-Zadeh & Khatib, 2017), encoding methods can be grouped into potential fields, dynamical systems and, additionally, probabilistic techniques.

2.2.1 Potential Fields

Potential fields represent a widely used approach to describe robot workspaces through a global potential function (Khansari-Zadeh & Khatib, 2017). The gradient of this function inherently guides the robot toward the target point, with the global potential often comprising an attractive component at the target and additional repulsive components to represent obstacles. The interplay of these fields generates a resultant force that dictates the robot’s motion direction. This approach has proven effective across various applications, offering flexibility and robustness in defining movement dynamics.

For instance, in (Kastritsi & Doulgeri, 2022), potential fields are leveraged to restrict a tool’s access to specific Forbidden Regions. Unlike traditional methods that rely on analytical functions, this work utilizes a Point Cloud representation to define the geometry of constraints, highlighting the versatility of potential fields in accommodating complex environments.

The utility of potential fields extends further in (Khansari-Zadeh & Khatib, 2017), where the authors encode data from the teaching phase into a tailored potential function. By designing this function, the robot’s motion can be regulated to reproduce the desired task while also incorporating variable stiffness behaviors. The gradient and curvature of the potential function serve to encode motion and stiffness profiles, respectively. Such an adaptive control policy enhances the robot’s ability to operate in dynamic environments, underscoring the non-geometric advantages of the framework.

Another notable application is found in teleoperation setups, as demonstrated in (Turro & Khatib, 2001). Here, potential fields are employed to create virtual fixture constraints, with attractive forces guiding the master device along a desired path and repulsive fields on the slave side ensuring obstacle avoidance. This concept was later extended in (Selvaggio et al., 2016), where the methodology was augmented with an online virtual fixture generation mechanism. A stereo vision system, trained offline, facilitates real-time detection of object poses, illustrating

the integration of visual data with potential field-based control.

In surgical robotics, potential fields play a critical role in dual-arm teleoperation, as explored in (Ferraguti et al., 2015). Given the precision required in such applications, the authors conduct a rigorous passivity analysis of the proposed architecture. They further refine the potential field approach to shape the resultant forces, ensuring the robot consistently follows the desired path while avoiding configurations that may hinder its performance.

An innovative extension of potential field application is presented in (Asl et al., 2020), where an Assist-as-Needed control algorithm is developed. This framework combines two distinct potential fields: one implements a Velocity Field Control (VFC) to guide a patient along a predefined path with a reference velocity profile, while the other introduces forces outside a defined area around the path to limit positional errors. This dual approach balances assistance with execution flexibility, allowing patients greater autonomy over task timing while maintaining precision.

These diverse applications of potential fields illustrate their adaptability and effectiveness across a range of robotic systems, from path-following and obstacle avoidance to adaptive human-robot interaction frameworks. Their ability to encode complex behaviors and integrate seamlessly with sensory data underscores their significance in advancing robotic control strategies.

2.2.2 Dynamical Systems

Dynamic Movement Primitives (DMP) (Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023) represent a foundational framework for encoding motor movements, introduced in (Ijspeert et al., 2013) as a method with strong stability properties. At their core, DMPs consist of a mass-spring-damper system enhanced by a non-linear forcing term. This addition allows for shaping the attractor landscape to reflect the desired dynamics (Ijspeert et al., 2002), making them a versatile tool for robotic motion generation. Their adaptability has been demonstrated across various applications, including virtual fixture frameworks for rehabilitation (Braglia et al., 2025; Lauretti et al., 2017), human-robot handovers (Dimeas et al., 2019; Sidiropoulos & Doulgeri, 2021), and manufacturing tasks (Kramberger et al., 2018). The flexibility inherent in DMPs stems from their non-geometric nature, which enables adjustments to target constraints by modifying the initial and final positions or by incorporating coupling terms (Ijspeert et al., 2013).

An interesting alternative to DMPs was proposed in (Hersch et al., 2008), where the authors introduced an acceleration model for defining robot trajectories through a second-order system. This model operates via a spring-damper dynamic system modulated by Gaussian Mixture Models (GMM). Unlike DMPs, which handle each dimension of motion separately, this hybrid approach combines task-space and joint-space information to guide robot control. Furthermore, rather than simply providing desired trajectories, the framework in (Hersch et al., 2008) learns the entire dynamical system. This emphasis on modeling the system dynamics leads to a more adaptive controller, offering advantages in dynamic and complex environments.

Noteworthy, DMPs could also be easily extended to the *Potential Field* class if we account for their ability to include coupling terms, which includes also potential terms for obstacle avoidance applications. Example are given by Ginesi, Meli, et al., 2021; Ginesi, Sansonetto, and Fiorini, 2021, where authors proposed a framework for obstacle avoidance based on superquadric potential functions to represent volumes. In this work, we extend our previous work to include the velocity of the system in the definition of the potential. Our formulations guarantee smoother behavior with respect to state-of-the-art point-like methods. Moreover, our new formulation allows obtaining a smoother behavior in proximity of the obstacle than when using a static (i.e. velocity independent) potential.

Indeed, DMPs can be included, to some extent, to the *Potential Field* class by leveraging their ability to incorporate coupling terms, including potential functions for obstacle avoidance. Examples of such extensions are presented in (Ginesi, Meli, et al., 2021; Ginesi, Sansonetto, & Fiorini, 2021), where the authors propose an obstacle avoidance framework based on superquadric potential functions to model volumes. In this work, authors incorporate system velocity into the potential function definition to ensure a smoother behavior compared to state-of-the-art point-like methods, yielding improved obstacle avoidance performance, particularly in close proximity to obstacles, compared to static (i.e., velocity-independent) potential functions. Another example of DMP for obstacle avoidance is provided by (Weitschat & Aschemann, 2018; Weitschat et al., 2016), where authors introduce an online motion generation framework for mirroring human arm motion such that possible collisions are detected in advance.

Another noteworthy contribution to dynamical system formulations is the Stable Estimator of Dynamical Systems (SEDS), introduced in (Khansari-Zadeh & Billard, 2011). SEDS focuses on computing parameters for a dynamical system to encode the information gathered during demonstrations of a target mo-

tion (Khansari-Zadeh & Billard, 2014; Khansari-Zadeh & Billard, 2012). Unlike DMPs, which treat each dimension of motion separately, SEDS employs a single multidimensional model. This unified approach ensures that the learned dynamical system parameters allow for a faithful reproduction of demonstrated tasks while also guaranteeing global asymptotic stability—an assurance absent in the model proposed by (Hersch et al., 2008).

SEDS has been successfully applied to virtual fixture frameworks, as evidenced in (Gopinath et al., 2016), where the authors formalize a human-robot interaction control policy using SEDS. This implementation highlights the robustness of the framework in establishing control dynamics that align closely with human-guided demonstrations, showcasing the potential of SEDS for enhancing robotic collaboration.

Collectively, these approaches illustrate the richness of dynamical system-based motion generation methods. While DMPs provide simplicity and versatility for many applications, the alternative frameworks offered by (Hersch et al., 2008) and (Khansari-Zadeh & Billard, 2011) push the boundaries of adaptability and stability, catering to more intricate task requirements and complex environments. Finally, a broader perspective on dynamical systems, extending beyond DMP and SEDS, is presented in (Calinon & Lee, 2017). The authors provide here a comprehensive overview of learning control, emphasizing the representation of skills to be acquired and exploring various learning strategies that contribute to the development of robust and adaptive controllers.

2.2.3 Probabilistic

Probabilistic constraints in robotics introduce a non-geometric framework where the virtual fixture’s geometry is shaped by a probability distribution. Within the realm of LfD, stochastic frameworks have gained prominence (Calinon, 2009), enabling the integration of multiple demonstrations into a unified model. This approach not only generalizes task representation but also accommodates the variability inherent in the teaching process (Calinon et al., 2007).

For instance, in (Muhlbauer et al., 2024), Gaussian Mixture Models (GMM) are utilized to encode multiple demonstrated trajectories into a probability distribution. The framework employs a Product of Experts approach to fuse position-based and visual servoing fixtures. Gating functions dynamically adjust the influence of each expert, adapting the virtual fixture assistance based on the robot’s interaction with the environment. This dynamic adaptability categorizes

the method as non-geometric, highlighting its responsiveness to environmental changes.

A compelling application of probabilistic methods is illustrated in (Duan et al., 2021), which addresses a robot-assisted craniotomy procedure. The task is divided into two phases: alignment and drilling. During alignment, compliance is emphasized with low-stiffness control, allowing the tool to move freely toward the pre-drilling position. Conversely, drilling requires high-stiffness position control to restrict the robot’s motion along the drilling axis. GMMs, learned from human demonstrations, generate a probability distribution of human forces, effectively mitigating measurement noise and discerning the surgeon’s intention—whether aligning or drilling. This approach enables dynamic modifications to the virtual fixture geometry as the task transitions between subtasks, categorizing the method as non-geometric. The constraints vary between soft (during alignment) and hard (during drilling), with position constrained in alignment and both position and orientation constrained in drilling.

Another notable example is the work of (Raiola, Lamy, & Stulp, 2015), further expanded in (Raiola et al., 2018). The authors encapsulate the geometry of virtual constraints within a GMM, which is updated dynamically based on the robot’s proximity to multiple virtual fixtures. Constraint selection is managed via probabilistic weighting (Raiola, Rodriguez-Ayerbe, et al., 2015). This framework also allows for incremental refinement of virtual guides through additional demonstrations. Newly acquired data is used to compute a GMM, which is then compared to the existing model. If the likelihood of the new GMM surpasses a predefined threshold, a new virtual fixture is created, ensuring that the system evolves with new demonstrations.

These applications underscore the versatility and adaptability of probabilistic frameworks in robotic systems. By embedding variability and learning into the constraints, these methods bridge the gap between rigid, predefined geometries and the dynamic, often unpredictable nature of real-world environments.

2.3 Robot control for Human-Robot Interaction

This work adopts a robot manipulator to evaluate the algorithm proposed throughout the thesis. The type of interaction between the robot and the environment (the human in this case) can be formulated analytically in order to account the manipulator dynamics. In the following, we summarize and introduce key concepts from (Sciavicco & Siciliano, 2012), aiming to clarify specific aspects related

to addressing human-robot interaction (HRI) at the robot's low-level control stage. These foundational concepts provide the theoretical framework necessary to understand how control strategies can be effectively designed to facilitate seamless and responsive interaction between humans and robots.

The equation of motion of an n -DOF robot manipulator with joint angles $\mathbf{q} \in \mathbb{R}^n$ it is usually represented as:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v(\mathbf{q})\dot{\mathbf{q}} + \mathbf{F}_s(\mathbf{q})\text{sgn}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \quad (2.1)$$

where $\mathbf{B}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the positive definite inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ is the matrix taking into account the Coriolis and centrifugal effects, $\mathbf{F}_v \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing the viscous friction coefficients, $\mathbf{F}_s \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing the Coulomb friction coefficients and multiplying the signed joint velocities sgn , finally the term $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$ is the gravity vector. Note that, if the end-effector of the robot is in contact with the environment, one should include and subtract to the right hand side of (2.1) the term $\mathbf{J}(\mathbf{q})\mathbf{w}_e$, denoting with $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ the body Jacobian of the manipulator, and $\mathbf{w}_e \in \mathbb{R}^n$ is the external applied wrench vector.

In order to study the interaction between the manipulator and the environment, we consider the addition of a force component $\mathbf{F}_e \in \mathbb{R}^d$ that measures the contact forces from the environment to the end-effector of the robot, that is

$$\boldsymbol{\tau} = \mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{J}^\top(\mathbf{q})\mathbf{F}_e, \quad (2.2)$$

with

$$\mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v(\mathbf{q})\dot{\mathbf{q}} + \mathbf{F}_s(\mathbf{q})\text{sgn}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}).$$

In (2.2) the change in nomenclature was done to simplify the notation, while the term $\mathbf{J}^\top(\mathbf{q})\mathbf{F}_e$ was added in order to compensate the external contact forces (Sciacicco & Siciliano, 2012).

We begin by considering motion control under inverse kinematics, where the joint accelerations $\ddot{\mathbf{q}}$ in (2.2) are computed as a function of the error $\mathbf{e}$ between the robot's actual pose $\mathbf{y} \in \mathbb{R}^d$ and its desired pose $\boldsymbol{\mu} \in \mathbb{R}^d$. This relationship is expressed as:

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\mathbf{M}^{-1}(\mathbf{M}\ddot{\mathbf{x}} + \mathbf{D}\dot{\mathbf{e}} + \mathbf{K}\mathbf{e} - \mathbf{M}\dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{F}_a), \quad (2.3)$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$, $\mathbf{D} \in \mathbb{R}^{d \times d}$, and $\mathbf{K} \in \mathbb{R}^{d \times d}$ are diagonal positive definite

matrices representing inertia, damping, and stiffness, respectively. The term $\mathbf{F}_a \in \mathbb{R}^d$ denotes the forces actuated by the robot at its end-effector.

Under the assumption of error-free force measurements, (2.3) simplifies to:

$$\mathbf{M}\ddot{\mathbf{e}} + \mathbf{D}\dot{\mathbf{e}} + \mathbf{K}\mathbf{e} = \mathbf{F}_a, \quad (2.4)$$

which defines a mechanical impedance connection. The interaction dynamics are characterized by the matrices of inertia $\mathbf{M}$, damping $\mathbf{D}$, and stiffness $\mathbf{K}$. By appropriately tuning these matrices, the robot's interaction with the environment can be modified, allowing it to behave in a compliant or stiff behavior as desired.

The system described in (2.4) can be interpreted as a controller that generates the output force $\mathbf{F}_a$ based on motion variable inputs $(\mathbf{e}, \dot{\mathbf{e}}, \ddot{\mathbf{e}})$. The selection of the matrices $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ plays a critical role in determining the nature of interaction with the environment. For example, a higher stiffness matrix $\mathbf{K}$ can handle external disturbances arising from imprecisions in the robot model or approximations in the inverse kinematics. However, this could result in overly dominant interaction forces, leading to an aggressive controller behavior that might not be suitable for human-robot interaction (F. J. Abu-Dakka & Saveriano, 2020; Ferraguti et al., 2019).

To address this challenge, the impedance control introduced earlier can be modified by decoupling it from the robot motion control problem. This approach involves defining a compliance direction that characterizes the desired behavior of the manipulator's end effector. The compliance direction is specified in terms of a compliance frame, which is characterized by its position $\mathbf{p}$, rotation matrix $\mathbf{R}$, and the associated linear and angular velocities and accelerations. These quantities are obtained by integrating:

$$\mathbf{M}\ddot{\tilde{\mathbf{e}}} + \mathbf{D}\dot{\tilde{\mathbf{e}}} + \mathbf{K}\tilde{\mathbf{e}} = \mathbf{F}_e, \quad (2.5)$$

where $\tilde{\mathbf{e}}$ represents the error between the desired reference frame and the compliance frame. The parameters $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ denote the impedance properties between the desired and compliance frames, and $\mathbf{F}_e$ is the vector of measured forces.

This formulation allows for a clear separation of responsibilities: the motion controller gains can be designed to handle disturbance rejection, while the impedance controller gains can be tuned to achieve a desired interaction behavior with the environment. Unlike the previous case, this approach is referred to as admittance control, as the controller generates reference trajectories for the

motion controller based on the force measurements $\mathbf{F}_e$.

2.4 Proposed framework

The intention of this work is to study and evaluate a LfD framework that involves human in the final control loop. The human can be in contact with the robot (see Chapter 5- 6) or not (see Chapter 7) but always sharing the same workspace in a collaborative environment.

The presented architecture always consists of three important steps, i.e. providing demonstrations, encoding the task, and establishing a proper robot controller.

For what concerns demonstrations, kinesthetic teaching was used to collect data and the end effector pose trajectory was recorded. This step is further explained and detailed in Chapter 3, where a novel and validated way to extrapolate consistent skill from demonstrations is introduced.

Encoding techniques are then applied once the desired task is extracted. In this work, the encoding step is used to (i) obtain a compact building block able to synthetically contain the sufficient information to reproduce the desired task, and (ii) to send to the robot desired pose trajectories that it has to track over the task execution. In particular, as the human is involved at the time of execution, the encoding block should respond to the human feedback adapting the output trajectory accordingly.

Finally, this work deals with guiding task, that is applications where the robot moves over a fixed constrain curve in its workspace. Apart from applications where no contact between human and robot occurs, this involves some considerations to comply with the user execution. In this case we distinguish between the nature of the encoding block and the one of the robot low level controller.

As it will be shown in the next chapters, the encoding block provides a desired reference to the robot in response to human feedback. Depending on the type of feedback, the reference can be updated as a result of an admittance law (Chapter 5), an optimization problem (Chapter 6) or an user-defined algorithm (Chapter 7). Instead, for what concerns the low level robot controller, when there is contact between the human and the robot it was decided to adopt an impedance type of controller. Therefore, the robot desired trajectories are always updated as a consequence of the encoding block output which moves the working point of the impedance controller. This establishes to each working point a tunable stiffness and damping profile that allows to account for human displacements with

respect to the target task, ensuring that the robot's task is always restored and the human execution feels the forces pushing towards the intended robot guide.

The primary goal of this work is to explore and evaluate a LfD framework that actively involves the human in the final control loop. The human may physically interact with the robot, as described in Chapters 5 and 6, or simply share the same workspace in a collaborative setup, as in Chapter 7. Regardless of the scenario, the interaction is always designed to account for a cooperative environment.

The proposed framework always adopts three fundamental steps: providing demonstrations, encoding the task, and designing an appropriate robot controller.

The first step, demonstration, employs kinesthetic teaching to collect data. Specifically, the end-effector pose trajectory is recorded as the robot is manually guided through the task. This process is elaborated in Chapter 3, where a novel methodology is introduced to reliably extract consistent skills from demonstrations. This ensures that the captured data serves as a robust foundation for subsequent steps.

Once the desired task is extracted, encoding techniques are applied to synthesize this information. The encoding step serves a dual purpose: (i) to create a compact representation that efficiently encapsulates the necessary information for reproducing the task, and (ii) to provide the robot with desired pose trajectories for task execution. A key aspect of this work is the responsiveness of the encoding block to human feedback during execution, allowing the generated trajectories to adapt dynamically.

The focus of this research lies in guiding tasks, where the robot navigates along a fixed constraint curve within its workspace. In scenarios involving human-robot contact, particular attention is given to ensuring compatibility with the user's input and execution. The framework distinguishes between the nature of the encoding block and the robot's low-level controller.

As demonstrated in subsequent chapters, the encoding block generates a reference trajectory for the robot based on human feedback. This reference is adjusted according to an admittance law (Chapter 5), an optimization problem (Chapter 6), or a user-defined algorithm (Chapter 7). On the other hand, the low-level controller operates differently depending on whether there is physical interaction. When there is no contact, a simple position controller was used. Instead, for cases involving human-robot contact, an impedance controller is employed. The latter ensures that the robot's desired trajectories are updated dynamically in response to the output of the encoding block. By shifting the working point of the impedance controller, the framework assigns a stiffness and damping pro-

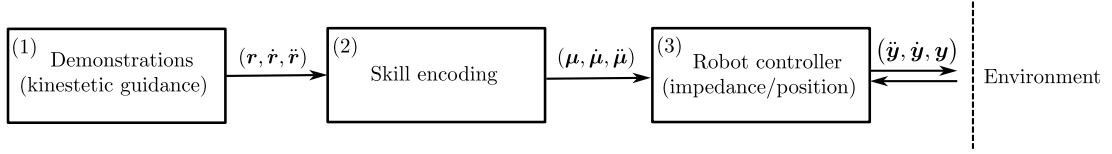


Figure 2.2: LfD steps of the proposed framework.

$(\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}})$: recorded demonstrated trajectories $(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}, \ddot{\boldsymbol{\mu}})$: desired reference trajectories $(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}})$: actual robot trajectories s : phase variable L : length of a curve T : task duration T_s : sampling time $m/\mathbf{M}$: inertia parameter/matrix $d/\mathbf{D}$: damping parameter/matrix $k/\mathbf{K}$: stiffness parameter/matrix

Table 2.1: Adopted notation during the thesis.

file to each working point. This configuration accommodates human deviations from the target task while preserving the robot's primary objective. Additionally, it ensures that the human experiences forces guidance toward the intended robot trajectory, achieving a homogeneous balance between assistance and task restoration. Figure 2.2 provides a visual representation of the three core steps discussed earlier: demonstration, encoding, and robot control. Additionally, Table 2.1 presents a summary of the nomenclature that will be consistently used throughout this work. While new symbols and variables may be introduced in the context of specific frameworks, certain notations will remain fixed to maintain clarity and ensure consistency.

Demonstrations alignment for consistent skill extrapolation

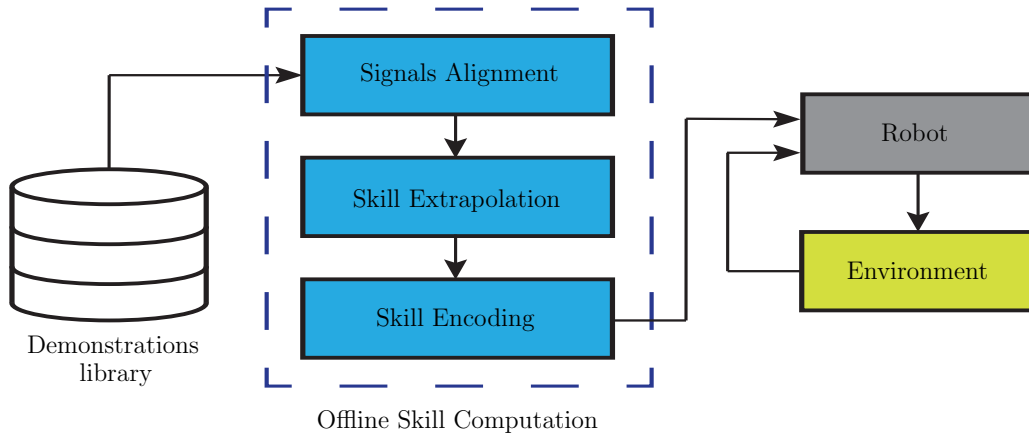


Figure 3.1: Learning from Demonstration (LfD) framework.

Robot Learning from Demonstration (LfD) teaches robots tasks via a few demonstrations (Calinon et al., 2010; Pastor et al., 2009). Specifically, LfD seeks to facilitate intuitive skill transfer, reduce the need for extensive data, and make robots more approachable to individuals without programming expertise (Pastor et al., 2009). LfD architectures, regardless of the target application, typically follow the workflow shown in Fig. 3.1. Initially, a set of demonstrations is analyzed to extract the most informative content, often referred as *skill* in literature (Calinon, 2009; Gaspar et al., 2018). This extracted skill is then sent to the robot, which is equipped with a controller that allows it to interact with its environment and adapt its behavior to cope with internal or external constraints (Bianco & Ghilardelli, 2017).

When computing offline skills, demonstrations are often pre-processed prior to sending commands to the robot. A critical step involves aligning the collected signals, typically achieved by defining a similarity or variance metric within the

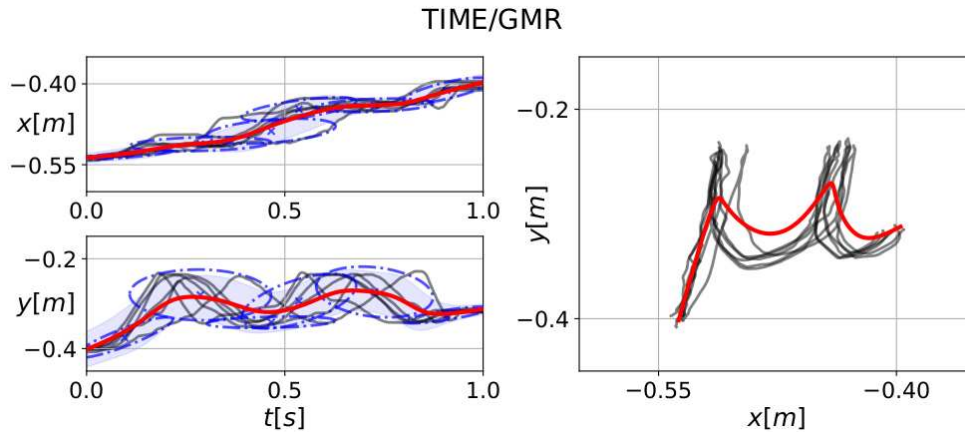


Figure 3.2: Extrapolating a skill through regression (red) of a GMM model (blue), trained over recorded data (black) with no previous alignment.

demonstration set to extract the skill. In robotics, Gaussian Mixture Models (GMM) (Calinon et al., 2007) are commonly used to model the dataset of demonstrations, followed by Gaussian Mixture Regression (GMR) to retrieve the desired skill. While this method is discussed in detail in Chapter 4, we will use the GMM algorithm here as a benchmark for evaluating the quality of the retrieved skill.

In this context of signals alignment, timing frequently introduces distortions, as temporal discrepancies can compromise the accuracy of comparing time series data (Braglia, Tebaldi, et al., 2024; Meattini et al., 2022; Su et al., 2021). An example is shown in Fig. 3.2, where it can be observed that the extrapolated skill (in red) in the 2D plot does not qualitatively reflect the provided demonstrations.

In signal processing, Dynamic Time Warping (DTW) is a simple yet robust and effective technique for achieving this (Muller, 2007; Senin, 2008). In the basic case of comparing two-time series, alignment refers to establishing a point-by-point correspondence between the signals, minimizing a chosen variance metric. For DTW, this metric is the Euclidean distance between points, and as such, DTW computes a distance matrix that allows tracing back an optimal alignment path between the two time series (Chiu et al., 2004).

3.1 Dynamic time warping

DTW is a signal processing technique or, more properly, a similarity measure used to find the optimal time alignment between two sequences of different duration (Chiu et al., 2004; Muller, 2007). It provides information about the correspondence of separate sequences, making a one-by-one comparison between their

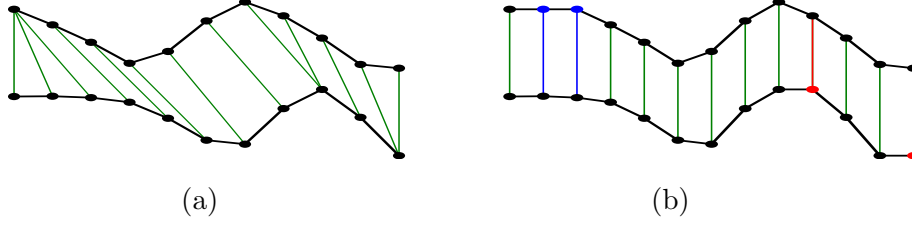


Figure 3.3: Aligning two time series with DTW.

samples. DTW is a well-known and widely used technique in pattern recognition, and it was used in diverse applications such as gesture recognition, robotics, and speech processing, among others (Senin, 2008). Given two generic time series $\mathbf{x} \in \mathbb{R}^{d \times N}$ and $\mathbf{y} \in \mathbb{R}^{d \times M}$ of same dimensionality d , DTW is defined as (Vayer et al., 2020):

$$DTW(\mathbf{x}, \mathbf{y}) = \min_{\pi \in \mathcal{A}(\mathbf{x}, \mathbf{y})} \sum_{(i,k) \in \pi} d(\mathbf{x}_i, \mathbf{y}_k), \quad (3.1)$$

where $\mathcal{A}(\mathbf{x}, \mathbf{y})$ is the set of all possible alignments π between $\mathbf{x}$ and $\mathbf{y}$, while $d(\mathbf{x}_i, \mathbf{y}_k) = \|\mathbf{x}_i - \mathbf{y}_k\|^2$ is the squared Euclidean distance.

The DTW algorithm is summarized in the Algorithm 1. For each element of the cost matrix $\mathbf{J}_{i,k}$, the distance between the points $\mathbf{x}_i$ and $\mathbf{y}_k$ is calculated; this term is then added to the minimum between the neighbors of $\mathbf{J}_{i,k}$. Depending on which neighbour has the lowest value, the algorithm characterizes this step as a *match*, an *insertion*, or a *deletion*. If a match occurs, the points $(\mathbf{x}_i, \mathbf{y}_k)$ are correspondent; if an insertion occurs one of the two points has to be replicated to enhance correspondence; lastly, deletion occurs if the correspondence between $\mathbf{x}$ and $\mathbf{y}$ is increased by removing either $\mathbf{x}_i$ or $\mathbf{y}_k$. This is shown in Fig. 3.3. After establishing a correspondence between the points in the two sequences in black (see Fig. 3.3(a), the alignment path allows padding the points where an insertion or deletion occurred (see Fig. 3.3(b)) in order to synchronize the two signals.

DTW tracks whether $\mathbf{J}$ values come from match, insertion, or deletion. This procedure facilitates the extraction of the alignment path once the cost matrix is filled. The alignment path π^* represents the minimum cost path through the cost matrix; indeed, as expressed by (3.1), $\pi^* \in \mathcal{A}(\mathbf{x}, \mathbf{y})$ minimizes the cumulative distance between $\mathbf{x}$ and $\mathbf{y}$. Note that DTW always enforces the first and last points of the two-time series to match (Muller, 2007). Finally, by looking at π^* , one can modify the two sequences to align them optimally. Figure 3.4 shows an example of how the minimum cost path π^* can be reconstructed from the cost matrix $\mathbf{J}$. Each of the computed cost is mapped with a color code which is

Algorithm 1 Dynamic Time Warping (DTW)**Input:** $\mathbf{x} \in \mathbb{R}^{d \times N}$, $\mathbf{y} \in \mathbb{R}^{d \times M}$ **Output:** cost matrix $\mathbf{J} \in \mathbb{R}^{N \times M}$, alignment path $\pi^* \in \mathbb{R}^{p \times 2}$ *Initialization:*1: $\mathbf{J}_{i,0} = \infty$, $i = 1, 2, \dots, N - 1$ 2: $\mathbf{J}_{0,k} = \infty$, $k = 1, 2, \dots, M - 1$ *Compute $\mathbf{J}$:*3: **for** $i = 1$ to $N - 1$ **do**4: **for** $k = 1$ to $M - 1$ **do**

5:

$$\mathbf{J}_{i,k} = d(\mathbf{x}_i, \mathbf{y}_k) + \min \begin{cases} \mathbf{J}_{i-1,k-1} & (\text{Match}) \\ \mathbf{J}_{i-1,k} & (\text{Insertion}) \\ \mathbf{J}_{i,k-1} & (\text{Deletion}) \end{cases}$$

6: **end for**7: **end for** *Compute alignment path π^* :*8: Trace back the warping path from $\mathbf{J}_{N-1,M-1}$ to $\mathbf{J}_{0,0}$ minimizing (3.1); π^* stores the (i, k) indexes of such path.9: **return** $\mathbf{J}$, π^*

greener for lower distance values. In the figure, π^* has been tracked in red and its relative cost, which minimizes (3.1), is 0.0302.

Note that white areas of $\mathbf{J}$ are the one where no cost has been calculated. Depending on the similarity between signals, one can decide whether the cost computation window is larger (until filling all the matrix) or thinner. As a matter of fact, it is not necessary to compute all the values of $\mathbf{J}$ as (i) it requires a higher computational burden and (ii) it can be demonstrated that π^* always lies in proximity of the diagonal of $\mathbf{J}$ (Muller, 2007; Senin, 2008).

3.1.1 Limitations of DTW

Techniques like DTW measure similarity regardless of timing, but DTW does not fully decouple time, which is crucial in many robotics applications (Muller, 2007; Senin, 2008). For instance, one may want to adjust robot trajectories afterward to account for geometric, kinematic, or external constraints, such as in guiding applications (Braglia, Onfiani, et al., 2024; Braglia et al., 2025). In these applications, the robot must ensure a virtual geometric constraint, allowing the user in contact with the robot to freely move along it (Bowyer et al., 2013). Consequently, the virtual guide should impose only a geometric constraint, independent of time.

In LfD, users typically provide multiple demonstrations that require alignment

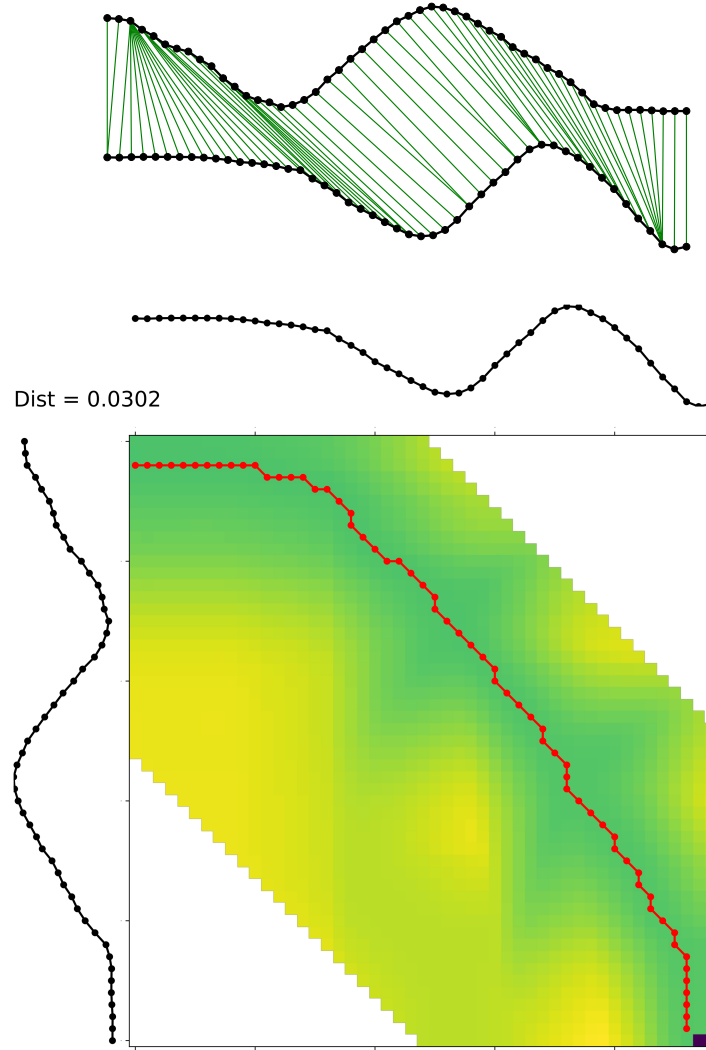


Figure 3.4: A visualization of the cost matrix J for the computation of the alignment path π^* in red.

for consistent skill extrapolation. Specifically, in guiding applications, multiple demonstrations are used to extrapolate the corresponding constraint path and generate the virtual guide. This problem can be divided into two main steps:

1. align recorded trajectories in the dataset to account for time discrepancies in demonstrations;
2. extract a time-agnostic geometric curve to impose the desired geometric constraint.

Since demonstration recordings are typically time-domain based, DTW struggles with step 2, as it inherently preserves the time information from the reference signal used to align other trajectories in the dataset. Consequently, the choice of

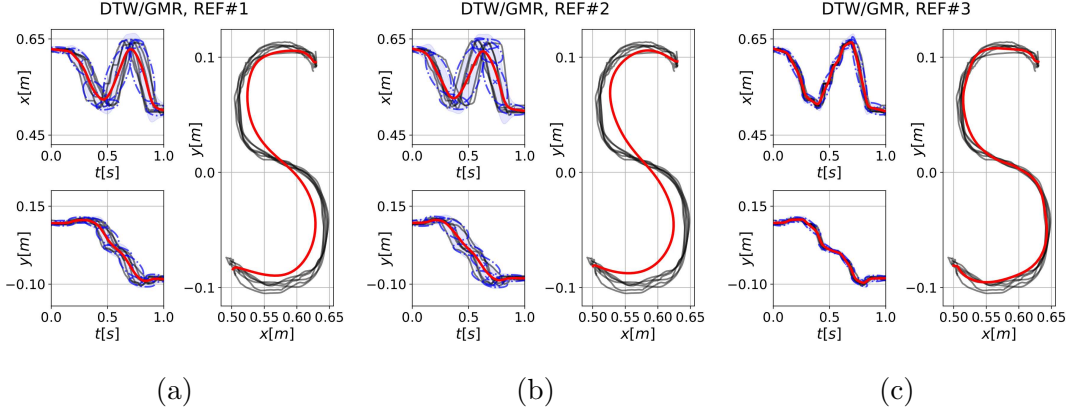


Figure 3.5: Skill computation results when changing the DTW alignment reference.

reference impacts the outcome, leading to variability in results. This limitation can also affect step 1, as different references may produce differing skill outcomes. This effect is illustrated in Fig. 3.5, where the extrapolated skill varies depending on the selected reference.

3.2 The Spatial Sampling algorithm

As shown in (Braglia, Tebaldi, et al., 2024), the issues inherent to the DTW algorithm can be resolved by shifting from time to arc-length parametrization. This change in the representation domain filters out the time information embedded in the demonstration recordings. Furthermore, it improves the synchronization of the demonstration dataset, effectively reducing noise caused by time discrepancies. To understand this approach, it is first necessary to introduce the concept of arc-length parametrization of a curve (for a detailed explanation please refer to Appendix A).

3.2.1 From time to arc length domain

As discussed earlier, time misalignment in demonstrations can introduce noise in skill extrapolation. While techniques like DTW can mitigate this issue, they come with certain limitations. To address this, additional data manipulation is required. From the differential geometry of curves (Toponogov, 2006), it is crucial to first distinguish between the concepts of position trajectory and path (Sciavicco & Siciliano, 2012).

Consider the robot’s workspace, where $\mu_t \in \mathbb{R}^3$, $t \in [0, T]$, represents a temporal sequence of values that the robot’s end-effector should track, with T being the

total movement duration. Ensuring that the robot starts at $\boldsymbol{\mu}_0$ and reaches the final position $\boldsymbol{\mu}_T$ in T seconds inherently provides information about its derivatives, such as $\dot{\boldsymbol{\mu}}_t$ and $\ddot{\boldsymbol{\mu}}_t$. Whenever a temporal sequence of points is defined, it is referred to as a position trajectory. Such a trajectory encodes spatial information (the position) and temporal information (the velocity) at each instant t .

The same sequence of points can be re-parametrized differently. Let us consider a point $\mathbf{p} \in \mathbb{R}^3$ and a smooth, regular function $\boldsymbol{\mu}(s)$ defined over the interval $[s_i, s_f] \in \mathbb{R}$. Considering the relation $\mathbf{p} = \boldsymbol{\mu}(s)$, the group of values of $\mathbf{p}$ as s varies is referred to as a curve or path in space. From Appendix A.3, the scalar parameter s is monotonic in $[s_i, s_f]$ and defines a direction of travel along the curve. For a generic point $\mathbf{p}$ and a fixed point $\mathbf{p}_i$ on the path $\boldsymbol{\mu}(s)$, the arc-length parameter s is defined as the length of the arc $\mathbf{p}\mathbf{p}_i$. The value of s is positive if $\mathbf{p}$ follows $\mathbf{p}_i$ and negative if $\mathbf{p}$ precedes $\mathbf{p}_i$. The point $\mathbf{p}_i$ is set as the curve's origin, with $s = 0$ associated to it. Finally, it is important to note that the monotonic nature of s , similar to the time parametrization t , helps prevent incorrect handling of self-intersecting curves by defining a consistent direction of travel along the trajectory. This property was further demonstrated in (Braglia, Tebaldi, et al., 2024), where the arc-length parametrization was extensively tested on a diverse set of curves from the novel Autolab Co-Manipulation dataset¹.

Being both t and s monotonic, one can easily find an invertible mapping $\gamma : t \in [0, T] \rightarrow [0, L]$, with L being the total length of the curve $\boldsymbol{\mu}(s)$, for which it holds

$$\gamma(t) = s \text{ and } t = \gamma^{-1}(s). \quad (3.2)$$

Consequently, it is possible to write the derivatives of $\boldsymbol{\mu}_t$ in term of the new arc length parametrization. Note that, from now on, the time dependence will be written as a subscript. For instance, accounting for velocity and acceleration, we have

$$\begin{aligned} \dot{\boldsymbol{\mu}}_t &= \boldsymbol{\mu}'(s_t) \dot{s}_t \text{ and} \\ \ddot{\boldsymbol{\mu}}_t &= \boldsymbol{\mu}'(s_t) \ddot{s}_t + \boldsymbol{\mu}''(s_t) \dot{s}_t^2, \end{aligned} \quad (3.3)$$

where the dot notation stands for the derivative w.r.t. time, i.e. $\frac{d(\cdot)}{dt}$, while the prime stands for the derivative w.r.t. the arc length parameter, i.e. $\frac{d(\cdot)}{ds}$.

To address the synchronization of recorded trajectories, improvements can be achieved by transitioning from the time domain to the arc-length domain, as illustrated in Fig. 3.6. In Fig. 3.6(a), 2D trajectories recorded from demonstrations

¹AutoLab Co-Manipulation Dataset: <https://github.com/AutoLabModena/AutoLab-Co-Manipulation-Dataset.git>

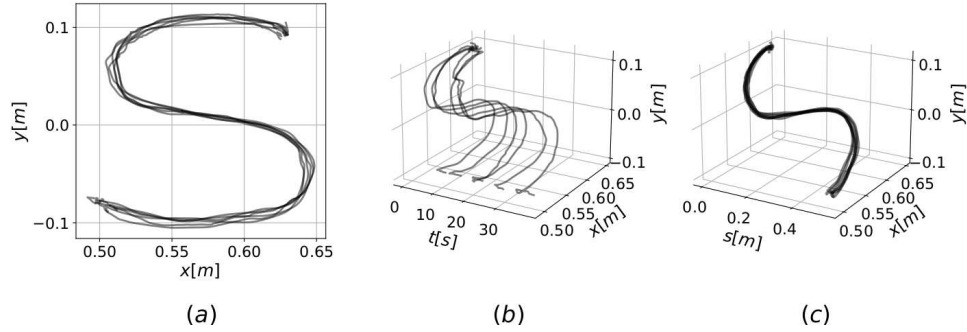


Figure 3.6: Difference between curve representation in time and arc length domain. (a) trajectories recordings on xy plane; (b) trajectories xy plot as a function of time t and (c) as a function of the arc length parameter s .

are shown, while Fig. 3.6(b)-(c) presents the same demonstrations represented in the time and arc length domains, respectively. The arc length representation clearly achieves better synchronization, at least qualitatively.

The intuition is that, since GMMs are typically parametrized in time, one can compare the variance obtained across data when using GMMs with curves represented in the arc-length domain, thereby analyzing their synchronization level. This requires the development of a filter that processes the demonstrated trajectories to remove all time-related information, retaining only the geometric data to provide the corresponding paths. This can be achieved using the Spatial Sampling algorithm, which is detailed in the next section.

3.2.2 Sampling in space

The proposed Spatial Sampling (SS) algorithm generates the sequence $\hat{\mathbf{r}}_k$ of points belonging to the new filtered curve $\hat{\mathbf{r}}(s_k)$, which is a function of the arc-length parameter s_k , starting from the sequence of points $\mathbf{r}_i = \mathbf{r}(iT_s)$, $i = 0, \dots, n$ of the demonstrated trajectory sampled with constant sampling-time T_s . The proposed algorithm works as follows:

1) Given the samples $\mathbf{r}_i$, $i = 0, \dots, n$, a linearly interpolating continuous-time function $\mathbf{r}_{L,t}$ is built. If the sampling period T_s is small enough, then the following holds true:

$$\mathbf{r}_{L,t} \approx \mathbf{r}_t. \quad (3.4)$$

2) A new sequence $\hat{\mathbf{r}}_k$ is created by imposing that $\hat{\mathbf{r}}_0 = \mathbf{r}_{L,0}$ and $\hat{\mathbf{r}}_k = \mathbf{r}_{L,t_k}$ for

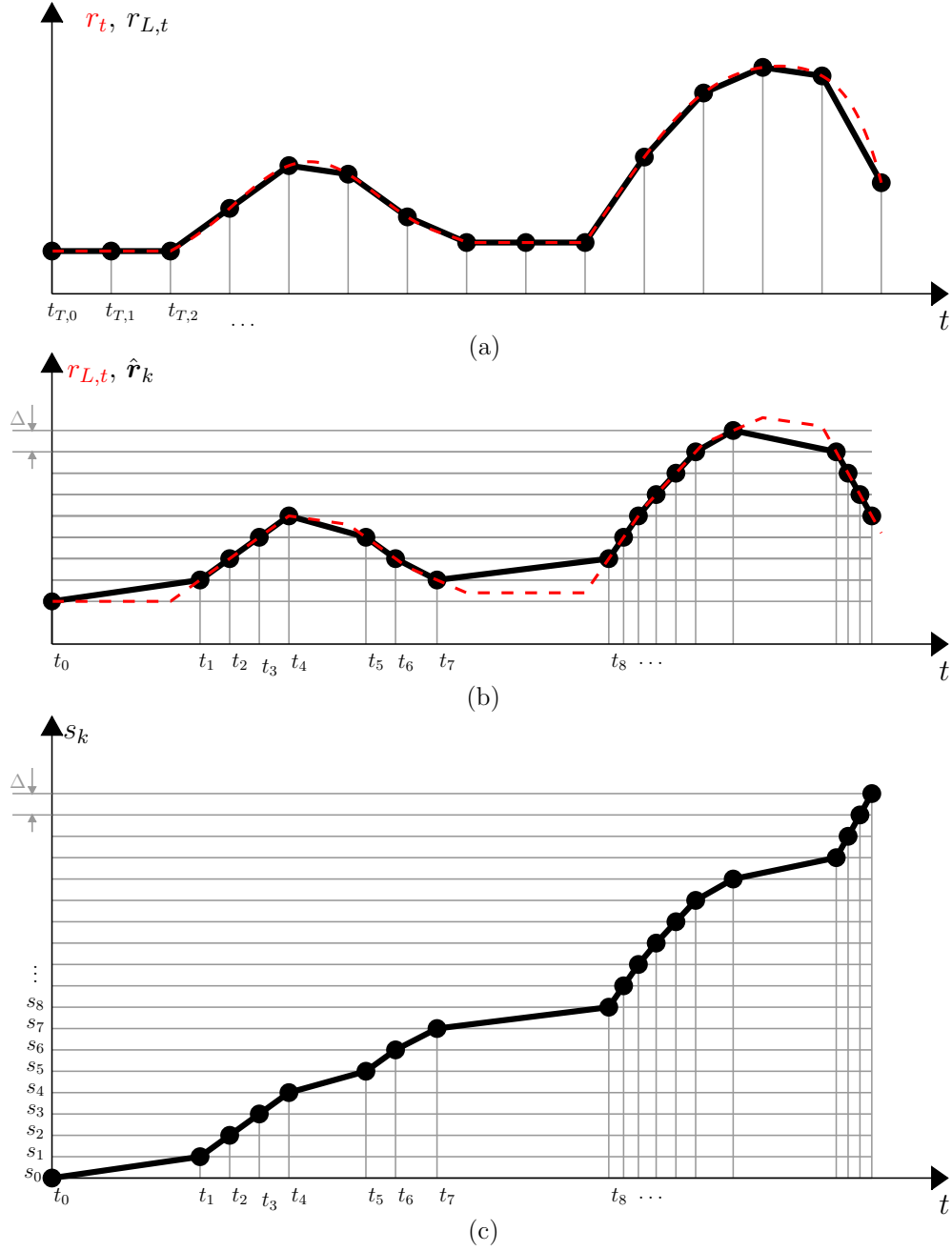


Figure 3.7: Sketch displaying the operation of the SS algorithm in a one-dimensional scenario. (a) Reference trajectory y_t sampled with constant period T and interpolating linear curve $y_{L,t}$. (b) Sequence of spatially sampled points $\hat{y}_k = Y_{L,t_k}$ with $\Delta = \tilde{\Delta}$. (c) Samples s_k of the function $s = \gamma(t)$ when using $\Delta = \tilde{\Delta}$.

$k > 0$, where t_k is the time instant that guarantees the following condition:

$$\|\hat{\mathbf{r}}_k - \hat{\mathbf{r}}_{k-1}\| = \Delta, \quad \text{for } k = 1, \dots, m. \quad (3.5)$$

The parameter Δ defines the geometric distance between consecutive samples of

the filtered trajectory and can be freely chosen. The condition in (3.5) implies that the total distance between the first point $\hat{\mathbf{r}}_0$ and the generic k -th point $\hat{\mathbf{r}}_k$ is given by $k\Delta$, which approximates the length of the curve $\mathbf{r}_{L,t}$ at the time instant t_k with a precision that increases as Δ decreases. Therefore, for Δ sufficiently small, the SS algorithm introduces the following mapping between the length $s_k = k\Delta$ and the position along the approximating linear curve $\mathbf{r}_L$:

$$\hat{\mathbf{r}}_k = \mathbf{r}_{L,t_k}, \quad \text{with} \quad t_k = \gamma^{-1}(s_k), \quad (3.6)$$

where $s_t = \gamma(t)$ is the particular timing law imposed during the trajectory demonstration, describing how the robot moves along the imposed geometric path since the variable s is the arc-length parameterization of the curve. From (3.4), (3.5) and (3.6), the resulting trajectory parameterized with respect to the arc-length parameter $\boldsymbol{\mu}(s) \approx \hat{\mathbf{r}}_L(s) = \mathbf{r}_L(\gamma^{-1}(s))$ can be obtained. The sequence $\hat{\mathbf{r}}_k$ can be viewed as the result of a sampling operation with a constant spatial period Δ , characterized by:

$$\left\| \frac{d\boldsymbol{\mu}(s)}{ds} \right\|_{s=s_k} \approx \frac{\|\hat{\mathbf{r}}_{k+1} - \hat{\mathbf{r}}_k\|}{\|s_{k+1} - s_k\|} = \frac{\Delta}{\Delta} = 1. \quad (3.7)$$

Consequently, the derivative in (3.7) will always be different from zero, and the tangential direction of the curve will always be well-defined. The operation of the spatial sampling algorithm is graphically shown in the case study of Fig. 3.7. The reference trajectory $\mathbf{r}_t$ is recorded with a sampling time T_s . In Fig. 3.7(a) the approximating linear function $y_{L,t}$ is shown, from which it can be seen that the original geometric distance between two consecutive samples y_i is not constant. Subsequently, in Fig. 3.7(b), the spatial sampling algorithm is applied to the curve $y_{L,t}$ using a certain spatial interval $\Delta = \tilde{\Delta}$, where it can be noticed that a more accurate approximation of the original curve can be achieved for smaller parameters Δ . Finally, Fig. 3.7(c) shows the samples s_k of the function $s = \gamma(t)$ introduced in (3.6). Fig. 3.7(a) shows that the proposed spatial sampling imposes no constraints on the demonstrated trajectory $\mathbf{r}_t$, which can also include parts with zero speed, which might occur if the user stops during the demonstration. This is an important feature since no segmentation of the demonstrated trajectory is required, as it is instead by alternative approaches (Cifuentes et al., 2013; Gaspar et al., 2018).

The pseudo-code of the Spatial Sampling algorithm is shown in Algorithm 2. Note that the SS maintains the same Match-Insertion-Deletion architecture as

Algorithm 2 Spatial Sampling (SS)

Input: $\mathbf{r} \in \mathbb{R}^{(N \times d)}$, $\mathbf{t} \in \mathbb{R}^{(N \times 1)}$, Δ
Output: $\mathbf{t}_\Delta \in \mathbb{R}^{(M \times 1)}$, $\mathbf{r}_\Delta \in \mathbb{R}^{(M \times d)}$, $\mathbf{s}_\Delta \in \mathbb{R}^{(M \times 1)}$ *Initialization:*

- 1: Initialize arrays: $\mathbf{t}_{\Delta,0}=0$, $\mathbf{r}_{\Delta,0}=\mathbf{r}_0$, $\mathbf{s}_{\Delta,0}=0$
- 2: Initialize current state: $\mathbf{r}_c=\mathbf{r}_0$, $i=0$
- Compute $\mathbf{t}_\Delta$, $\mathbf{r}_\Delta$, $\mathbf{s}_\Delta$:*
- 3: **while** $i < N$ **do**
- 4: **if** $\|\mathbf{r}_{i+1} - \mathbf{r}_c\| = \Delta$ **then** $\{(Match)\}$
- 5: Update arrays: $\mathbf{t}_\Delta = [\mathbf{t}_\Delta, t_{i+1}]$, $\mathbf{r}_\Delta = [\mathbf{r}_\Delta, \mathbf{r}_{i+1}]$, $\mathbf{s}_\Delta = [\mathbf{s}_\Delta, \mathbf{s}_\Delta(end) + \Delta]$
- 6: Update current state: $\mathbf{r}_c = \mathbf{r}_\Delta(end)$, $i = i + 1$
- 7: **else if** $\|\mathbf{r}_{i+1} - \mathbf{r}_c\| > \Delta$ **then** $\{(Insertion)\}$
- 8: Compute $\hat{\mathbf{r}}_k$ belonging to the segment $\overline{\mathbf{r}_i \mathbf{r}_{i+1}}$ and such that $\|\hat{\mathbf{r}}_k - \mathbf{r}_c\| = \Delta$
- 9: Compute corresponding t_k as in (3.6)
- 10: Update arrays: $\mathbf{t}_\Delta = [\mathbf{t}_\Delta, t_k]$, $\mathbf{r}_\Delta = [\mathbf{r}_\Delta, \hat{\mathbf{r}}_k]$, $\mathbf{s}_\Delta = [\mathbf{s}_\Delta, \mathbf{s}_\Delta(end) + \Delta]$
- 11: Update current state: $\mathbf{r}_c = \mathbf{r}_\Delta(end)$
- 12: **else if** $\|\mathbf{r}_{i+1} - \mathbf{r}_c\| < \Delta$ **then** $\{(Deletion)\}$
- 13: $i = i + 1$
- 14: **end if**
- 15: **end while**
- 16: **return** $\mathbf{t}_\Delta$, $\mathbf{r}_\Delta$, $\mathbf{s}_\Delta$

the DTW in Algorithm 1. Specifically, a match occurs every-time the distance between the current sample $\mathbf{y}_c$ and its consecutive one $\mathbf{r}_{i+1}$ equals Δ ; an insertion occurs every time their distance is greater than Δ and, finally, deletion occurs whenever the distance is lower than Δ , for which the sample $\mathbf{r}_{i+1}$ is removed.

3.2.3 Arc length and its analogies

Ensuring condition (3.7), as established in Appendix A.1-A.2, guarantees that the output curve of the SS algorithm remains regular, meaning that

$$\frac{d\boldsymbol{\mu}(s)}{ds} \neq \mathbf{0}, \quad \forall s \in [0, \mathcal{L}],$$

where $\mathcal{L}$ is the total length of the curve (Toponogov, 2006).

In robotics, arc-length parametrization is widely utilized, particularly for defining curvilinear coordinates or the Frenet-Serret frame (Sciavicco & Siciliano, 2012). As illustrated in Fig. A.2, each point on a regular curve is associated with three unit vectors: the tangent $\vec{\tau}$, the normal $\vec{n}$, and the binormal $\vec{\beta} = \vec{\tau} \times \vec{n}$. These parameters are crucial for various applications, including time-optimal path parametrization (Reynoso-Mora et al., 2013; Verschuer et al., 2009) and robotic

guidance (Raiola, Lamy, & Stulp, 2015; Selvaggio et al., 2018).

This thesis explores how these parameters enhance robotic guidance by aiding the projection of human forces along task trajectories (Chapter 5) and by providing an analytical framework for identifying Euclidean Distance Singularities (Chapter 6).

While arc-length parametrization is theoretically advantageous, some curve-specific calculations are more conveniently performed using their native parameters. Nevertheless, general formulas can be derived, often requiring the Frenet-Serret equations in their arc-length form. The SS algorithm provides a robust method for converting time-based trajectories into their arc-length counterparts, ensuring consistency in this transformation.

Once this parametrization is obtained, basis function approximation (explored in Chapter 4.1) can be applied to derive an analytical representation of the curve $\boldsymbol{\mu}(s)$. From this, the Frenet-Serret frame can be easily computed as:

$$\vec{\tau} = \boldsymbol{\mu}'(s), \quad \vec{n} = \frac{\boldsymbol{\mu}''(s)}{\|\boldsymbol{\mu}''\|}, \quad \vec{\beta} = \vec{\tau} \times \vec{n},$$

where the prime notation denotes differentiation with respect to arc-length s .

3.2.4 Optimization of Parameter Δ

The function $\hat{\mathbf{r}}(s)$ is an approximation of the original trajectory that depends on parameter Δ . The error between the original trajectory $\mathbf{r}_t$ and the filtered trajectory $\hat{\mathbf{r}}(s)$ can be evaluated using the well-known Hausdorff distance d_H (Donoso et al., 2008) as a metric:

$$d_H = \max \left\{ \sup_{\mathbf{r}_i \in S_r} d(\mathbf{r}_i, S_{\hat{\mathbf{r}}}), \sup_{\hat{\mathbf{r}}_k \in S_{\hat{\mathbf{r}}}} d(S_r, \hat{\mathbf{r}}_k) \right\}, \quad (3.8)$$

where $S_{\hat{\mathbf{r}}}$ denotes the set of all the samples $\hat{\mathbf{r}}_k$ of the filtered trajectory $\hat{\mathbf{r}}(s)$, S_r denotes the set of all the samples $\mathbf{r}_i$ of the original trajectory $\mathbf{r}_t$, and

$$d(\mathbf{r}_i, S_{\hat{\mathbf{r}}}) = \inf_{\hat{\mathbf{r}}_k \in S_{\hat{\mathbf{r}}}} d(\mathbf{r}_i, \hat{\mathbf{r}}_k),$$

$$d(S_r, \hat{\mathbf{r}}_k) = \inf_{\mathbf{r}_i \in S_r} d(\mathbf{r}_i, \hat{\mathbf{r}}_k).$$

The choice of parameter Δ in the SS algorithm directly affects d_H in (3.8): $d_H = d_H(\Delta)$. Therefore, it is of interest to set up the following optimization problem.

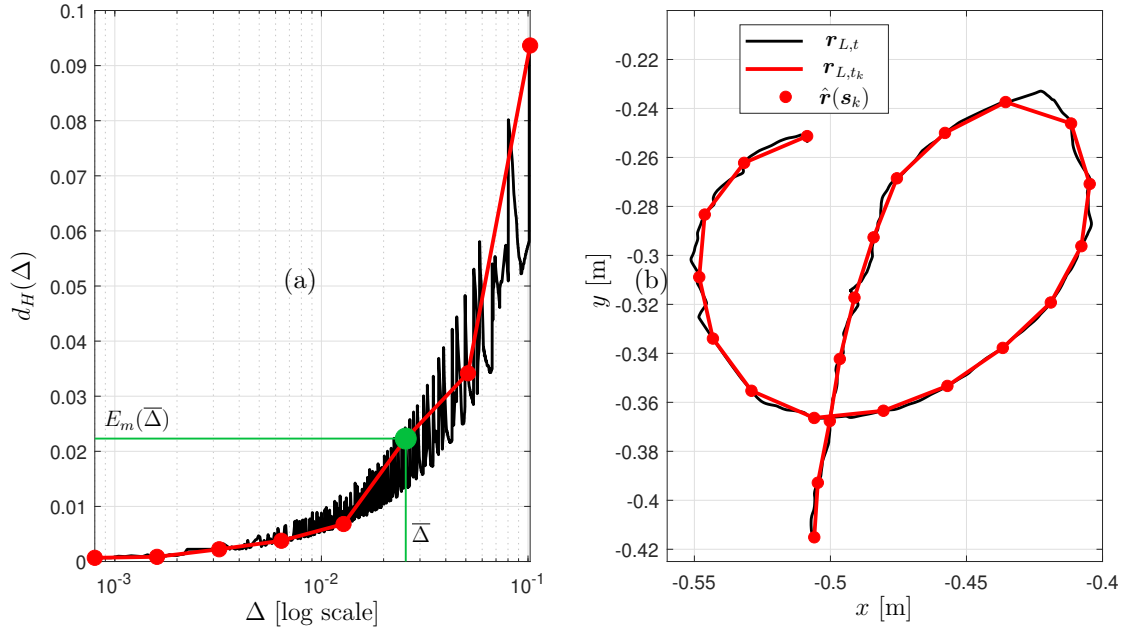


Figure 3.8: Numerical example of Sec. 3.2.4. (a) Result of the Optimization Problem 1 applied to the trajectory shown in (b) using $d_H^* = 0.025$ and $d_{H,t} = 0.1 \cdot 10^{-3}$. (b) Linear interpolation $\mathbf{r}_{L,t}$ of the samples of the original trajectory, samples $\hat{\mathbf{r}}(\mathbf{s}_k)$ of the filtered trajectory using the optimal $\bar{\Delta} = 0.0256$, and their linear interpolation $\mathbf{r}_{L,t_k}$.

Optimization Problem 1. Let d_H^* be the maximum admissible value for the Hausdorff distance, and let $d_{H,t}$ be the acceptable lower tolerance boundary for d_H^* . The objective is to find the optimal value $\bar{\Delta}$ of parameter Δ such that:

$$\min_{\Delta} J(\Delta), \quad J(\Delta) = \begin{cases} e_{d_H}(\Delta) & \text{if } 0 < e_{d_H}(\Delta) \leq d_{H,t}, \\ \infty & \text{otherwise,} \end{cases} \quad (3.9)$$

where $e_{d_H}(\Delta) = d_H^* - d_H(\Delta)$.

The optimal value $\bar{\Delta}$ solving (3.9) is the maximum value of parameter Δ such that $d_H(\bar{\Delta})$ approaches d_H^* from the left.

Numerical case study

A numerical case study has been implemented using the Optimization Problem 1, by setting $d_H^* = 0.025$ and $d_{H,t} = 0.1 \cdot 10^{-3}$, and using the original trajectory described by the linear interpolation $\mathbf{r}_{L,t}$ shown in Fig. 3.8(b). The black characteristic of Fig. 3.8(a) shows the evolution of function $d_H(\Delta)$ as a function of Δ , showing a non-monotonic behavior. The red characteristic shows the evolution of

function $d_H(\Delta)$ when Δ takes on values from the following geometric progression:

$$\Delta \in [\Delta_1, 2\Delta_1, 4\Delta_1, \dots], \quad (3.10)$$

where Δ_1 is a proper initial condition, showing a monotonic behavior in this case. By using the geometric progression (3.10), the Optimization Problem 1 is therefore well-defined and has a unique solution. The resulting $\bar{\Delta}$ for the considered numerical case study is 0.0256, for which the corresponding $d_H(\bar{\Delta})$ is 0.0223, as shown in Fig. 3.8(a). The linear interpolation $\mathbf{r}_{L,t}$ of the original trajectory, the samples $\hat{\mathbf{r}}(\mathbf{s}_k)$ of the filtered trajectory using the optimal $\bar{\Delta} = 0.0256$, and their linear interpolation $\mathbf{r}_{L,t_k}$ are shown in Fig. 3.8(b).

3.3 Experiments and evaluation

This section aims to demonstrate that time can introduce noise into the synchronization of geometric time series. Instead, by inducing the transformation in (3.6) thanks to the SS algorithm, one can extrapolate a skill that more accurately represents the informative content within the demonstration library. This improvement will be quantified in terms of the quality of the alignment (Sec. 3.3.1) and quality of the approximation (Sec. 3.3.2). Please note that the metrics mentioned below have been briefly introduced and referenced for the sake of readability and space. Readers interested in a detailed explanation of these metrics are encouraged to consult the Appendix C.

3.3.1 Alignment Algorithms Comparisons Based on LASA Handwriting Dataset

This simulation experiment evaluates the effectiveness of the SS algorithm in improving time series alignment. To achieve this, we conducted experiments using the LASA handwriting dataset (Khansari-Zadeh & Billard, 2011), a well-established benchmark for diverse robotic applications. For each handwriting in the dataset, we calculated the corresponding barycenter, which served as a reference for aligning each group. Simulations were conducted both in the time domain and using the SS algorithm to move to the arc-length domain of the handwriting curves. For the SS algorithm, we choose $\Delta = 0.02$. In both domains, we calculated metrics to evaluate the level of alignment between the handwriting and their respective barycenter.

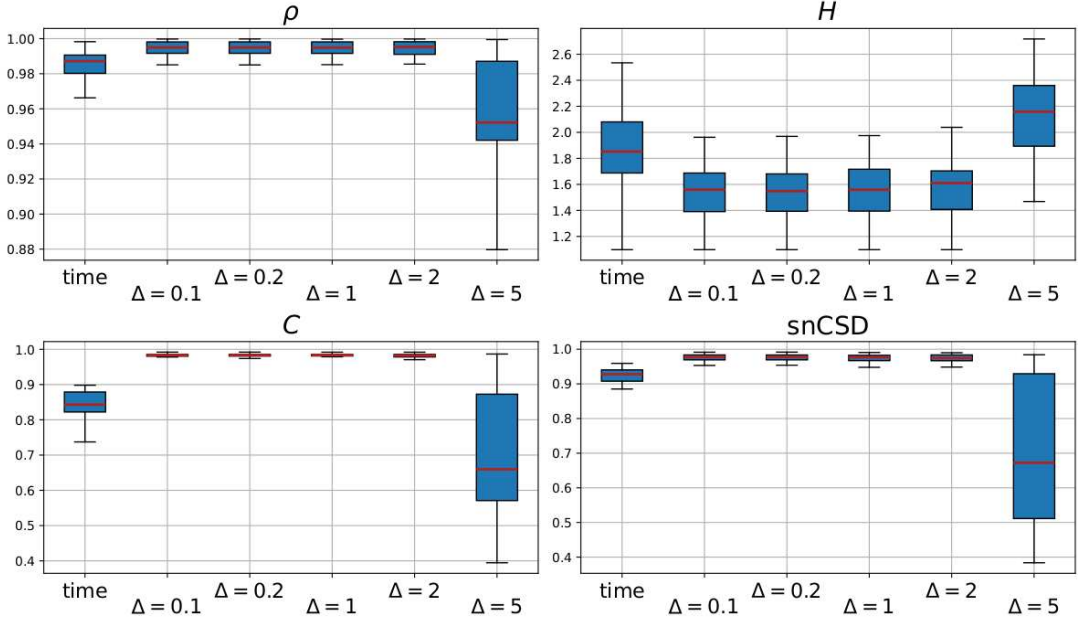


Figure 3.9: Box plots of simulation results on LASA handwriting dataset (Khansari-Zadeh & Billard, 2011) when comparing time versus arc-length parametrization obtained through SS with different values of Δ .

To measure multivariate synchrony, thus not considering just pairwise relationships, we utilized the multiSyncPy package (Hudson et al., 2023), which comprises metrics from different information theory studies. For our analysis, we focused on the following metrics: Rho (ρ), Entropy (H), Sum-Normalized Cross-Spectral Density (snCSD) and ℓ_2 norm error. The meaning of the metrics, instead, is summarized in the following.

Rho (ρ) (Richardson et al., 2012): Rho compares the phase of signals to assess their similarity. Calculating an aggregate relative phase across multiple signals measures how closely each signal’s phase aligns with the overall phase. *Entropy* (H) (Stevens & Galloway, 2014): from information theory, entropy quantifies the regularity and predictability of time series. Lower entropy values indicate synchronized, regular behavior, while higher entropy reflects increased variability and unpredictability. *Sum-Normalized Cross-Spectral Density* (*snCSD*) (Hudson et al., 2023): unlike coherence, which analyzes the power spectrum of the signal and averages across frequencies, ignoring single amplitude variations (White, 1984), snCSD accounts for the cross-spectral density at each frequency. This metric helps reduce the impact of noise, providing a less biased measure of coherence. *ℓ_2 norm error*: given two signals $\mathbf{x}_i$ and $\mathbf{y}_i$, $i = 0, 1, \dots, K$ with K being the number of samples, this metric computes the sum of each error $\|\mathbf{x}_i - \mathbf{y}_i\|_2$ and serves to express a geometric approximation metric within the aligned signals.

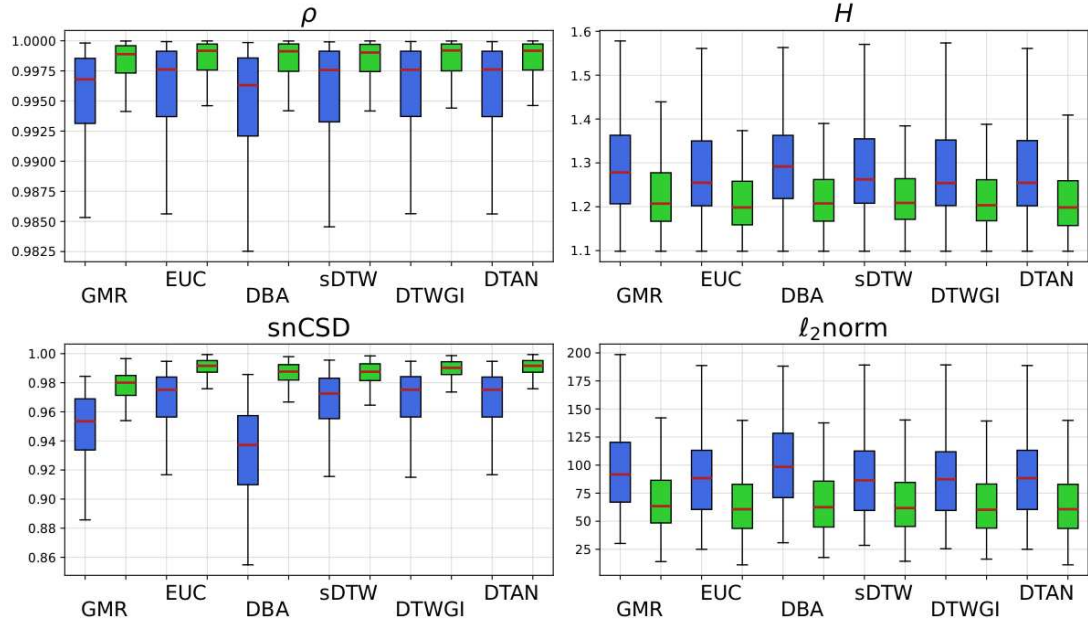


Figure 3.10: Box plots of simulation results on LASA handwriting dataset (Khansari-Zadeh & Billard, 2011) when comparing time (blue) versus arc-length (green) parametrization obtained through SS. From the left, the box reports the following metrics: Rho (ρ), Entropy (H), Sum-Normalized Cross-Spectral Density (snCSD), ℓ_2 norm error.

Preliminary tests compared time and arc length parametrization. In Fig. 3.9, the boxplots display results from an analysis of the LASA handwriting dataset trajectories. Synchronization metrics were computed for time-parametrized trajectories and arc-length curves using different Δ values. Results are promising, showing a substantial improvement for the arc-length parametrization. However, for high Δ values, such as $\Delta = 5$, the performance of the SS algorithm degrades. This occurs because larger Δ values increase the spacing between samples in the SS output, leading to a less accurate representation of the geometric information in the input data.

Further results from the simulations are reported in Fig. 3.10 and Table 3.1. We decided to compare different algorithms for barycenter computation, whose choice was based on (i) their popularity and (ii) the availability of their codes. The algorithms are Gaussian Mixture Regression (GMR) (Calinon et al., 2007), Euclidean distance averaging (EUC) (Tavenard et al., 2020), DTW Barycenter Averaging (DBA) (Petitjean et al., 2011), soft-DTW (sDTW) (Cuturi & Blondel, 2017), DTW with Global Invariance (DTWGI) (Vayer et al., 2020), and Diffeomorphic Temporal Alignment Nets (DTAN) (Shapira Weber et al., 2019). In Fig. 3.10, the results for each algorithm are shown with the time domain in blue

Table 3.1: Mean and standard deviation values of synchrony metrics for time (t) versus arc-length (s) parameterization tests.

		GMR	EUC	DBA
ρ	t	0.994 ± 0.0058	0.995 ± 0.0061	0.994 ± 0.0066
	s	0.997 ± 0.0025	0.998 ± 0.0027	0.998 ± 0.0027
H	t	1.292 ± 0.1187	1.281 ± 0.1169	1.301 ± 0.1179
	s	1.228 ± 0.0850	1.214 ± 0.0753	1.222 ± 0.0750
snCSD	t	0.942 ± 0.0467	0.963 ± 0.0437	0.927 ± 0.0482
	s	0.975 ± 0.0167	0.990 ± 0.0086	0.986 ± 0.0106
ℓ_2 norm	t	98.371 ± 46.5363	92.250 ± 47.1034	105.341 ± 52.3183
	s	70.236 ± 32.6700	66.967 ± 33.4304	69.136 ± 32.7610

		sDTW	DTWGI	DTAN
ρ	t	0.995 ± 0.0061	0.995 ± 0.0061	0.995 ± 0.0060
	s	0.998 ± 0.0026	0.998 ± 0.0026	0.998 ± 0.0025
H	t	1.287 ± 0.1175	1.281 ± 0.1172	1.282 ± 0.1170
	s	1.220 ± 0.0750	1.218 ± 0.0744	1.214 ± 0.0753
snCSD	t	0.961 ± 0.0440	0.963 ± 0.0436	0.963 ± 0.0437
	s	0.985 ± 0.0132	0.988 ± 0.0091	0.990 ± 0.0086
ℓ_2 norm	t	92.657 ± 48.4249	92.168 ± 47.4000	92.248 ± 47.1020
	s	68.717 ± 32.8724	67.632 ± 33.3254	66.970 ± 33.4248

and the arc-length domain in green. It is clear that whenever the SS algorithm is applied, the metrics improve significantly across the dataset, as summarized by the statistics of the boxplots. This indicates synchrony is enhanced when trajectories are considered in their arc-length domain. This is crucial as it allows for extracting a more representative skill (e.g., the barycenter in this case) from the demonstrations, unaffected by timing discrepancies. In particular, Table 3.1 shows that the SS-DTAN algorithm performs slightly better than other methods. Similar outcomes are seen with SS-EUC, though its performance drops when there are significant time variations in the demonstrations (e.g., pauses or varying speeds). Therefore, only DTAN will be further considered in the next section.

3.3.2 Skill Synthesis from Demonstrations

This section analyzes the geometric approximation error of the computed barycenter. To do this, we created a custom dataset of end-effector position recordings. Our dataset comprises 21 symbols recorded using a Franka Emika Panda robot

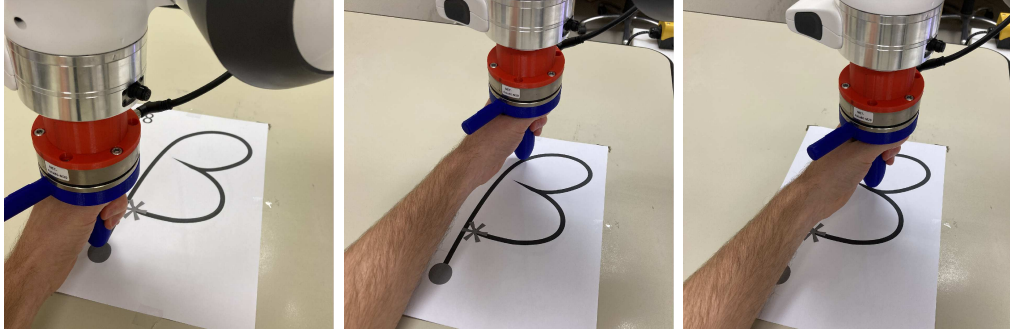


Figure 3.11: Frames of robot demonstrations over the Autolab Co-Manipulation dataset

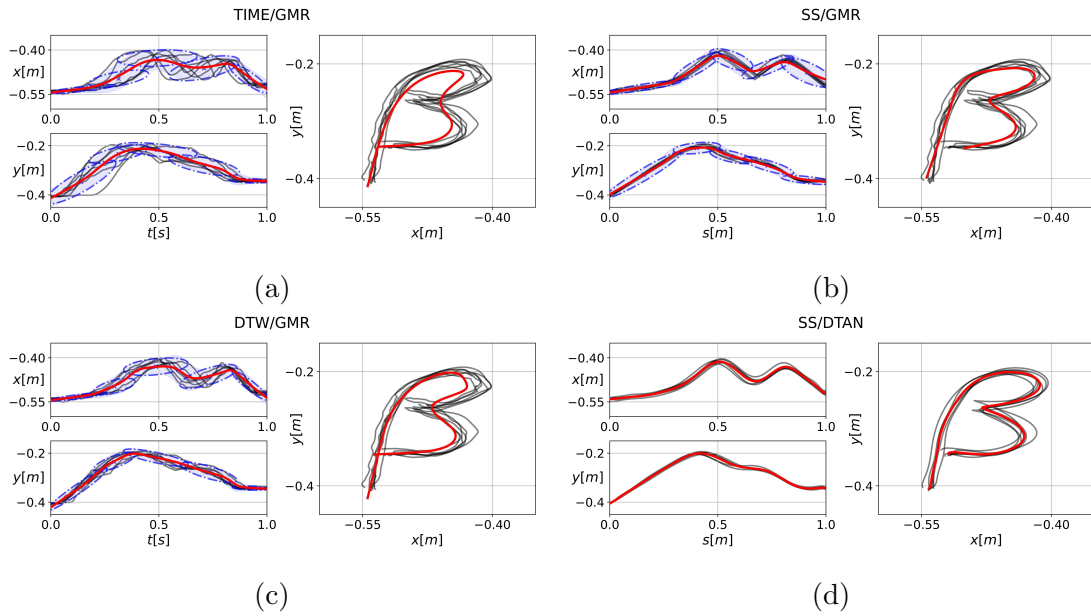


Figure 3.12: Barycenter computation using time-based (a)(c) and arc-length parametrized (b)(d) curves, with normalized intervals for better visualization.

in gravity compensation mode, see Fig. 3.11. Six recordings were made for each symbol, maintaining the same start and endpoint, but varying the speed along the curve and introducing pauses at the user’s discretion. This concern differs from the LASA handwriting dataset, where recordings have shared endpoints but varying starting positions and minimal timing discrepancies.

We evaluated the results using two metrics: the Hausdorff distance (d_H) (Donoso et al., 2008) as defined in (3.8) and, without loss of consistency, the DTW score (d_{DTW}) (Muller, 2007) as defined in (3.1). While the former captures the maximum approximation error, the latter provides a clearer view of the overall approximation error when comparing two signals. We tested four scenarios: (i) feeding time-based recordings to GMR (TIME/GMR), (ii) filtering recordings with SS

Table 3.2: Mean and standard deviation values for approximation metrics.

	TIME/GMR	DTW/GMR	SS/GMR	SS/DTAN
$d_H(e^{-2})$	2.4 ± 0.81	2.1 ± 0.65	2.0 ± 0.63	0.9 ± 0.26
d_{DTW}	1.34 ± 0.645	0.96 ± 0.432	0.09 ± 0.029	0.06 ± 0.018

before sending them to GMR (SS/GMR), (iii) using DTW to align trajectories before applying GMR (DTW/GMR), and (iv) combining SS with DTAN. For the SS algorithm, we choose $\delta = 0.005$. GMR is widely used in robotics for skill extraction, which justifies its inclusion in this study. Additionally, the DTW/GMR combination has been adopted in many seminal robotics works, making it a relevant comparison algorithm (Calinon et al., 2007; Su et al., 2021; Zheng et al., 2022).

Figure 3.11 shows the results for the β -shape recordings. The aligned demonstrated trajectories (black) were fitted with a GMM using $N = 5$ components (blue) before computing the barycenter (red) with GMR, except for the SS/DTAN case where GMR was not used. In Fig. 3.12(a), temporal distortions lead to high variance, reducing the skill quality. Lower variances are observed in the SS/GMR (b) and DTW/GMR (c) cases, with the latter showing a less accurate barycenter due to the reference alignment choice. To conclude, the SS/DTAN (d) case appears to be the most promising scenario, combining the alignment provided by the SS algorithm and the regression performed by the DTAN model.

Quantitative results are presented in Fig. 3.13 and Table 3.2. As shown on the figure’s left, the lack of prior alignment (TIME/GMR case) can result in significant displacement from the demonstrated trajectories when computing the barycenter. In contrast, both DTW/GMR and SS/GMR perform better, with comparable outcomes. However, differences become noticeable when considering the second metric, d_{DTW} , which evaluates the entire signal. Here, the SS/GMR combination significantly improves over both TIME/GMR and DTW/GMR. To conclude, the SS/DTAN method achieves the best performance in terms of both d_H and d_{DTW} , which can benefit from better alignment through the arc-length domain and improved barycenter computation with the DTAN algorithm. The quality of the SS/DTAN outcome can be further appreciated in Fig. 3.14, which shows in red the barycenter computed for each symbol of the dataset.

3.3.3 Discussion

The key advantage of the SS algorithm is that it produces time-agnostic curves, moving from time to the arc-length domain (Braglia et al., 2025). As shown in

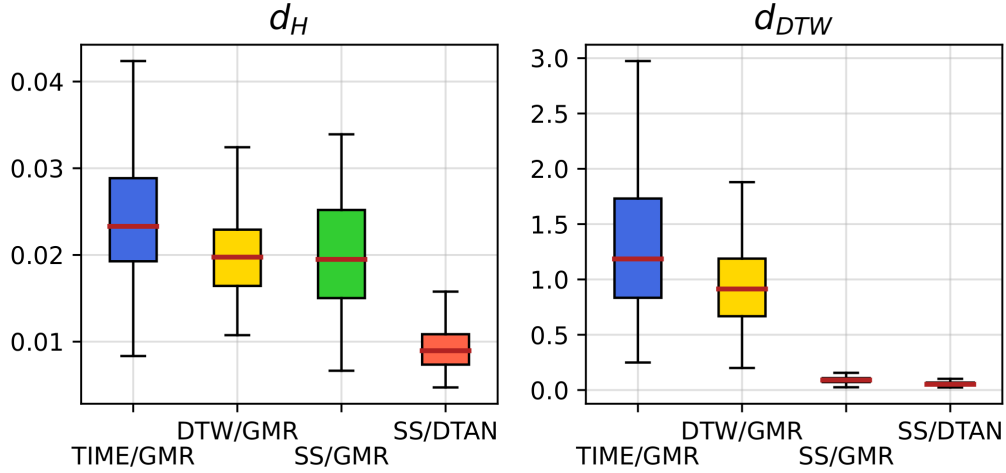


Figure 3.13: Box plots for approximation metrics result.

Section 3.3.1, this decoupling enhances alignment reliability and improves the performance of the analyzed algorithms.

The LASA handwriting dataset consists of trajectory recordings with a common endpoint starting from different positions, but it lacks significant timing discrepancies such as pauses or speed variations. To further evaluate the SS algorithm, we created our own handwriting dataset. Unlike LASA, this dataset introduces significant time distortions while maintaining the same path and endpoints across repetitions. This setup allowed us to analyze the approximation error introduced when computing the barycenter of SS-filtered curves. We found that computing the barycenter directly from time-domain trajectories often introduces distortions. However, SS and GMR yielded performance comparable to state-of-the-art methods like DTW/GMR and a refined DTAN model due to SS’s preprocessing of the trajectories. Unlike SS/GMR, DTW/GMR requires prior alignment, and DTAN needs a training phase to fine-tune parameters. In contrast, SS simply filters trajectories, balancing alignment and geometric accuracy. It also generates curves with well-defined derivatives, allowing different timing laws to be applied later, making SS a strong alternative for signal alignment in robotic trajectories.

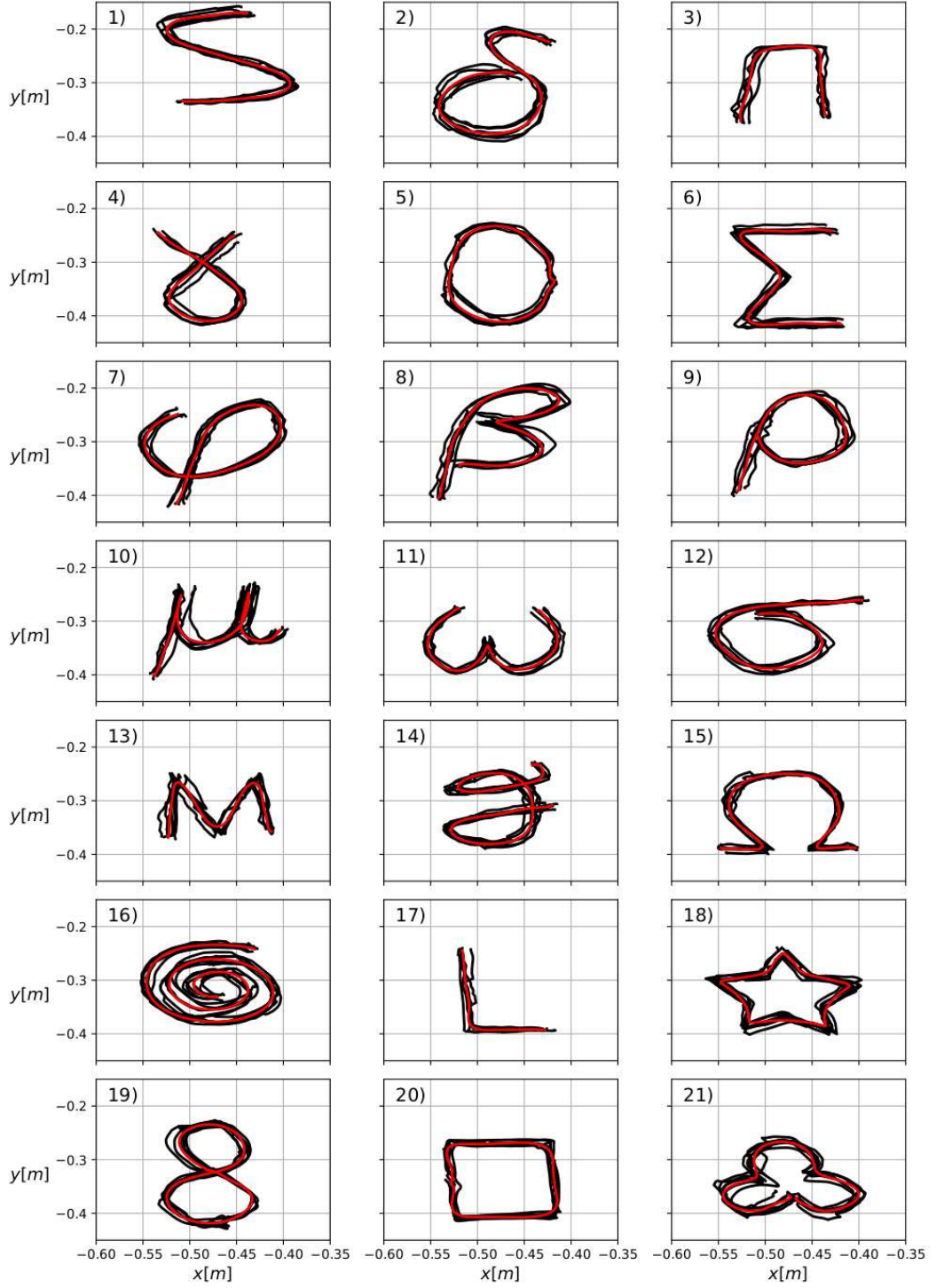


Figure 3.14: Autolab Co-Manipulation Dataset. Recordings are reported in black, while results for SS/DTAN are shown in red.

Motion Representation

In LfD, after multiple demonstrations are provided, an average behavior is extrapolated from the recorded dataset. This average behavior is commonly referred to as a skill, and it must be representative of the given demonstrations while still allowing space for possible further teaching refinements or task generalization.

Once the skill has been correctly retrieved from the demonstration data, the extracted information needs to be adapted such that it can be translated into robot movements. In particular, from an LfD perspective, it is common practice to manipulate the skill with the following steps: (i) skill expression, (ii) skill encoding, (iii) skill execution. In the next paragraphs, each of these steps is analyzed.

4.1 Skill Expression

When the constrain curve $\boldsymbol{\mu}$ is built upon the demonstration of a certain task, it is convenient to adopt encoding techniques to reproduce the target trajectory with good approximation and low computational effort. In kinesthetic teaching, for example, the target trajectory $\boldsymbol{r}_t$, $t \in [0, T]$, is obtained from the recordings of sensory and motor information. However, in case our intention is to reproduce the demonstrated task, it is convenient to retrieve an analytical expression to it. More generally, we distinguish among two types of fittings, i.e. *interpolation* and *approximation* (Biagiotti & Melchiorri, 2008). The former consists in imposing the condition that the fitting curve $\boldsymbol{\mu}$ should pass through some points of $\boldsymbol{r}$; the latter, usually finds a non-linear mapping such that we obtain $\boldsymbol{\mu} \approx \boldsymbol{r}$ with some degree of accuracy (Calinon et al., 2007).

Focusing on approximation techniques, recently a wide variety of applications rely on basis functions approximation for trajectory encoding (Calinon, 2020b). In this regression technique, the non-linear mapping is given by a weighted su-

perposition of non-linear terms, namely

$$\boldsymbol{\mu}(s) = \sum_{i=1}^N \boldsymbol{\omega}_i \phi_i(s). \quad (4.1)$$

In 4.1, the constrain curve is built as a sum of N basis function ϕ_i , each weighted by its respective parameter $\boldsymbol{\omega}_i$ which is usually obtained as the solution of (Ijspeert et al., 2013)

$$\boldsymbol{\omega}_i = \underset{\boldsymbol{\omega}_i}{\operatorname{argmin}} \sum_{t=0}^T [\mathbf{r}_t - \boldsymbol{\omega}_i \phi_i(s_t)]^2. \quad (4.2)$$

Note that, in many cases, the minimization problem in 4.2 can be solved analytically, which further simplifies the computation of $\boldsymbol{\mu}$ (Biagiotti & Melchiorri, 2008). In particular, the optimal control problem in (4.2) can be written in a matrix form as:

$$\boldsymbol{\omega} = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \|\mathbf{r} - \boldsymbol{\Psi}^\top \boldsymbol{\omega}\|^2, \quad (4.3)$$

where $\boldsymbol{\omega} \in \mathbb{R}^{(N \times 1)}$ and:

$$\boldsymbol{\Psi} = \begin{bmatrix} \phi_1(s_0) & \cdots & \phi_1(s_T) \\ \phi_2(s_0) & \cdots & \phi_2(s_T) \\ \vdots & \ddots & \vdots \\ \phi_N(s_0) & \cdots & \phi_N(s_T) \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} \mathbf{r}_0 \\ \vdots \\ \mathbf{r}_T \end{bmatrix} \quad (4.4)$$

with $\boldsymbol{\Psi} \in \mathbb{R}^{(N \times T)}$, $\mathbf{r} \in \mathbb{R}^{(T \times 1)}$, and T being the total duration $\mathbf{r}$.

Typically, one may want to ensure that the approximation given by basis functions matches some points of interest, otherwise called via-points (Paraschos et al., 2013). More commonly, one may want to guarantee that the start and end points are correctly interpolated. For this purpose, it is possible to add constraints on the problem in (4.3) as

$$\sum_{i=1}^N \omega_i \phi_i(s_0) = \mathbf{r}_0, \quad \sum_{i=1}^N \omega_i \phi_i(s_T) = \mathbf{r}_T, \quad (4.5)$$

or, alternatively, (4.5) can be written in a matrix form as:

$$\mathbf{C}\boldsymbol{\omega} = \mathbf{H}, \quad \text{where } \mathbf{C} = \begin{bmatrix} \boldsymbol{\Phi}(s_0)^\top \\ \boldsymbol{\Phi}(s_T)^\top \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{r}_0 \\ \mathbf{r}_T \end{bmatrix}, \quad (4.6)$$

where $\boldsymbol{\Phi}(s_t) = [\phi_1(s_t), \phi_2(s_t), \dots, \phi_N(s_t)]^\top$. Note that, during the learning

of weights $\boldsymbol{\omega}$ we assumed the reference goal $\mathbf{g}_r = \mathbf{r}_T$.

The optimal control problem (4.3) subject to (4.6) represents a constrained least-squares minimization problem, which can be analytically solved using the Lagrange multipliers method (Biagiotti & Melchiorri, 2008). Specifically, (4.3) subject to (4.6) is the same as solving:

$$\text{tr} [(\mathbf{r} - \boldsymbol{\Psi}^\top \boldsymbol{\omega})^\top \mathbf{Q}(\mathbf{r} - \boldsymbol{\Psi}^\top \boldsymbol{\omega}) + \mathbf{L}^\top (\mathbf{C}\boldsymbol{\omega} - \mathbf{H})] \quad (4.7)$$

being $\mathbf{L} = [\lambda_1, \lambda_2]^\top$ the vector of Lagrange multipliers, $\mathbf{Q} \in \mathbb{R}^{(T \times T)}$ is precision matrix over the approximation constraint and “tr” denotes the trace operator. By differentiating (4.7) and equating it to zero, after some algebraic manipulations one obtains:

$$\begin{aligned} \boldsymbol{\Psi} \mathbf{Q} \boldsymbol{\Psi}^\top \boldsymbol{\omega} + \mathbf{C}^\top \mathbf{L} &= \boldsymbol{\Psi} \mathbf{Q} \mathbf{r}, \\ \mathbf{C} \boldsymbol{\omega} &= \mathbf{H}. \end{aligned} \quad (4.8)$$

If matrices $\mathbf{P} = \boldsymbol{\Psi} \mathbf{Q} \boldsymbol{\Psi}^\top$ and $\mathbf{R} = \mathbf{C}(\boldsymbol{\Psi} \mathbf{Q} \boldsymbol{\Psi}^\top)^{-1} \mathbf{C}^\top$ are both invertible, the solution of (4.8) boils down to:

$$\begin{aligned} \mathbf{L} &= \mathbf{R}^{-1}(\mathbf{C} \mathbf{P}^{-1} \boldsymbol{\Psi} \mathbf{Q} \mathbf{r} - \mathbf{H}), \\ \boldsymbol{\omega} &= \mathbf{P}^{-1} \boldsymbol{\Psi} \mathbf{Q} \mathbf{r} - \mathbf{P}^{-1} \mathbf{C}^\top \mathbf{L}, \end{aligned}$$

providing the optimal weights vector $\boldsymbol{\omega}$ solving the constrained optimization problem (4.3) subject to (4.6).

While guaranteeing an analytical expression of the target trajectory $\mathbf{r}$, basis function approximation provides prior knowledge on its continuity, including the possibility of adjust smoothness properties of $\boldsymbol{\mu}$ (Biagiotti & Melchiorri, 2008; Sidiropoulos et al., 2023), thus generalize the curve with respect to different via-points or goal positions (Jankowski et al., 2023). Additionally, basis function approximation proves beneficial in all those cases where a low dimensional representation of the demonstrated task is required, as the representation of $\mathbf{r}$ is basically included into the weight vector $\boldsymbol{\omega}$ (Lauretti et al., 2017; Pastor et al., 2009; Ruckert & d’Avella, 2013).

In the next paragraph, we briefly introduce three widely types of basis functions, namely *Radial Basis Functions*, *Bernstein Polynomials* and *Fourier Series*, whose trajectory approximation is summarized in Fig. 4.1. For the sake of simplicity, the following paragraphs will refer to the approximation of 1D target trajectories $\mathbf{r}$, but the presented formulation can be easily extended to the multidimensional case (Calinon, 2009).

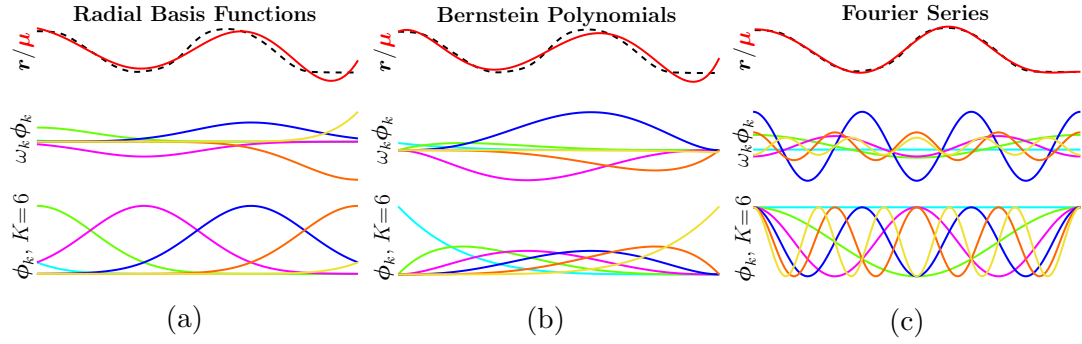


Figure 4.1: Approximation of 1D trajectory $\mathbf{r}$ with $\boldsymbol{\mu} = \sum_{k=1}^K \omega_k \phi_k$, distinguishing among three cases where different types of basis functions ϕ_k are superimposed.

4.1.1 Radial Basis Functions

Radial Basis Functions (RBFs) are shown in Fig. 4.1(a) and can be computed as

$$\phi_i(s) = \exp\left(\frac{-(s - c_i)^2}{2h}\right), \quad (4.9)$$

being c_i the center of $\phi_i(s)$, where usually the centers for all N RBFs are linearly spaced in s , and h is a common bandwidth. Famous application of RBFs are Dynamic Movement Primitives (Ijspeert et al., 2013) and Probabilistic Movement Primitives (Paraschos et al., 2013). One of the main characteristic of RBFs is that the exponential nature of those basis functions leads them to be infinite differentiable, guaranteeing smoothness of high order derivatives. RBFs can also be used in their scaled form, by using $\phi_i(s) = \frac{\phi_i(s)}{\sum_{i=1}^N \phi_i(s)}$.

4.1.2 Bernstein Polynomials

Bernstein polynomials are shown in Fig. 4.1(b) and can be computed as

$$\phi_{i,N}(s) = \binom{N}{i} (1-s)^{N-i} s^i, \quad \text{with} \quad \binom{N}{i} = \frac{N!}{i!(N-i)!}. \quad (4.10)$$

Note that 4.10 describes $\phi_{i,N}$ as N -degree Bernstein basis polynomial for $s \in [0, 1]$. If s spans a larger interval, let's say $s \in [0, L]$, it is sufficient to substitute in 4.10 $(L - s)$ instead of $(1 - s)$.

One of the benefits of the formulation in 4.10 is that it allows to obtain easily and recursively an analytical expression for higher order derivatives, namely (Doha

et al., 2011):

$$\frac{d^p}{ds^p}(\phi_{i,N}(s)) = \frac{N!}{(N-p)!} \sum_{k=\max(0, i+p-N)}^{\min(i,p)} (-1)^{k+p} \binom{p}{k} \phi_{i-k, N-p}(s),$$

where p indicates the order of the derivative to be computed. Bernstein polynomials have been widely used in disparate applications, including computer graphics (Farouki, 2012) and motion planning (Biagiotti & Melchiorri, 2008; Jankowski et al., 2023).

4.1.3 Fourier Series

Fourier series are shown in Fig. 4.1-right and can be computed as

$$\phi_i(s) = \exp(s i 2\pi j), \quad (4.11)$$

indicating with j the imaginary unit ($j^2 = -1$). Notably, in Fourier series the index i in 4.1 is equal to $[-N+1, -N+2, \dots, N-1]$. However, for the case where $\boldsymbol{\mu}(s)$ is real and even, one can simplify 4.11 to

$$\phi_i(s) = \cos(s i 2\pi j),$$

with $i = [0, 1, \dots, N-1]$ (Calinon, 2020b).

Fourier series dates back to early 1800s and are widely known as a powerful instrument and essential instrument to provide a connection between time and frequency domains. Fourier series applications are countless, including image, signal and speech processing, among others (Brunton & Kutz, 2022).

One of the benefits of basis function approximation introduced in (4.1), is the possibility to easily compute higher order derivatives since the expression of $\boldsymbol{\mu}(s)$ is analytical. For instance, once the phase variable $s = s_t$ has been defined, the first and second derivatives of the parameterized trajectory $\boldsymbol{\mu}(s_t)$ with respect to time can be computed as

$$\begin{aligned} \dot{\boldsymbol{\mu}}(s_t) &= \frac{\partial \boldsymbol{\mu}(s_t)}{\partial s_t} \frac{ds_t}{dt} = \boldsymbol{\mu}'(s_t) \dot{s}_t, \\ \ddot{\boldsymbol{\mu}}(s_t) &= \frac{d\dot{\boldsymbol{\mu}}(s_t)}{dt} = \boldsymbol{\mu}''(s_t) \dot{s}_t^2 + \boldsymbol{\mu}'(s_t) \ddot{s}_t, \end{aligned} \quad (4.12)$$

where the prime and dot notations denote the differentiation with respect to the phase variable s and with respect to time, respectively.

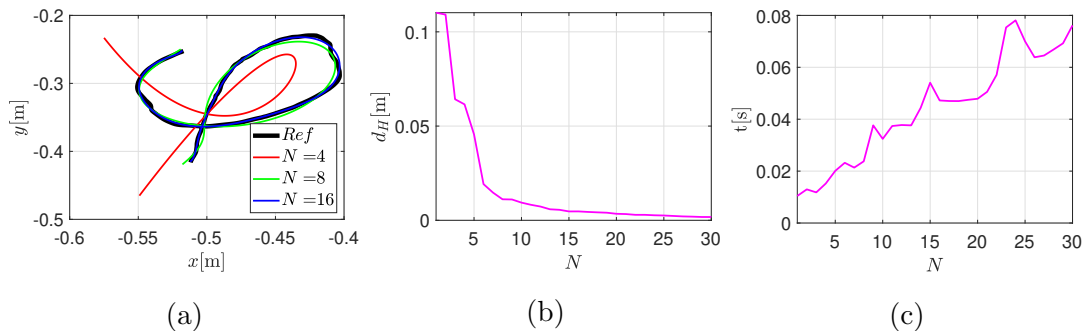


Figure 4.2: Basis functions approximation results when varying the number of components N .

Another concern when using basis functions approximation is on the choice of the number of components N in (4.1) to be used. While no clear heuristic is present on literature, it is reasonable to adjust N based on the accuracy and computational needs. For instance, an example of the outcome for different values of N is shown in Fig. 4.2, where Bernstein polynomials were used. One can clearly observe in Fig. 4.2(a) how increasing the number N leads to a better approximation of the reference curve depicted in black. This is further validated in Fig. 4.2(b), which shows how the Hausdorff distance d_H between the approximated and reference curves decreases for higher values of N . Indeed, this comes at the expenses of an higher computational time as shown in Fig. 4.2(c). To conclude, depending on the level of required precision, it is often common practice to choose N as low as possible, in order to improve the efficiency and memory requirements of codes.

4.2 Skill Encoding with Gaussian Mixture Models

Probabilistic inference methods (Fisac et al., 2018; Mukadam et al., 2018; Tous-saint & Goerick, 2010) provide a powerful framework for trajectory optimization, integrating task constraints, goals, and motion priors through the use of joint probability distributions that capture conditional dependencies. Among these approaches, Gaussian Mixture Models (GMM) have emerged as a versatile tool for modeling and learning complex tasks. In (Calinon et al., 2007), GMMs are employed within a LfD framework to encode multiple tasks efficiently. This enables the retrieval of associated skills using regression techniques, showcasing the effectiveness of GMMs in representing and generalizing motion patterns. This technique is explained in the following paragraphs, following the concepts intro-

duced in (Calinon, 2009).

4.2.1 Gaussian Mixture Models

Consider a dataset $[\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_T]$, where $\mathbf{r}_t \in \mathbb{R}$. These can represent, for example, the samples of a trajectory collected during the demonstration process. For a Gaussian Mixture Model (GMM) with K components, the likelihood of a sample $\mathbf{r}_t$ being characterized by the GMM is given by

$$\mathcal{P}(\mathbf{r}_t) = \sum_{k=1}^K \mathcal{P}(k) \mathcal{P}(\mathbf{r}_t|k), \quad (4.13)$$

where $\mathcal{P}(k)$ denotes the prior probability and $\mathcal{P}(\mathbf{r}_t|k)$ represents the conditional probability density function parameterized as

$$\mathcal{P}(k) = \pi_k, \quad \mathcal{P}(\mathbf{r}_t|k) = \mathcal{N}(\mathbf{r}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_k|}} \exp^{-\frac{1}{2} (\mathbf{r}_t - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{r}_t - \boldsymbol{\mu}_k)}. \quad (4.14)$$

The GMM is defined by the set of parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, which correspond to the prior probabilities, mean vectors, and covariance matrices of the model. The prior probabilities $\pi_k \in [0, 1]$ satisfy $\sum_{k=1}^K \pi_k = 1$, indicating the likelihood of the k -th component being responsible for the observed outcome.

To determine the parameters of the GMM, the model is fitted to the dataset. This training is typically carried out using the Expectation-Maximization (EM) algorithm. The EM algorithm performs maximum likelihood estimation by alternately estimating the GMM parameters and optimizing the model, repeating these steps until convergence. Starting from (4.13), the posterior probability $\mathcal{P}(k|\mathbf{r}_t)$ is computed using Bayes' theorem:

$$\mathcal{P}(k|\mathbf{r}_t) = \frac{\mathcal{P}(k) \mathcal{P}(\mathbf{r}_t|k)}{\sum_{j=1}^K \mathcal{P}(j) \mathcal{P}(\mathbf{r}_t|j)}. \quad (4.15)$$

For simplicity, let $p_{k,t} = \mathcal{P}(k|\mathbf{r}_t)$. The sum of posterior probabilities in (4.15) is then denoted as $E_k = \sum_{t=1}^T p_{k,t}$. Using the EM algorithm, the parameters

$\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, E_k\}$ describing the GMM are iteratively updated as follows:

Expectation step:

$$p_{k,t}^u = \frac{\pi_k^u \mathcal{N}(\mathbf{r}_t; \boldsymbol{\mu}_k^u, \boldsymbol{\Sigma}_k^u)}{\sum_{j=1}^K \pi_j^u \mathcal{N}(\mathbf{r}_t; \boldsymbol{\mu}_j^u, \boldsymbol{\Sigma}_j^u)},$$

$$E_k^u = \sum_{t=1}^T p_{k,t}^u.$$

Maximization step:

$$\pi_k^{u+1} = \frac{E_k^u}{T}, \quad (4.16)$$

$$\boldsymbol{\mu}_k^{u+1} = \frac{\sum_{t=1}^T p_{k,t}^u \mathbf{r}_t}{E_k^u},$$

$$\boldsymbol{\Sigma}_k^{u+1} = \frac{\sum_{t=1}^T p_{k,t}^u (\mathbf{r}_t - \boldsymbol{\mu}_k^{u+1})(\mathbf{r}_t - \boldsymbol{\mu}_k^{u+1})^\top}{E_k^u},$$

where u denotes the index for the u th iteration of the EM algorithm, and $\mathcal{N}(\mathbf{r}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is as defined in (4.14). The two steps in (4.16) are repeated until a specified convergence criterion is met. Specifically, EM evaluates the relative increase in the log-likelihood $\mathcal{L}$ across iterations, stopping when the increase ratio falls below a predefined threshold C , i.e.

$$\frac{\mathcal{L}^{u+1}}{\mathcal{L}^u} < C, \quad \text{where } \mathcal{L} = \sum_{t=1}^T \log(\mathcal{P}(\mathbf{r}_t)). \quad (4.17)$$

4.2.2 Gaussian Mixture Regression

Once the EM algorithm has converged, the trained GMM can be used to evaluate the likelihood of new data points $\mathbf{r}$ belonging to the model. This is achieved by calculating $\mathcal{P}(\mathbf{r})$ using (4.13)- (4.14) as follows:

$$\mathcal{P}(\mathbf{r}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{k=1}^K \pi_k \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_k|}} \exp^{-\frac{1}{2}((\mathbf{r} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{r} - \boldsymbol{\mu}_k))}. \quad (4.18)$$

Once the GMM has been fitted to the dataset, it is possible to analyze the behavior of its components by estimating the average trend in the data. This can be done using regression over the model, providing an input variable, such as t , to compute the corresponding output $\hat{\mathbf{r}}_t$. This approach is commonly referred to as Gaussian Mixture Regression (GMR).

To achieve this, input and output components are separated and represented as:

$$\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^I \\ \boldsymbol{\mu}_k^O \end{bmatrix}, \quad \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^I & \boldsymbol{\Sigma}_k^{IO} \\ \boldsymbol{\Sigma}_k^{OI} & \boldsymbol{\Sigma}_k^O \end{bmatrix}. \quad (4.19)$$

For a datapoint $\mathbf{x} = [t, \mathbf{r}_t]$ with input variable t , the expected distribution of the output $\mathbf{r}_t$ for a given Gaussian component k is defined as:

$$\mathcal{P}(\mathbf{r}_t|t, k) \sim \mathcal{N}(\hat{\mathbf{r}}_{k,t}, \hat{\boldsymbol{\Sigma}}_{k,t}), \quad \text{where} \quad \begin{aligned} \hat{\boldsymbol{\mu}}_{k,t} &= \boldsymbol{\mu}_k^O + \boldsymbol{\Sigma}_k^{OI} (\boldsymbol{\Sigma}_k^I)^{-1} (t - \boldsymbol{\mu}_k^I), \\ \hat{\boldsymbol{\Sigma}}_{k,t} &= \boldsymbol{\Sigma}_k^O - \boldsymbol{\Sigma}_k^{OI} (\boldsymbol{\Sigma}_k^I)^{-1} \boldsymbol{\Sigma}_k^{IO}. \end{aligned} \quad (4.20)$$

When the input t is known, the expected distribution of the output $\mathbf{r}_t$ can be estimated as:

$$\mathcal{P}(\mathbf{r}_t|t) \sim \sum_{k=1}^K h_k \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\boldsymbol{\Sigma}}_k), \quad (4.21)$$

where $h_k = \mathcal{P}(k|t)$ represents the probability that the k th Gaussian component is responsible for t , calculated as:

$$h_k = \frac{\mathcal{P}(k)\mathcal{P}(t|k)}{\sum_{j=1}^K \mathcal{P}(j)\mathcal{P}(t|j)} = \frac{\pi_k \mathcal{N}(t; \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I)}{\sum_{j=1}^K \pi_j \mathcal{N}(t; \boldsymbol{\mu}_j^I, \boldsymbol{\Sigma}_j^I)}. \quad (4.22)$$

For instance, if $\mathbf{x} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})$, an affine transformation $\mathbf{A}\mathbf{x}$ under a matrix $\mathbf{A}$ follows the linear transformation property of Gaussian distributions:

$$\mathbf{A}\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \text{where} \quad \boldsymbol{\mu} = \mathbf{A}\tilde{\boldsymbol{\mu}}, \quad \boldsymbol{\Sigma} = \mathbf{A}\tilde{\boldsymbol{\Sigma}}\mathbf{A}^T. \quad (4.23)$$

Using this property, one can exploit (4.22) to compute the conditional expectation of $\mathbf{r}_t$ given t , directly from the distribution $\mathcal{N}(\hat{\mathbf{x}}, \hat{\boldsymbol{\Sigma}})$ with the following parameters:

$$\hat{\mathbf{x}} = \sum_{k=1}^K h_k \hat{\mathbf{x}}_k \quad \text{and} \quad \hat{\boldsymbol{\Sigma}} = \sum_{k=1}^K h_k \hat{\boldsymbol{\Sigma}}_k. \quad (4.24)$$

An example of encoding using GMM/GMR is illustrated in Fig. 4.3. In each subfigure, the demonstration trajectories are shown in black, the Gaussian covariances of the GMM are displayed in blue, and the expected output $\hat{\mathbf{r}}_t = [\hat{x}_t, \hat{y}_t]$ obtained through GMR is represented in red. From Fig. 4.3(a) to Fig. 4.3(c), the number of components K is increased to demonstrate how adjusting K enhances the fit to the dataset. It should be noted that there is no established heuristic in the literature for choosing K . Using too few components may fail to capture the dataset's structure, while too many components may lead to overfitting,

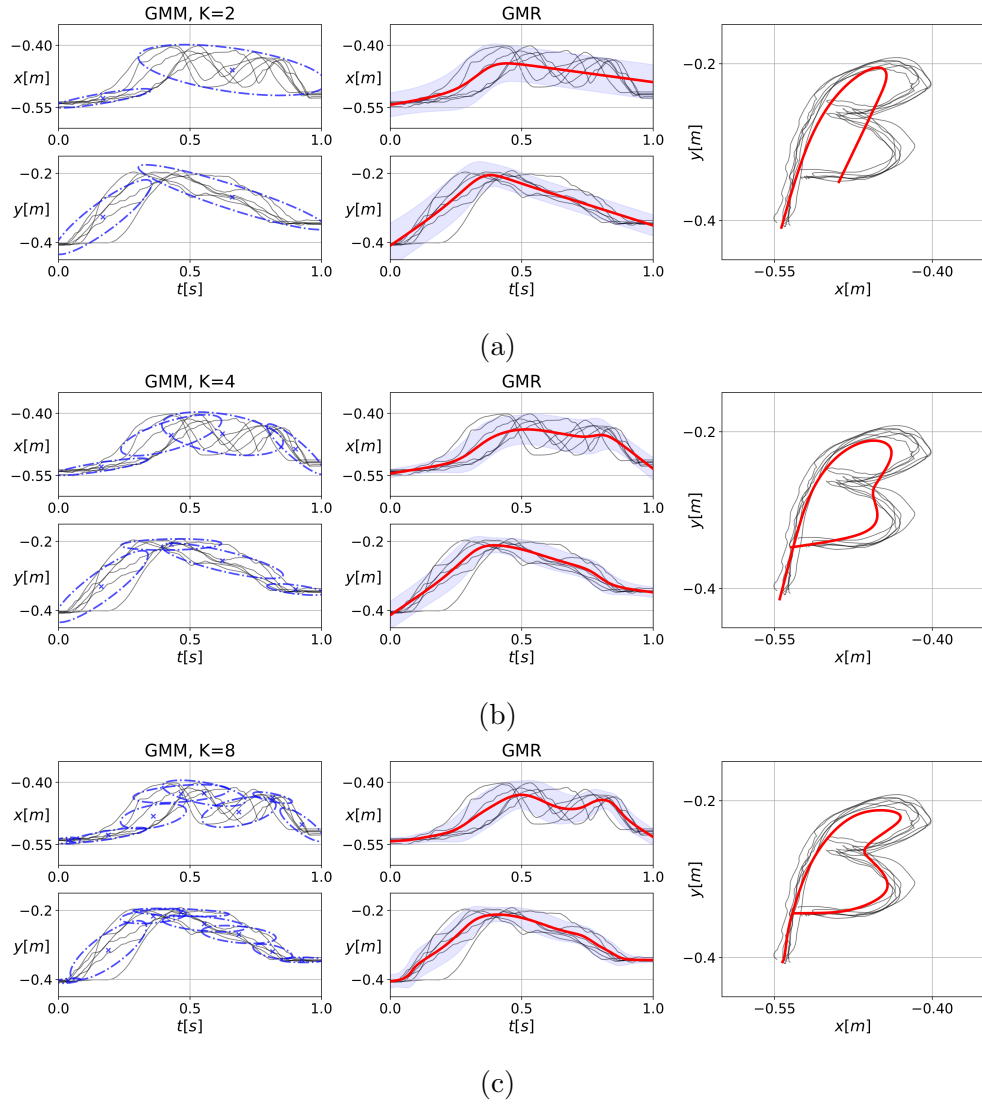


Figure 4.3: GMM results when varying the number of components K .

which compromises generalization to new data. Additionally, increasing K can significantly increase computational demands, which is particularly undesirable in real-time applications.

4.3 Skill Execution: Dynamic Movement Primitives

One of the most widespread encoding/planning techniques available in the literature are Dynamic Movement Primitives (DMP) (Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023), which were first introduced in (Ijspeert et al., 2013) as a way to encode motor movement with convenient stability properties. DMP con-

sist of a mass-spring-damper system with the addition of a forcing term (Ijspeert et al., 2002) and can be classified into rhythmic or discrete DMP, depending on whether the attractor system is a limit cycle or a point (Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023), respectively. Discrete DMP are typically formulated as follows (Ijspeert et al., 2002; Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023):

$$\tau \dot{z}_t = \alpha[\beta(g - y_t) - z_t] + f(s_t), \quad (4.25)$$

$$\tau \dot{y}_t = z_t, \quad (4.26)$$

$$\tau \dot{s}_t = -\delta s_t, \quad (4.27)$$

where Eqs. (4.25)-(4.26) are referred to as the Transformation System (TS), whereas Eq. (4.27) is referred to as the Canonical System (CS). The parameter τ is a time constant, α and β are system parameters chosen to have a critically damped TS (i.e. $\beta = \alpha/4$), g is the target goal and the forcing term $f(s_t)$ shapes the generated trajectory y_t through the TS dynamics. The TS (4.25)- (4.26) is a second-order system which can be rewritten as the following second-order differential equation:

$$\tau^2 \ddot{y}_t + \alpha \tau \dot{y}_t + \alpha \beta (y_t - g) = f(s_t), \quad (4.28)$$

representing a globally stable second-order system characterized by the unique equilibrium point $(y_t, \dot{y}_t) = (g, 0)$ when $f(s_t) = 0$. The CS (4.27) is a first-order system representing the evolution of the phase variable s_t , with δ being a positive constant (Ijspeert et al., 2002). The forcing term $f(s_t)$ in (4.25) is typically expressed as the linear combination of radial basis functions (Ijspeert et al., 2013):

$$f(s_t) = \sum_{i=1}^N \omega_i \phi_i(s_t) \cdot s_t \cdot (g - y_0), \quad (4.29)$$

where

$$\phi_i(s_t) = \frac{b_i(s_t)}{\sum_{j=1}^N b_j(s_t)}, \quad b_i(s_t) = \exp\left(\frac{-(s_t - c_i)^2}{2h}\right). \quad (4.30)$$

Note that, since $s_t \rightarrow 0$ when $t \rightarrow \infty$ from (4.27), $f(s_t) \rightarrow 0$ from (4.29), therefore its impact in the TS (4.25)- (4.26) is limited in time. The constant $(g - y_0)$ in (4.29) is a modulation term enabling the TS (4.25)- (4.26) to be an orientation-preserving homeomorphism with respect to changes in the initial and goal positions y_0 and g (Devaney, 1976; Ijspeert et al., 2013). The weight coefficients ω_i

in (4.29) are calculated by Locally Weighted Regression (LWR) (Atkeson, 1989) as follows:

$$\omega_i = \operatorname{argmin}_{\omega_i} \sum_{t=0}^T \phi_i(s_t) [f_{r,t} - \omega_i \xi_t]^2, \quad (4.31)$$

where $\xi_t = s_t(g - y_0)$ and the nominal forcing term $f_{r,t}$ is computed from (4.28) by using the reference position, velocity, and acceleration trajectories $r_t, \dot{r}_t$ and $\ddot{r}_t$:

$$f_{r,t} = \tau^2 \ddot{r}_t + \alpha \tau \dot{r}_t + \alpha \beta (r_t - g_r), \quad (4.32)$$

where $g_r = r_T$. The parameter T is the total duration of the reference trajectory r_t , and the time constant τ is commonly assumed to be unitary.

DMP represent a simple yet effective encoding technique, featuring useful properties such as stability, time scaling, spatial modulation, and compatibility with additional coupling terms (Ijspeert et al., 2013; Pastor et al., 2009; Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023). Thanks to their compact and versatile formulation, DMP were successfully applied to various tasks, such as tennis swing (Ijspeert et al., 2002), bi-manual manipulation (Nemec et al., 2014), peg-in-hole (Sidiropoulos & Doulgeri, 2021), among others (Lauretti et al., 2017; Nemec et al., 2013; Ruckert & d'Avella, 2013; Sidiropoulos & Doulgeri, 2022).

4.3.1 Quaternion-based representation for DMP

The classical DMP formulation described by equations (4.25)- (4.27) can be repeated for each degree of freedom (DoF), as per (4.43). Indeed this can also be extended for joint trajectories, with the CS in (4.27) serving as synchronizing variable for all the DMP trajectories. While this is possible, like for Cartesian trajectories, since each DoF evolves independently for the others, orientation elements like unit quaternions or orientation matrices should be treated differently. This comes from the fact that orientation trajectories are constrained, since they should evolve along an orientation manifold over which orientation properties are preserved (Sola, 2017). While during this work, the orientation of the robot is always kept fixed, for the sake of completeness it is worth to add a brief introduction to the orientation formulation of DMP. In particular the quaternion DMP is introduced, but also the rotation matrix DMP can be found in literature (Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023; Ude et al., 2014).

The quaternion DMP was firstly introduced in (Pastor et al., 2011), but its formulation was further refined in (F. J. Abu-Dakka et al., 2015) in order to account also for the geometry of $\mathcal{SO}^3$. Unit quaternion $q = q_0 + \mathbf{q} \in \mathcal{S}^3$, with

$q_0 \in \mathbb{R}$ and $\mathbf{q} \in \mathbb{R}^3$ provides a representation of the orientation of the robot's end-effector, where $\mathcal{S}^3$ is a unit sphere in $\mathbb{R}^4$ (for a short introduction over quaternions, please refer to Appendix B). In particular, from (F. J. Abu-Dakka et al., 2015), the DMP equations for the TS in (4.25)- (4.26) can be written on their quaternion counterpart as:

$$\tau \dot{\gamma}_t = \alpha [\beta 2 \text{Log}^q(q_d \otimes q_t^* - \gamma_t)] + f_q(s_t), \quad (4.33)$$

$$\tau \dot{q}_t = \frac{1}{2} \gamma_t \otimes q, \quad (4.34)$$

where $q_d \in \mathcal{S}^3$ is the desired goal orientation, $\gamma \in \mathbb{R}^3$ is the scaled angular velocity ω , q^* is the conjugate of q and $\otimes$ is the quaternion product operator. Finally, in (4.33), the $\text{Log}^q(\cdot) : \mathcal{S}^3 \rightarrow \mathbb{R}^3$ function is computed as

$$\text{Log}^q(q) = \begin{cases} \arccos(q_0) \frac{\mathbf{q}}{\|\mathbf{q}\|}, & \mathbf{q} \neq \mathbf{0}, \\ [0 \ 0 \ 0]^\top, & \text{otherwise.} \end{cases} \quad (4.35)$$

Note that, during execution, (4.34) can be integrated as

$$q(t + \delta_t) = \text{Exp}^q\left(\frac{\delta_t}{\tau} \gamma_t\right) \otimes q_t, \quad (4.36)$$

with δ_t being a small positive constant. The function $\text{Exp}^q(\cdot) : \mathbb{R}^3 \rightarrow \mathcal{S}^3$ function is computed as

$$\text{Exp}^q(\omega) = \begin{cases} \cos(\|\omega\|) + \sin(\|\omega\|) \frac{\omega}{\|\omega\|}, & \omega \neq \mathbf{0}, \\ 1 + [0 \ 0 \ 0]^\top, & \text{otherwise.} \end{cases} \quad (4.37)$$

Both mappings in (4.35) and (4.37) are continuously differentiable and inverse to each other if the input domain of $\text{Log}^q(\cdot)$ is restricted to $\mathcal{S}^3 - -1 + [0 \ 0 \ 0]^\top$, while the input domain for the function $\text{Exp}^q(\cdot)$ should impose $\|\omega\| < \pi$ (F. J. Abu-Dakka et al., 2015).

4.3.2 A reversible DMP formulation

The state trajectories of a dynamic physical system naturally evolve with time. However, if the considered dynamic model does not represent a physical system, the system trajectories may be desired to evolve according to a different timing law. For instance, scaling the original time by a suitable factor can enhance flexibility, offering advantageous properties like feedback linearizability (Kiss &

Szadeczky-Kardoss, 2008; Sampei & Furuta, 1986).

In the specific case of DMP, the timing law is typically governed by the Canonical System (CS) in (4.27), where the variation of the phase variable s_t can be modulated by properly tuning the parameter τ , thus giving a time-scalable Transformation System (TS) in (4.25)- (4.26) and maintaining the topological equivalence of the output trajectories (Ijspeert et al., 2013; Sidiropoulos & Doulgeri, 2021, 2022). Choosing the CS in (4.27) to exhibit an exponential decay ensures that the contribute of $f(s_t)$ in (4.25) fades out when reaching the goal point g (Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023). However, this solution inevitably leads to very high weights in (4.31) when close to the final part of the reference trajectory (Samant et al., 2016), which can be undesirable when combining multiple DMP together (Lauretti et al., 2017; Pastor et al., 2009; Ruckert & d’Avella, 2013). To cope with this issue, other types of CS were adopted, such as piece-wise linear (Samant et al., 2016) or sigmoidal-like phase profiles (Kulvicius et al., 2011).

The key aspect of having a timing law governed by the CS in (4.27) is that the evolution of the trajectory y_t in the TS (4.25)- (4.26) can be effectively slowed down or speeded up by tuning the parameter τ , without introducing any geometric variation in the resulting trajectory y_t of the DMP (Devaney, 1976; Ijspeert et al., 2013). In this case, an issue arises when the trajectory is required to momentarily stop during its reproduction, as this would imply $\tau \rightarrow \infty$. This problem was noted in (Ijspeert et al., 2002; Nemec et al., 2019), then further analyzed in (Nemec et al., 2013).

An important aspect when dealing with timing laws is the possibility of letting the trajectory move backward and forward between the initial and goal positions, y_0 and g respectively, while preserving the trajectory topology, i.e. while allowing $\dot{s}_t$ in (4.27) to change sign (Sidiropoulos & Doulgeri, 2021; Verscheure et al., 2009). This feature is called phase reversibility. Even though it offers many benefits in various scenarios (Nemec et al., 2019), the classical DMP formulation in (4.25)- (4.27) does not fulfill the phase reversibility requirement (Nemec, Ude, et al., 2016). In fact, the restriction of having a strictly monotonic $\dot{s}_t$ implies that the new timing law neither allows the trajectory to be reproduced backward nor allows it to be stopped (Sampei & Furuta, 1986).

The problem of reversibility was considered in (Nemec, Zlajpah, et al., 2018) and further investigated in (Iturrate et al., 2019; Sidiropoulos & Doulgeri, 2021), where it was shown that having a reversible phase can affect the DMP global stability property (Nemec et al., 2019). In (Iturrate et al., 2019), the authors

formulate the design requirements for guaranteeing phase reversibility in DMP, which is a desirable property in some DMP formulation (Escarabajal et al., 2023; Koutras & Doulgeri, 2020b; Sidiropoulos & Doulgeri, 2021, 2022; Sidiropoulos et al., 2023). Firstly, a reversible DMP should maintain global stability properties as the original one (Iturrate et al., 2019), meaning that y_0 must be an attractor point when moving backward, whereas g must be an attractor point when moving forward. Secondly, the trajectory topology must be preserved when moving forward and backward. Finally, it would be desirable to adopt the same forcing term when progressing along the intended path.

Whenever a new DMP formulation is proposed, it is also important to verify that global asymptotic stability is guaranteed. In (Iturrate et al., 2019), a candidate for such a system is given by the Logistic Differential Equation (LDE) $\tau \dot{y} = \alpha(y - y_0)(g - y)$. However, it can be demonstrated that the region of global stability of the LDE is restricted to $y \in (y_0, g)$ with $y_0 < g$ (Wiggins et al., 2003). Consequently, two separate forcing terms are learned to switch along the desired curve whenever the sign of $\dot{s}$ changes (Zhao et al., 2023).

When dealing with phase reversibility, a considerable step forward was provided in (Sidiropoulos & Doulgeri, 2021). In this work, the authors present a new formulation of the original DMP satisfying all phase reversibility requirements by using a unique forcing term $f(s_t)$, based on a parameterization of the reference trajectory r_t and of its time-derivatives (Jankowski et al., 2023; Verschueren et al., 2009). In this case, there is no need to limit the forcing term $f(s_t)$ by multiplying it by s_t , as the terms $y_t, \dot{y}_t, \ddot{y}_t$ are bounded by construction and the derivatives tend to zero when the system approaches the initial/goal positions (Sidiropoulos et al., 2023). Additionally, backward reproduction is achieved without impacting the DMP stability by simply reversing the sign of the phase velocity $\dot{s}_t$ (Sidiropoulos & Doulgeri, 2022).

The reversible DMP formulation in (Sidiropoulos & Doulgeri, 2021) is now introduced and its structure illustrated in Fig. 4.4. The forcing term $f(s_t)$ in (4.28) is computed as follows:

$$f(s_t) = \eta f^*(s_t), \quad \text{where} \quad \eta = \frac{g - y(s_0)}{g_r - \mu_0} \quad (4.38)$$

is a scaling factor depending on the initial and goal positions μ_0 and $g_r = \mu(s_T)$ of the reference trajectory, as well as on the initial and desired goal positions $y(s_0)$ and g . The objective of the scaling factor η is to preserve the topology of the curve (Devaney, 1976; Koutras & Doulgeri, 2020b; Sidiropoulos & Doulgeri,

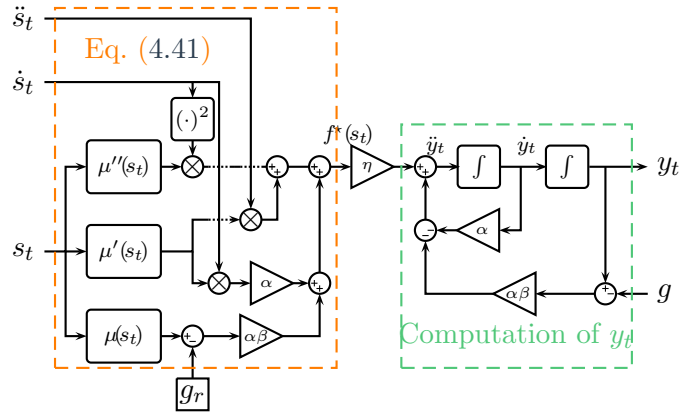


Figure 4.4: Block-scheme representation of the TS (4.28) by incorporating the forcing term (4.41), which ensures the independence from the phase variable s_t .

2022).

In the classical DMP, the forcing term $f^*(s)$ is obtained as in (4.29)- (4.9) and (4.31), where the nominal forcing term $f_{r,t}$ is computed as a linear combination of the demonstrated position, velocity, and acceleration profiles as in (4.32). In our case, the forcing term $f^*(s)$ in (4.38) is obtained using the linear combination (4.32) by replacing the demonstrated position, velocity, and acceleration profiles with the corresponding parameterizations $\mu(s)$, $\dot{\mu}(s)$ and $\ddot{\mu}(s)$. Specifically, given the sequence of demonstrated positions $r(s)$, $s = 0, \dots, s_T$, the coefficients ω_i of the parametric function

$$\mu(s_t) = \sum_{i=1}^N \omega_i \phi_i(s_t) \quad (4.39)$$

are determined by minimizing the cost function

$$J = \sum_{t=0}^T \|r_t - \omega_i \phi_i(s_t)\|^2, \quad (4.40)$$

with the boundary conditions

$$\sum_{i=1}^N \omega_i \phi_{i,0} = r_0, \quad \text{and} \quad \sum_{i=1}^N \omega_i \phi_i(s_T) = r(s_T).$$

The radial basis functions $\phi_i(\cdot)$ are defined in (4.30), and the timing law $s = s_t : [0, T] \rightarrow [0, s_T]$ is a monotonic function of class C^1 such that $\dot{s}_0 = \dot{s}_T = 0$ and $\ddot{s}_0 = \ddot{s}_T = 0$. The derivatives $\dot{\mu}(s_t)$ and $\ddot{\mu}(s_t)$ of the parameterized trajectory $\mu(s_t)$ can be computed as in (3.3). Therefore, with the proposed formulation, it

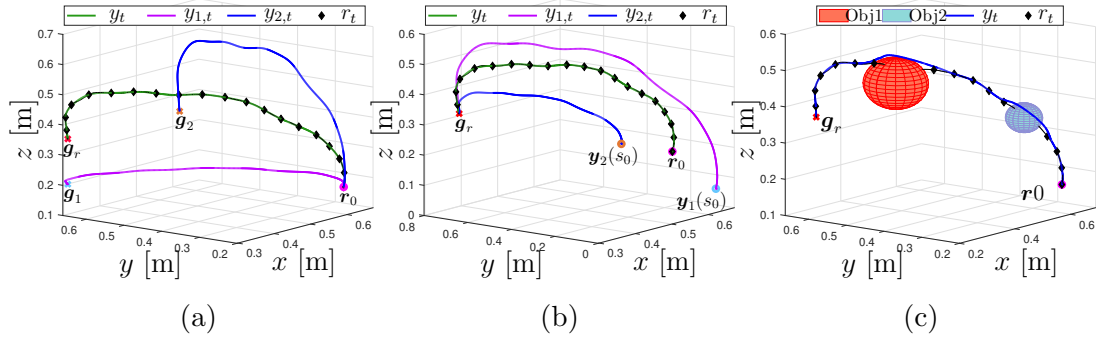


Figure 4.5: Verification that the DMP formulation in Fig. 4.4 maintains the characteristic properties of the original one: spatial adaptation with respect to (a) different target goals, (b) different initial positions and (c) presence of obstacles.

is not necessary to know the derivatives of the reference trajectory, but only the position profile r_t is sufficient. Using (4.38), (4.39) and (3.3), the forcing term $f(s_t)$ in (4.28) can be written as:

$$f(s_t) = \eta \left[\mu''(s_t) \dot{s}_t^2 + \mu'(s_t) \ddot{s}_t + \alpha \mu'(s_t) \dot{s}_t + \alpha \beta (\mu(s_t) - g_r) \right]. \quad (4.41)$$

By distributing the scaling factor η across all the terms on the right side of (4.41), $f(s_t)$ can be expressed as a linear combination of

$$\mu_\eta(s_t) = \eta \mu(s_t) \quad (4.42)$$

and its derivatives. The complete schematic view of the TS (4.25)- (4.26) with the forcing term (4.41) is reported in Fig. 4.4, and enables the following features (Sidiropoulos & Doulgeri, 2021; Sidiropoulos et al., 2023): time modulation by varying the evolution of the phase variable s_t ; phase reversibility allowing forward and backward reproduction of the trajectory, when $\dot{s}_t > 0$ and $\dot{s}_t < 0$, respectively; spatial modulation of the curve varying the initial position $y(s_0)$ and the goal g of the task.

The TS can then be generalized to the multidimensional case. If the considered application requires the robot position only to be considered, a Cartesian representation of the TS can be employed:

$$\ddot{\mathbf{y}}(t) + \mathbf{A}\dot{\mathbf{y}}(t) + \mathbf{AB}(\mathbf{y}(t) - \mathbf{g}) = \mathbf{E}f^*(s_t), \quad (4.43)$$

where $\mathbf{A} = \text{diag}(\alpha_x, \alpha_y, \alpha_z)$, $\mathbf{B} = \text{diag}(\beta_x, \beta_y, \beta_z)$, $\mathbf{E} = \text{diag}(\eta_x, \eta_y, \eta_z)$, and

$$\mathbf{y}(t) = \begin{bmatrix} y_{x,t} \\ y_{y,t} \\ y_{z,t} \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix}, \quad \mathbf{f}^*(s) = \begin{bmatrix} f_x^*(s_t) \\ f_y^*(s_t) \\ f_z^*(s_t) \end{bmatrix}. \quad (4.44)$$

Note that Eqs. (4.43)- (4.44) can also be generalized to consider the end-effector orientation by adopting, for example, an Euler-angle or a quaternion-based representation (Ijspeert et al., 2013). The resulting TS in Fig. 4.4 still maintains the characteristic properties of the original formulation, such as spatial adaptation with respect to different goal/initial position and with respect to the presence of obstacles by adding proper coupling terms (Dragan et al., 2015; Ijspeert et al., 2013). Figure 4.5 shows an example of application of such properties: Fig. 4.5a and Fig. 4.5b show the possibility of changing the goals and initial positions $\mathbf{g}$ and $\mathbf{y}(s_0)$ in (4.38) with respect to the reference ones $\mathbf{g}$ and $\mathbf{r}_0$, while Fig. 4.5c shows how the introduction of coupling terms, in this case repulsive potential fields (Gams et al., 2016; Sidiropoulos et al., 2023), allows to avoid obstacles along the path.

4.4 Linear Quadratic Regulator

Another important execution technique is represented by the well-known Linear Quadratic Regulator (LQR) algorithm (Calinon, 2023b). As it will be shown in the next sections, the LQR algorithm, especially its tracking version, exhibits similarities with Dynamic Movement Primitives (DMP). Here, its classical formulation is explained, with slight modifications introduced to include a parameterized reference curve $\boldsymbol{\mu}(s)$, which can be obtained from a LfD approach.

We consider an augmented state $\mathbf{y}_t = [\mathbf{y}_t^\top, s_t]^\top \in \mathbb{R}^D$, with $\mathbf{y}_t^\top \in \mathbb{R}^d$, $D = d + 1$, the tracking targets $\boldsymbol{\mu}(s_t) = \boldsymbol{\phi}(s_t)^\top \boldsymbol{\omega}$ defined as a superposition of basis functions, and the control command $\mathbf{u}_t = [\mathbf{u}_t^y(s_t)^\top, u_t^s]^\top$. The phase variable s_t is preferably taken such that $s_0 = 0$ and $s_T = 1$ ¹. We define the cost as

$$c(\mathbf{y}_t, \mathbf{u}_t) = \sum_{t=1}^T \mathbf{f}(\mathbf{y}_t)^\top \mathbf{Q}_t^y \mathbf{f}(\mathbf{y}_t) + \mathbf{u}_t^\top \mathbf{R}_t \mathbf{u}_t + \mathbf{h}(s_t)^\top \mathbf{Q}_t^s \mathbf{h}(s_t) \quad (4.45)$$

¹The interval $s \in [0, 1]$ is common but not strict. If s is the arc-length parameter, obtained for instance from the Spatial Sampling algorithm in 3.2, one can use $s_T = L$, with L being the total length of the curve.

with

$$\mathbf{f}(\mathbf{y}_t) = \mathbf{y}_t - \boldsymbol{\mu}(s_t), \quad \mathbf{h}(s_t) = s_T - 1 \quad (4.46)$$

subject to the dynamical system

$$\begin{aligned} s_{t+1} &= s_t + u_t^s \Delta t \\ \mathbf{y}_{t+1} &= \mathbf{y}_t + \mathbf{u}_t^y(s_t) u_t^s \Delta t \end{aligned} \quad (4.47)$$

where $\mathbf{Q}_t^y$, $\mathbf{Q}_t^s$ and $\mathbf{R}_t$ in (4.45) are the weight matrices trading off task and control costs. The problem stated in (4.45), because of the term $\mathbf{f}(\mathbf{y}_t)$, does not converge as the controller tries to reduce $\mathbf{f}$ either by moving the state $\mathbf{y}_t$ closer to the target (what we do want) or by adjusting the phase variable s_t s.t. the reference $\boldsymbol{\mu}(s_t)$ is closer to the state. The result is that LQR does not converge to a feasible solution, not following properly the desired reference path and, simultaneously, not generating a correct phase profile.

By considering the system's dynamics described in (4.47), we can avoid the coupling issue by split the problem into a 2-step LQR solving first

$$c_s(s_t, u_t^s) = \min_{u_t^s} \mathbf{h}^\top \mathbf{Q}_t^s \mathbf{h} + u_t^{s\top} \mathbf{R}_t^s u_t^s \quad \text{s.t.} \quad s_{t+1} = s_t + u_t^s \Delta t \quad (4.48)$$

which leads to a solution $\hat{u}_t^s$ that can subsequently plugged into the tracking problem

$$\begin{aligned} c_y(\mathbf{y}_t, \mathbf{u}_t^y(s_t)) &= \min_{\mathbf{u}_t^y(s_t)} \mathbf{f}(\mathbf{y}_t)^\top \mathbf{Q}_t^y \mathbf{f}(\mathbf{y}_t) + \mathbf{u}_t^y(s_t)^\top \mathbf{R}_t^y \mathbf{u}_t^y(s_t) \\ &\text{subject to } \mathbf{y}_{t+1} = \mathbf{y}_t + \mathbf{u}_t^y(s_t) \hat{u}_t^s \Delta t. \end{aligned} \quad (4.49)$$

4.4.1 Batch formulation of two-step LQR

Splitting (4.45) into two sub-optimization problems is actually quite similar to most encoding approaches. For instance, all DMP literature is based on dynamical systems that take as input a phase variable (Ijspeert et al., 2002). In the same way here, the phase is preliminary computed as solution of (4.48), then its profile is used as a constant input to (4.49), where $\mathbf{u}$ becomes the only decision variable.

We thus consider (4.48). LQR performs at each iteration a Taylor expansion of both the cost function and the dynamical system around a point $(\hat{\mathbf{y}}_t, \hat{\mathbf{u}}_t)$ in order to linearize the problem. It can be demonstrated (Calinon, 2023b), considering

the whole trajectory from $t = 0$ to $t = T$, that (4.48) can be linearized as follows

$$\Delta c_s(\Delta \mathbf{s}, \Delta \mathbf{u}_s) \approx 2\Delta \mathbf{s}^\top \mathbf{J}_s^\top \mathbf{Q}_s \mathbf{h} + 2\Delta \mathbf{u}_s^\top \mathbf{R}_s \mathbf{u}_s + \Delta \mathbf{s}^\top \mathbf{J}_s^\top \mathbf{Q}_s \mathbf{J}_s \Delta \mathbf{s} + \Delta \mathbf{u}_s^\top \mathbf{R}_s \Delta \mathbf{u}_s$$

subject to $\Delta \mathbf{s} = \mathbf{S}_{us} \Delta \mathbf{u}_s$

(4.50)

being $\mathbf{J}_s$ the Jacobian matrix defined as $\frac{\partial \mathbf{h}}{\partial \mathbf{s}}$, $\mathbf{Q}_s = \text{blockdiag}(\mathbf{Q}_1^s, \mathbf{Q}_1^s, \dots, \mathbf{Q}_T^s)$, $\mathbf{R}_s = \text{blockdiag}(\mathbf{R}_1^s, \mathbf{R}_1^s, \dots, \mathbf{R}_T^s)$ and $\mathbf{S}_{us}$ being the control transfer matrix at trajectory level². By further differentiating the cost function c_s by u_s and equating to zero we obtain:

$$\Delta \hat{\mathbf{u}}_s = (\mathbf{S}_{us}^\top \mathbf{J}_s^\top \mathbf{Q}_s \mathbf{J}_s \mathbf{S}_{us} + \mathbf{R}_s)^{-1} (-\mathbf{S}_{us}^\top \mathbf{J}_s^\top \mathbf{Q}_s \mathbf{h} - \mathbf{R}_s \mathbf{u}_s) \quad (4.51)$$

The next step will be finding the optimal control $\hat{\mathbf{u}}_s^y$ in order to solve (4.49), including the optimal phase control command $\hat{\mathbf{u}}_s$ found in the previous step. By linearizing with Taylor expansion, the cost $c_y(\mathbf{y}_t, \mathbf{u}_t^y(s_t))$ at trajectory level can be approximated as

$$\Delta c_y(\Delta \mathbf{y}, \Delta \mathbf{u}_y) \approx 2\Delta \mathbf{y}^\top \mathbf{J}_y^\top \mathbf{Q}_y \mathbf{f} + 2\Delta \mathbf{u}_y^\top \mathbf{R}_y \mathbf{u}_y + \Delta \mathbf{y}^\top \mathbf{J}_y^\top \mathbf{Q}_y \mathbf{J}_y \Delta \mathbf{y}_t + \Delta \mathbf{u}_y^\top \mathbf{R}_y \Delta \mathbf{u}_y$$

subject to $\Delta \mathbf{y} = \mathbf{S}_{uy}(\Delta \mathbf{u}_y + \Delta \mathbf{u}'_y)$,

(4.52)

being $\mathbf{J}_y$ the Jacobian matrix defined as $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$, $\mathbf{Q}_y = \text{blockdiag}(\mathbf{Q}_0^y, \dots, \mathbf{Q}_1^y)$ and $\mathbf{R}_y = \text{blockdiag}(\mathbf{R}_0^y, \dots, \mathbf{R}_T^y)$. In particular, as we are considering a two-step LQR, the evolution matrix $\mathbf{S}_{uy}$ embeds the optimal phase command as, for every instant, include the term $\hat{u}_t^s \Delta t$.

Before moving on it is opportune to further discuss the linearization of the dynamical system describing the evolution of the state $\mathbf{y}_t$. Let's simplify the dynamical system as $\mathbf{y}_{t+1} = \mathbf{d}(\mathbf{y}_t, \mathbf{u}_t)$, its first order Taylor expansion around point $(\hat{\mathbf{y}}_t, \hat{\mathbf{u}}_t)$ then becomes

$$\Delta \mathbf{y}_{t+1} \approx \frac{\partial \mathbf{d}}{\partial \mathbf{y}_t} \Delta \mathbf{y}_t + \frac{\partial \mathbf{d}}{\partial \mathbf{u}_t} \Delta \mathbf{u}_t, \quad \text{with } \Delta \mathbf{y}_t = \mathbf{y}_t - \hat{\mathbf{y}}_t, \quad \Delta \mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t \quad (4.53)$$

Now calling the evolution of the state $\mathbf{y}_t$ described in (4.47), we can apply to the latter (4.53), substituting the result of (4.51) and obtain

$$\Delta \mathbf{y}_{t+1} \approx \Delta \mathbf{y}_t + \hat{u}_t^s \Delta t \frac{\partial \mathbf{u}_t^y(s_t)}{\partial s} \Delta s_t + \hat{u}_t^s \Delta t \Delta \mathbf{u}_t^y(s_t) \quad (4.54)$$

²For a detailed explanation on how state and control transfer matrices at trajectory level are computed, please refer to Appendix D.1.

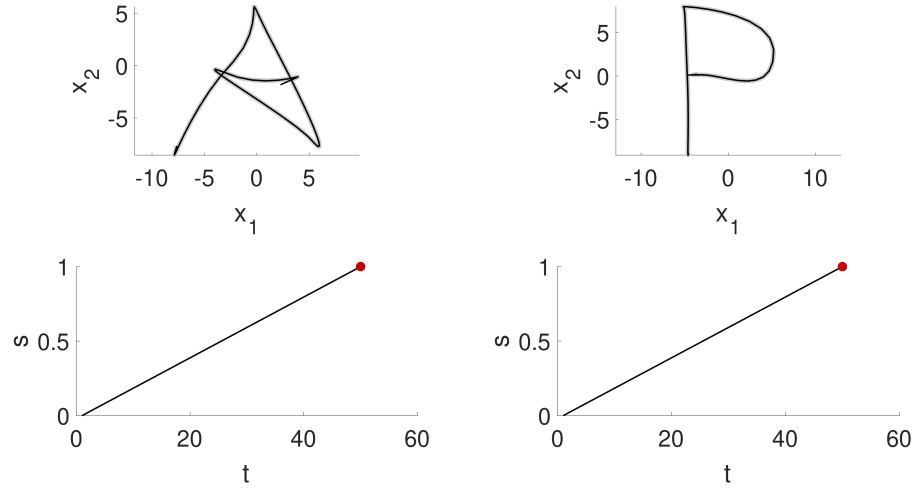


Figure 4.6: Performing 2-step LQR on Letters' dataset. Each letter was used to create the desired path reference as $\boldsymbol{\mu}(s_t) = \boldsymbol{\phi}(s_t)^\top \boldsymbol{\omega}$. By controlling the phase variable s_t we are thus able to move the reference along the curve and track it properly with the state $\mathbf{y}_t$. For every letter the optimal control command $\mathbf{u}_t$ was found in less than 5 iterations.

This result can be better visualized if we rewrite (4.54) as

$$\frac{\Delta \mathbf{y}_{t+1} - \Delta \mathbf{y}_t}{\Delta t} = (\Delta \mathbf{u}_t^{\mathbf{y}}(s_t) + \frac{\partial \mathbf{u}_t^{\mathbf{y}}(s_t)}{\partial s_t} \Delta s_t) \hat{u}_t^s \quad (4.55)$$

showing that the control command $\mathbf{u}_t^{\mathbf{y}}(s_t)$ approximates to the sum of the velocity term $\Delta \mathbf{u}_t^{\mathbf{y}}(s_t)$ in s and the acceleration term $\mathbf{u}_t^{\mathbf{y}'}(s_t) = \frac{\partial \mathbf{u}_t^{\mathbf{y}}(s_t)}{\partial s_t} \Delta s_t$ is single integrated in s . Thus, by multiplying by the phase velocity control $\hat{u}_t^s$ we obtain the velocity command for the tracking problem as shown in (4.55).

Again, by differentiating the cost function $c_{\mathbf{y}}$ w.r.t. $(\Delta \mathbf{u}_{\mathbf{y}}, \Delta \mathbf{u}_{\mathbf{y}}')$ we obtain the control commands at trajectory level

$$\begin{aligned} \Delta \hat{\mathbf{u}}_{\mathbf{y}} &= \mathbf{u}_{\mathbf{y}} \\ \Delta \hat{\mathbf{u}}_{\mathbf{y}}' &= (\mathbf{S}_{\mathbf{u}\mathbf{y}}^\top \mathbf{H}_{\mathbf{y}} \mathbf{S}_{\mathbf{u}\mathbf{y}})^{-1} (-\mathbf{S}_{\mathbf{u}\mathbf{y}}^\top \mathbf{g}_{\mathbf{y}} - \mathbf{S}_{\mathbf{u}\mathbf{y}}^\top \mathbf{H}_{\mathbf{y}} \mathbf{S}_{\mathbf{u}\mathbf{y}} \Delta \mathbf{u}_{\mathbf{y}}) \end{aligned} \quad (4.56)$$

where we used the gradient $\mathbf{g}_{\mathbf{y}} = \mathbf{J}_{\mathbf{y}}^\top \mathbf{Q}_{\mathbf{y}} \mathbf{f}$ and the Hessian $\mathbf{H}_{\mathbf{y}} = \mathbf{J}_{\mathbf{y}}^\top \mathbf{Q}_{\mathbf{y}} \mathbf{J}_{\mathbf{y}}$.

4.4.2 Batch formulation of two-step LQR with Control Primitives

We now reformulate the problem in order to use control primitives (Calinon, 2023b), such that the control command on the tracking task can now take the form $\mathbf{u}_t^y(s_t) = \phi(s_t)\mathbf{w}$, i.e. $\mathbf{u}_t^y(s_t)$ is a sum of N basis functions weighted by the vector $\mathbf{w}$. The problem in (4.49) is thus reformulated to find the best vector $\hat{\mathbf{w}}$ that minimizes the tracking cost $\mathbf{c}_y$. To do that we start considering the dynamical system

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \phi(s_t)\mathbf{w}_y(s_t)\hat{u}_t^s\Delta t \quad (4.57)$$

where now the control variable is directly the phase s_t . Note that now the control $\mathbf{u}_y(s_t)$ is written as a superposition of basis function, including the possibility to extended the control command also to the state space, thus having a more general framework. Moreover, by considering different types of basis functions, $\mathbf{u}_y$ could be designed to feature useful characteristics as, for example, continuity of derivatives or local modulations (see Section 4.1).

At this point we stick to the same procedure used in the previous paragraphs. First we aim at linearize in a small interval around the working point $(\hat{\mathbf{y}}_t, \hat{s}_t)$ the minimization problem in (4.49) which leads, at trajectory level, to

$$\begin{aligned} \Delta c_y(\Delta \mathbf{y}, \Delta \mathbf{s}) &\approx 2\Delta \mathbf{y}^\top \mathbf{J}_y^\top \mathbf{Q}_y \mathbf{f} + 2\Delta \mathbf{s}^\top \mathbf{R}_y \mathbf{s} + \Delta \mathbf{y}^\top \mathbf{J}_y^\top \mathbf{Q}_y \mathbf{J}_y \Delta \mathbf{y}_t + \Delta \mathbf{s}^\top \mathbf{R}_y \Delta \mathbf{s} \\ s.t. \quad \Delta \mathbf{y} &= \mathbf{S}_{uy} \Psi'(s) \Delta s \mathbf{w}_y. \end{aligned} \quad (4.58)$$

As we are considering a two-step LQR problem, once minimizing the cost function Δc_y the optimal value of the phase is already known from the solution of (4.52), therefore we can minimize the cost on (4.58) by equating to zero its derivative w.r.t. $\mathbf{w}_y$ which gives

$$\hat{\mathbf{w}}_y = (\mathbf{S}_{uy}^\top \mathbf{H}_y \mathbf{S}_{uy} \Psi'(s) \Delta \mathbf{s})^\dagger (-\mathbf{S}_{uy}^\top \mathbf{g}_y), \quad (4.59)$$

using $\Delta \mathbf{s} = \mathbf{S}_{us} \Delta \mathbf{u}_s$ and denoting with $\dagger$ the pseudo-inverse operator. Again, note that $\mathbf{S}_{uy}$ is computed such that it embeds the optimal control $\Delta \hat{u}_t^s$, calculated in the first step of the LQR algorithm.

4.4.3 Recursive formulation of two-step LQR

As an alternative to batch computation of optimal control, the recursive solution of LQR allows to provide the controller with feedback gains to cope with online perturbation of the state. The problem description is here simplified, we encourage to read (Calinon, 2023b) for further details for a short introduction for the nomenclature that will be used in the following paragraphs.

The recursive formulations for a generic state $\mathbf{y}$ and a generic control $\mathbf{u}$ basically consists in computing at every single iteration the optimal $\hat{\mathbf{u}}_t$ by proceeding backwards in time, i.e. solving

$$\hat{v}_t(\Delta \mathbf{y}_t) = \min_{\Delta \mathbf{u}_t} \underbrace{c_t(\Delta \mathbf{y}_t, \Delta \mathbf{u}_t) + \hat{v}_{t+1}(\Delta \mathbf{y}_{t+1})}_{q_t(\Delta \mathbf{y}_t, \Delta \mathbf{u}_t)} \quad (4.60)$$

where $\hat{v}_t(\Delta \mathbf{y}_t)$ is called *value-function* and describes the partial cumulative cost at instant t w.r.t. the optimal control $\hat{\mathbf{u}}_t$. Starting at time step T , it is possible back-propagate the values of $\hat{v}_{t+1}$ in (4.60) as we know that $\hat{v}_T$ does not depend on the control commands. In particular, $\hat{v}_T$ takes the quadratic form

$$\hat{v}_T = \mathbf{y}_T^\top V_T \mathbf{y}_T \quad (4.61)$$

and it can be demonstrated that even $\hat{v}_{T-1}$ can be reduced to the same quadratic form, allowing the recursive computation from $t = T - 1$ to $t = 1$. Therefore we obtain

$$\hat{v}_{t+1} = \begin{bmatrix} \mathbf{1} \\ \Delta \mathbf{y}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{0} & \mathbf{v}_{\mathbf{y},t+1}^\top \\ \mathbf{v}_{\mathbf{y},t+1} & \mathbf{V}_{\mathbf{yy},t+1} \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \Delta \mathbf{y}_{t+1} \end{bmatrix} \quad (4.62)$$

therefore write (4.60) as

$$\begin{aligned} \hat{v}_t = \min_{\Delta \mathbf{u}_t} & \begin{bmatrix} \mathbf{1} \\ \Delta \mathbf{y}_t \\ \Delta \mathbf{u}_t \end{bmatrix}^\top \begin{bmatrix} \mathbf{0} & \mathbf{g}_{\mathbf{y},t}^\top & \mathbf{g}_{\mathbf{u},t}^\top \\ \mathbf{g}_{\mathbf{y},t} & \mathbf{H}_{\mathbf{yy},t} & \mathbf{H}_{\mathbf{uy},t}^\top \\ \mathbf{g}_{\mathbf{y},t} & \mathbf{H}_{\mathbf{uy},t} & \mathbf{H}_{\mathbf{uu},t}^\top \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \Delta \mathbf{y}_t \\ \Delta \mathbf{u}_t \end{bmatrix} \\ & + \begin{bmatrix} \mathbf{1} \\ \Delta \mathbf{y}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{0} & \mathbf{v}_{\mathbf{y},t+1}^\top \\ \mathbf{v}_{\mathbf{y},t+1} & \mathbf{V}_{\mathbf{yy},t+1} \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \Delta \mathbf{y}_{t+1} \end{bmatrix} \end{aligned} \quad (4.63)$$

Then, by substituting the linearized dynamics of the system $\Delta \mathbf{y}_{t+1} = \mathbf{A}_t \Delta \mathbf{y}_t +$

$B_t \Delta \mathbf{u}_t$ into (4.63) the value of $\hat{v}_t$ can be rewritten as

$$\hat{v}_t = \min_{\Delta \mathbf{u}_t} \begin{bmatrix} 1 \\ \Delta \mathbf{y}_t \\ \Delta \mathbf{u}_t \end{bmatrix}^\top \begin{bmatrix} 0 & \mathbf{q}_{y,t}^\top & \mathbf{q}_{u,t}^\top \\ \mathbf{q}_{y,t} & Q_{yy,t} & Q_{uy,t}^\top \\ \mathbf{q}_{y,t} & Q_{uy,t} & Q_{uu,t}^\top \end{bmatrix} \begin{bmatrix} 1 \\ \Delta \mathbf{y}_t \\ \Delta \mathbf{u}_t \end{bmatrix} \quad (4.64)$$

where $\begin{cases} \mathbf{q}_{y,t} = \mathbf{g}_{y,t} + \mathbf{A}_t^\top \mathbf{v}_{y,t+1}, \\ \mathbf{q}_{u,t} = \mathbf{g}_{u,t} + \mathbf{B}_t^\top \mathbf{v}_{y,t+1}, \\ Q_{yy,t} \approx H_{yy,t} + \mathbf{A}_t^\top V_{yy,t} \mathbf{A}_t, \\ Q_{uu,t} \approx H_{uu,t} + \mathbf{B}_t^\top V_{yy,t} \mathbf{B}_t, \\ Q_{uy,t} \approx H_{uy,t} + \mathbf{B}_t^\top V_{yy,t} \mathbf{A}_t, \end{cases}$

with gradients $\mathbf{g}_{y,t}$, $\mathbf{g}_{u,t}$, $\mathbf{v}_{y,t}$, $\mathbf{q}_{y,t}$, $\mathbf{q}_{u,t}$, Jacobian matrices $\mathbf{A}_t$, $\mathbf{B}_t$ and Hessian matrices $H_{yy,t}$, $H_{uu,t}$, $H_{uy,t}$, $V_{yy,t}$, $Q_{yy,t}$, $Q_{uu,t}$, $Q_{uy,t}$. Then differentiating and equating to zero (4.64) w.r.t. $\Delta \mathbf{u}_t$ enables to obtain the optimal controller

$$\Delta \mathbf{u}_t = \mathbf{k}_t + \mathbf{K}_t \Delta \mathbf{y}_t, \quad \text{with} \quad \begin{cases} \mathbf{k}_t = -Q_{uu,t}^{-1} \mathbf{q}_{u,t}, \\ \mathbf{K}_t = -Q_{uu,t}^{-1} Q_{ux,t} \end{cases} \quad (4.65)$$

and getting the recursive updates

$$\begin{aligned} \mathbf{v}_{y,t} &= \mathbf{q}_{y,t} - Q_{uy,t}^\top Q_{uy,t}^{-1} \mathbf{q}_{u,t}, \\ V_{yy,t} &= Q_{yy,t} - Q_{uy,t}^\top Q_{uu,t}^{-1} Q_{uy,t} \end{aligned} \quad (4.66)$$

Thus, after convergence of the algorithm, by using (4.65) it is possible to update the controller at each time step as

$$\mathbf{u}_t = \hat{\mathbf{u}}_t + \mathbf{K}_t (\hat{\mathbf{y}}_t - \mathbf{y}_t) \quad (4.67)$$

being $(\hat{\mathbf{y}}_t, \hat{\mathbf{u}}_t)$ the nominal state/control trajectories.

4.5 An optimal formulation of Dynamic Movement Primitives

Dynamic Movement Primitives (DMP) consist of two main components: one ensures convergence to the final goal position, while the other focuses on replicating the acceleration profiles to capture the shape of a demonstrated trajectory. Similarly, Linear Quadratic Tracking (LQT) is characterized by a cost function that integrates multiple components. These components operate at different levels of

state and control, with references that can either take the form of a complete trajectory or a final goal position (Calinon, 2023b). Specifically, LQT's cost function can be designed to track a reference velocity or acceleration profile while simultaneously ensuring that the system reaches a predefined target state by the end of the movement. The importance of these objectives, accurately tracking the desired profiles and achieving the final goal position, is managed through weight matrices that balance the priorities of these tasks. Given the structural and functional parallels between DMP and LQT, DMP can be reinterpreted within an LQT framework, as shown in (Calinon, 2023a; Y. Zhang et al., 2024). In this formulation, the acceleration profile derived from the demonstrated trajectory serves as the reference trajectory for LQT, while the attractor in DMP is treated as the goal state to be achieved at the movement's conclusion. This reformulation highlights the underlying alignment between the two approaches and allows for leveraging LQT's mathematical structure to understand or optimize DMP-based methods.

To do so, we start from the formulation introduced in Sec. 4.4 for the LQR problem, and make it more explicit to highlight the similarities with the DMP. In this case, without loss of consistencies with the sections above, we will refer to a Linear Quadratic Tracking (LQT) problem and omit the phase variable s_t .

$$\begin{aligned} c &= (\boldsymbol{\mu}_T - \mathbf{y}_T)^\top \mathbf{Q}_T (\boldsymbol{\mu}_T - \mathbf{y}_T) + \sum_{t=1}^{T-1} ((\boldsymbol{\mu}_t - \mathbf{y}_t)^\top \mathbf{Q}_t (\boldsymbol{\mu}_t - \mathbf{y}_t) + \mathbf{u}_t^\top \mathbf{R}_t \mathbf{u}_t) \\ &= (\boldsymbol{\mu} - \mathbf{y})^\top \mathbf{Q} (\boldsymbol{\mu} - \mathbf{y}) + \mathbf{u}^\top \mathbf{R} \mathbf{u}, \end{aligned} \quad (4.68)$$

with $\mathbf{y} = [\mathbf{y}_0^\top, \mathbf{y}_1^\top, \dots, \mathbf{y}_T^\top]^\top$ representing the evolution of the state variables, $\mathbf{u} = [\mathbf{u}_0^\top, \mathbf{u}_1^\top, \dots, \mathbf{u}_{T-1}^\top]^\top$ the evolution of the control commands, $\boldsymbol{\mu} = [\boldsymbol{\mu}_0^\top, \boldsymbol{\mu}_1^\top, \dots, \boldsymbol{\mu}_T^\top]^\top$ the evolution of the tracking targets, $\mathbf{Q} = \text{blockdiag}(\mathbf{Q}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_T)$ the evolution of the precision matrices $\mathbf{Q}_t$, and $\mathbf{R} = \text{blockdiag}(\mathbf{R}_0, \mathbf{R}_1, \dots, \mathbf{R}_{T-1})$ representing the evolution of the control weight matrices $\mathbf{R}_t$. Even in this case, the state $\mathbf{y}_t$ evolves accordingly to a linear system described by:

$$\mathbf{y}_{t+1} = \mathbf{A}_t \mathbf{y}_t + \mathbf{B}_t \mathbf{u}_t, \text{ yielding at trajectory level } \mathbf{y} = \mathbf{S}_y \mathbf{y}_0 + \mathbf{S}_u \mathbf{u}. \quad (4.69)$$

Then, to find the controller that minimizes the cost function in (4.68), one can substitute (4.69) and differentiate with respect to $\mathbf{u}$, which gives the optimal open-loop control command (Calinon, 2023b)

$$\hat{\mathbf{u}} = (\mathbf{S}_u^\top \mathbf{Q} \mathbf{S}_u + \mathbf{R})^{-1} \mathbf{S}_u^\top \mathbf{Q} (\boldsymbol{\mu} - \mathbf{S}_y \mathbf{y}_0). \quad (4.70)$$

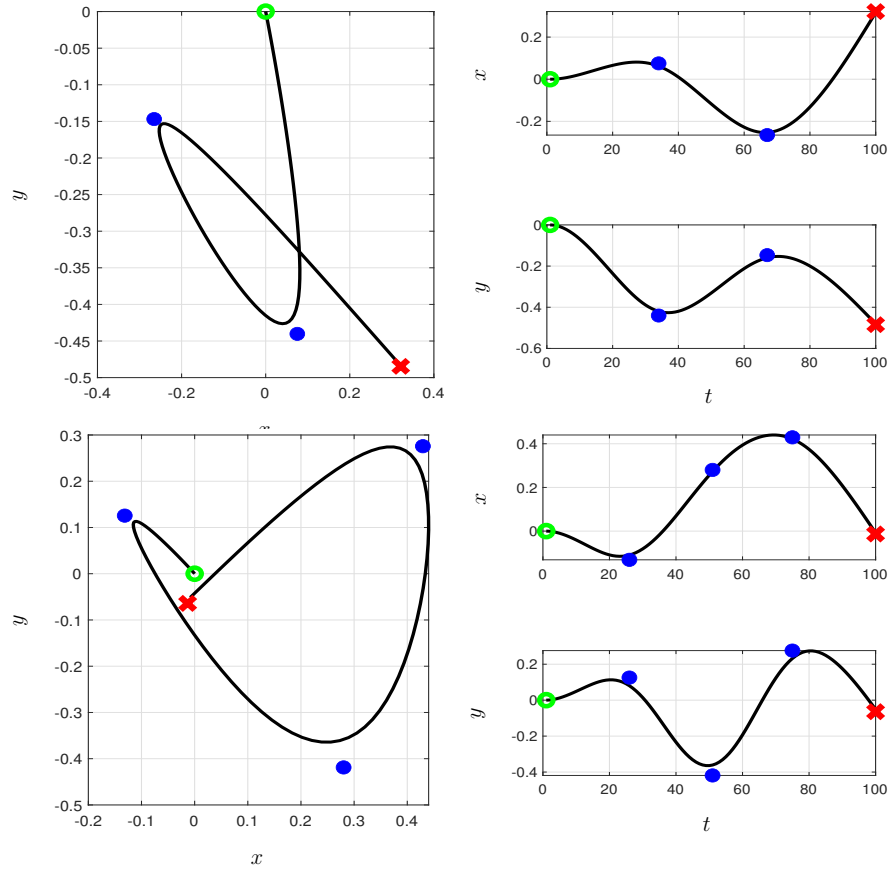


Figure 4.7: LQT-CP when adding via-points (in blue) between the start (green) and end (red) positions.

It is important to underline the simplicity of the result above, relying only on straightforward linear algebra calculations. This simplicity sets it apart from DMPs while offering additional flexibility. Unlike DMPs, the LQT controller described in (4.70) can be effectively applied in LfD scenarios where multiple demonstrations are available. In such cases, variations observed across tasks can be systematically incorporated into the LQT framework by defining them within its objective function. This formulation enables the derivation of a trajectory distribution in the control space, which can subsequently be transformed into a trajectory distribution in the state space. This transformation is made possible by the design of the precision matrix $\mathbf{Q}$, which plays a crucial role. The precision matrix encodes the relative importance of accurately tracking different aspects of the demonstrated trajectories while also capturing the correlation constraints inherent in the dataset. Indeed, if one would use GMM to encode a dataset of demonstrated trajectories, the precision matrix $\mathbf{Q}$ is the inverse of the covariance Σ of the GMM fitted to the training data (Calinon et al., 2007). This approach not only enhances the ability to generalize across demonstrations but

also facilitates a more structured representation of task variability.

An approach similar to DMP, where the forcing term in (4.29) is composed as a sum of basis functions, can be adopted for the controller term in (4.70) by means of the so-called Control Primitives (CP), i.e. assuming that the control command takes the form $\mathbf{u} = \Psi\boldsymbol{\omega}$ (Calinon, 2023a; Y. Zhang et al., 2024). With this, one can rewrite the objective function in (4.68) as:

$$\min_{\boldsymbol{\omega}} (\boldsymbol{\mu} - \mathbf{y})^\top \mathbf{Q} (\boldsymbol{\mu} - \mathbf{y}) + \Psi^\top \boldsymbol{\omega}^\top \mathbf{R} \Psi \boldsymbol{\omega} \quad \text{s.t.} \quad \mathbf{y} = \mathbf{S}_y \mathbf{y}_0 + \mathbf{S}_u \Psi \boldsymbol{\omega}, \quad (4.71)$$

from which one can compute the optimal weight $\hat{\boldsymbol{\omega}}$ as

$$\hat{\boldsymbol{\omega}} = (\Psi^\top \mathbf{S}_u^\top \mathbf{Q} \mathbf{S}_u \Psi + \Psi^\top \mathbf{R} \Psi)^{-1} \Psi^\top \mathbf{S}_u^\top \mathbf{Q} (\boldsymbol{\mu} - \mathbf{S}_y \mathbf{y}_0). \quad (4.72)$$

The computed trajectory $\hat{\mathbf{u}} = \Psi \hat{\boldsymbol{\omega}}$ represents the series of control commands arranged in vector form within the control space. This formulation facilitates the straightforward conversion of the trajectory from the control space to the state space using the relation $\hat{\mathbf{y}} = \mathbf{S}_y \mathbf{y}_0 + \mathbf{S}_u \hat{\mathbf{u}}$. This linear transformation ensures that all operations remain linear throughout the process. Consequently, a covariance matrix defined in the parameter space of $\boldsymbol{\omega}$ can be directly translated into corresponding covariance matrices in the control space $\mathbf{u}$ and in the state space $\mathbf{y}$. This capability provides a structured way to map uncertainties and variations across spaces, enhancing the adaptability of the control framework.

Figure 4.7 shows the LQT-CP behavior when adding via-points in the desired trajectory towards the goal position. It is possible to observe that the framework can seamlessly incorporate via-point specifications. This is achieved by marking the via-points directly in the reference trajectory $\boldsymbol{\mu}$ and assigning equal importance to via-point tracking and goal-reaching within the precision matrix $\mathbf{Q}$. This flexibility makes LQT-CP highly adaptable for capturing complex trajectory constraints and dynamics.

4.5.1 Recursive formulation of LQT with control primitives

In Sec. 4.4.3 it was shown that, by exploiting a recursive version of the LQR algorithm, one can re-write the optimal control command in a form $\mathbf{u} = -\mathbf{F}\tilde{\mathbf{y}}_0$. Indeed, with CP the same control command can be written in the form $\mathbf{u} = -\Psi \mathbf{W} \tilde{\mathbf{y}}_0$, with $\mathbf{W}$ being the control weights' matrix. The state $\tilde{\mathbf{y}}_0$ indicates the augmented state which is taken in order to write the LQT problem in (4.68) into

a corresponding LQR version, that is

$$\min_{\mathbf{W}} = \tilde{\mathbf{y}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{y}} + (\Psi \mathbf{W} \tilde{\mathbf{y}}_0)^\top \mathbf{R} (\Psi \mathbf{W} \tilde{\mathbf{y}}_0) \quad \text{s.t.} \quad \tilde{\mathbf{y}} = (\tilde{\mathbf{S}}_y - \tilde{\mathbf{S}}_u \Psi \mathbf{W}) \tilde{\mathbf{y}}_0. \quad (4.73)$$

This version of LQT is most of the time referred to as LQT-CP and was introduced in (Calinon, 2023b; Y. Zhang et al., 2024). In (4.73), $\tilde{\mathbf{S}}_y$ and $\tilde{\mathbf{S}}_u$ are the state and control transfer matrices, respectively, composed by the augmented matrices $\tilde{A}_t$ and $\tilde{B}_t$ defining the augmented dynamical system

$$\tilde{\mathbf{y}}_{t+1} = \tilde{A}_t \tilde{\mathbf{y}}_t + \tilde{B}_t \mathbf{u}_t, \quad \text{with} \quad \tilde{A}_t = \begin{bmatrix} \mathbf{A}_t & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \tilde{B}_t = \begin{bmatrix} \mathbf{B}_t \\ 0 \end{bmatrix}; \quad (4.74)$$

the term $\tilde{\mathbf{y}}$, instead, accounts for the concatenation of the augmented state $\tilde{\mathbf{y}}_t = [\mathbf{y}_t^\top, 1]^\top$, while the term $\tilde{\mathbf{Q}}$ concatenates the augmented precision matrices $\tilde{Q}_t$ computed as

$$\tilde{Q}_t = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\boldsymbol{\mu}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_t & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \boldsymbol{\mu}_t \\ \mathbf{0} & 1 \end{bmatrix}. \quad (4.75)$$

Solving (4.73) one obtains the optimal control matrix the optimal weight $\hat{\omega}$ as

$$\hat{\mathbf{W}} = (\Psi^\top \tilde{\mathbf{S}}_u^\top \tilde{\mathbf{Q}} \tilde{\mathbf{S}}_u \Psi + \Psi^\top \mathbf{R} \Psi)^{-1} \Psi^\top \tilde{\mathbf{S}}_u^\top \tilde{\mathbf{Q}} \tilde{\mathbf{S}}_y, \quad (4.76)$$

which can be used as a recursive formulation to generate a feedback controller that can handle external perturbation in real-time execution, providing a feedback controller $\mathbf{u}_t = -\tilde{\mathbf{K}}_t \tilde{\mathbf{y}}_t$ where $\tilde{\mathbf{K}}_t$ is the feedback gain matrix at time step t . In particular, $\tilde{\mathbf{K}}_t$ can be recursively calculated as

$$\tilde{\mathbf{K}}_t = \Psi_t \hat{\mathbf{W}} \mathbf{P}_t \quad \text{with} \quad \mathbf{P}_t = \mathbf{P}_{t-1} (\tilde{A}_{t-1} - \tilde{B}_{t-1} \tilde{\mathbf{K}}_{t-1})^{-1}, \quad (4.77)$$

where $\tilde{\mathbf{K}}_0 = \Psi_0 \hat{\mathbf{W}}$, $\mathbf{P}_0 = \mathbf{I}$.

The LQT-CP framework offers enhanced flexibility through the design of the precision matrix $\mathbf{Q}$. This matrix can be tailored to prioritize tracking different profiles of the demonstrated trajectory, such as acceleration profiles and goal positions, while also capturing their correlation constraints. By carefully crafting $\mathbf{Q}$, it becomes possible to transform the demonstrated trajectory into an optimal control formulation of Dynamic Movement Primitives (DMP).

For scenarios with a single demonstration, $\mathbf{Q}$ can be designed with arbitrarily high values, provided the relative importance of trajectory tracking and goal reaching is properly balanced. This approach is analogous to setting the stiffness

and damping gains in traditional DMP formulations.

In cases where multiple demonstrations are available, $\mathbf{Q}$ can be constructed as the inverse of the observed covariance matrix. This ensures that the system replicates the task while preserving the same variations and correlations found in the demonstration set.

4.5.2 Discussion

This chapter primarily introduces Dynamic Movement Primitives (DMP) and Linear Quadratic Tracking (LQT) as encoding techniques. In the following chapter, these methods are implemented in the context of human hand-guiding, where a robot assists in constraining human upper limb movements along predefined paths.

Such applications present two main challenges. First, ensuring a natural and intuitive guidance experience that remains compliant with human feedback. Second, addressing potential unwanted effects and instabilities arising from direct human-robot contact, which could pose safety risks. DMP and LQT offer promising theoretical foundations to tackle these issues, although their integration into human-guiding applications remains relatively unexplored.

On one hand, DMP provides generalization capabilities, allowing the transfer of demonstrated skills across different subjects. Additionally, its simple yet robust structure facilitates the inclusion of coupling terms, enhancing adaptability. On the other hand, LQT presents a more generalized framework for encoding dynamical systems like DMP, as discussed in the previous section. Its formulation seamlessly integrates into various optimization problems and benefits from extensive theoretical research.

One aspect that, to the best of our knowledge, remains insufficiently explored is the study of these techniques within a Human-Robot Interaction (HRI) framework, particularly with human-oriented evaluation metrics. As an additional contribution, the following chapters will propose novel evaluation metrics and analyses to better interpret the recorded results and assess the effectiveness of these approaches in practical HRI scenarios.

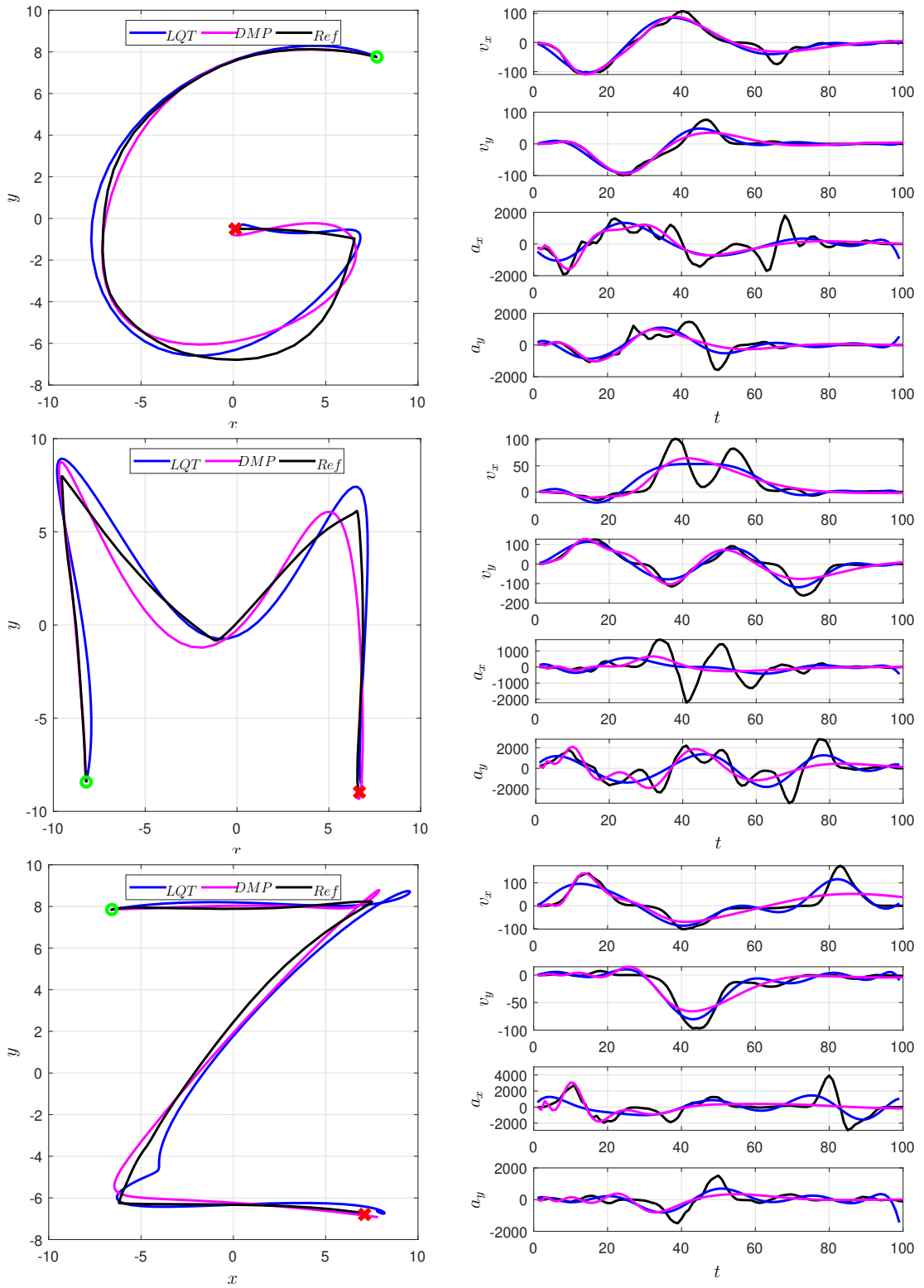


Figure 4.8: LQT-CP (blue) and DMP (magenta) comparisons with respect to reference trajectory (in black).

Enhancing human-robot interaction: Geometric Dynamic Movement Primitives

Dynamic Movement Primitives (DMP) introduced in Chapter 4.3 have been used in the past for several applications (Nemec et al., 2013; Ruckert & d’Avella, 2013; Sidiropoulos & Doulgeri, 2022). However, their usage in co-manipulation tasks still needs to be investigated. For instance, in (Lauretti et al., 2017), DMP is adopted in a LfD framework used in sequence for an assistive task, introducing computational results that lack metrics evaluating the interaction with the human itself. The same applies to (Dahlin & Karayiannidis, 2021; Nemec, Likar, et al., 2018; Sidiropoulos & Doulgeri, 2021), where DMP is used to encode tasks of the robot that should be performed in cooperation with a human user, yet lacking the formulation and analysis of a comprehensive framework adaptable to different co-manipulation tasks.

For instance, during co-manipulation, the user is in contact with the robot, which is used as an auxiliary instrument to ensure (i) correct guidance, (ii) task repeatability, (iii) task generalization to different targets and users. Because of this, the following features for the robot controller were identified:

- the robot should adopt virtual fixtures to constrain its movements to a specific area of its workspace;
- guidance is performed in terms of geometric constraints, but speed variations in robot movements have to be chosen as a response to human feedback;
- the framework should allow compliance with different user limitations;

- the framework should include a mechanism to facilitate task reproduction in different scenarios.

These requirements lead to a human-in-the-loop architecture, where the validation process must consider not only the nature of the robot controller but also its robustness and safety in relation to the user. The user, as an integral part of the system, should not be treated as a separate entity. In this context, we identified Dynamic Movement Primitives (DMP) as a promising encoding framework. DMP can be seamlessly integrated into a Learning from Demonstration (LfD) framework, allowing for consistent task encoding and generalization (see Fig. 4.5).

However, the application of DMP in co-manipulation tasks still requires further investigation. For instance, in virtual fixture applications, which are explored in this chapter, one might wish to impose a predefined path that constrains the robot's movement without enforcing a specific timing law. The phase variable $s(t)$ in such cases can be arbitrarily adjusted, allowing the system to progress along the predefined path. In this case, classical DMP implementation introduces limitations because the recorded reference trajectories are influenced by the time dependency observed during task demonstrations. These may include variations in velocity or pausing intervals.

To address the challenges discussed in this section, we propose the concept of Geometric DMP (GDMP). This approach integrates the SS algorithm introduced in Chapter 3, enabling a geometric interpretation of the phase variable $s(t)$. Unlike the standard approach, the new phase variable is no longer constrained by the timing law of the demonstration and can be adapted based on the specific needs of the application.

5.1 Geometric Dynamic Movement Primitives

Geometric Dynamic Movement Primitives (GDMP) were introduced in (Braglia et al., 2025) as a novel DMP architecture tailored for human-robot co-manipulation tasks. This approach combines the reversible DMP framework described in Section 4.3.2 with the Sampling Algorithm (SS) from Chapter 3.2. The SS algorithm ensures the regularity of the reference curve $\mu(s)$ by parametrizing it with the arc-length s . This decouples the phase profile of the DMP from its transformation system, enabling flexible definitions of the canonical system. In this sense, the term *Geometric* was chosen to emphasize that the proposed DMP framework

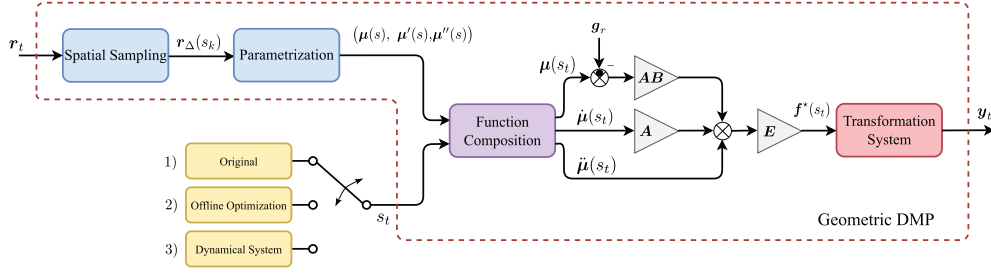


Figure 5.1: Schematic of the proposed framework based on Geometric DMP.

is based on the geometrical path of the demonstrated trajectory, rather than its time-based demonstration. For instance, the phase variable s_t can be chosen based on the three distinct scenarios depicted in Fig. 5.1:

- 1) Original: the demonstrated trajectory is reproduced using the original timing law $s_t = \gamma_t$.
- 2) Offline Optimization: the phase variable s_t results from an optimization problem, which optimizes the motion along the geometric path to achieve a desired objective, as discussed in Sec. 5.2.
- 3) Dynamical System: the phase variable s_t is defined by a dynamical system, possibly dependent on external inputs. This scenario includes co-manipulations tasks, as discussed in Sec. 5.3 and Sec. 5.4.

The first scenario is straight forward and can be easily demonstrated. Given the arc-length parametrization of the reference curve $\boldsymbol{\mu}(s)$ obtained by the SS algorithm, its velocity can be computed as $\dot{\boldsymbol{\mu}}(s_t) = \boldsymbol{\mu}'(s_t)\dot{s}_t$. If replicating the demonstrated reference $\mathbf{r}_t$, $t \in [0, T]$, is of interest, this translates into finding $\dot{s}_t^*$ such that $\dot{\boldsymbol{\mu}}(s_t^*) = \dot{\mathbf{r}}_t$. Indeed this conditions gives

$$\dot{\boldsymbol{\mu}}(s_t) = \boldsymbol{\mu}'(s_t)\dot{s}_t = \dot{\mathbf{r}}_t \implies \dot{s}_t^* \triangleq (\boldsymbol{\mu}'(\gamma_t))^{-1} \cdot \dot{\mathbf{r}}_t. \quad (5.1)$$

Note that in (5.1), $\boldsymbol{\mu}(\gamma_t)$ can be easily obtained from LWR in (4.31), while the existence of $(\boldsymbol{\mu}'(\gamma_t))^{-1}$ is guaranteed from (3.7) since the arc-length parametrization obtained through the SS algorithm is regular.

In the following sections, the second and third scenario listed above are investigated.

5.2 Phase Optimization in Geometric DMP

Many planning techniques focus on minimizing the execution time while satisfying task and kinematic constraints (Gattringer et al., 2022; Jankowski et al., 2023;

Kanazawa et al., 2019; Sidiropoulos et al., 2023; Verschure et al., 2009). This optimization problem can be effectively addressed using the proposed GDMP framework, where the phase variable s_t is computed as the output of this dynamic system (i.e. a chain of two integrators):

$$\dot{\mathbf{s}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{s} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u,$$

where $\mathbf{s} = [s_t, \dot{s}_t, \ddot{s}_t]^\top$ and $u = \ddot{s}_t$. The optimization problem can be formulated as follows (Gattringer et al., 2022):

$$\min_{u(\cdot)} T = \int_0^T 1 dt = \int_0^{\Delta M} \frac{1}{\dot{s}_t} ds_t, \quad (5.2)$$

subject to:

$$\begin{aligned} s_0 &= 0, \quad s_T = \Delta M, \\ \dot{s}_0 &= \dot{s}_T = 0, \quad \dot{s} \geq 0, \\ \ddot{s}_0 &= \ddot{s}_T = 0. \end{aligned}$$

Additional constraints can be imposed on $\mathbf{s}$ and u :

$$|\dot{s}_t| \leq \dot{s}_{\text{MAX}}, \quad |\ddot{s}_t| \leq \ddot{s}_{\text{MAX}}, \quad |\ddot{s}_t| \leq \ddot{s}_{\text{MAX}}, \quad (5.3)$$

or on the robot velocity and acceleration in the task space:

$$\begin{aligned} \|\eta \dot{\boldsymbol{\mu}}(s_t)\| &\leq \dot{\mathbf{y}}_{\text{MAX}} \Leftrightarrow \|\dot{\boldsymbol{\mu}}(s_t)\| \leq \dot{\mathbf{y}}_{\text{MAX}}/\eta, \\ \|\eta \ddot{\boldsymbol{\mu}}(s_t)\| &\leq \ddot{\mathbf{y}}_{\text{MAX}} \Leftrightarrow \|\ddot{\boldsymbol{\mu}}(s_t)\| \leq \ddot{\mathbf{y}}_{\text{MAX}}/\eta. \end{aligned} \quad (5.4)$$

Alternatively, or in addition, constraints in the joint space can be considered as well, typically caused by the physical limitations of the actuation systems. Let $\mathbf{q}^*(s_t) \in \mathbb{R}^n$, with n the number of joints, be the joint trajectory profiles computed as follows:

$$\mathbf{q}^*(s_t) = \text{IK}(\dot{\boldsymbol{\mu}}_\eta(s_t), \mathbf{R}^*(s_t)),$$

where $\text{IK}(\cdot)$ is the Inverse Kinematics function of the robot. Function $\boldsymbol{\mu}_\eta(s_t)$ is the demonstrated (position) trajectory defined in (4.42), while $\mathbf{R}^*(s_t)$ is the desired orientation (in the simulation, $\mathbf{R}^*(s_t)$ is kept constant but, in principle, can be

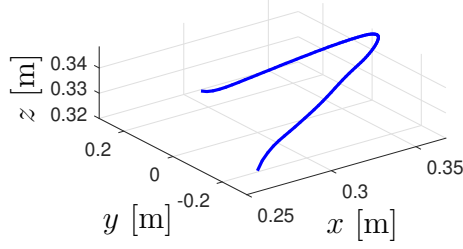


Figure 5.2: Demonstrated curve used for optimization.

also varied during the demonstration). Velocity and acceleration constraints¹ in the joint space can be expressed as:

$$|\dot{\mathbf{q}}^*(s_t)| \leq \dot{\mathbf{q}}_{\text{MAX}}, \quad |\ddot{\mathbf{q}}^*(s_t)| \leq \ddot{\mathbf{q}}_{\text{MAX}}. \quad (5.5)$$

The considered optimization problem has been implemented in MATLAB using the CasADi optimal solver (Andersson et al., 2019). The reference curve $\mu_\eta(s)$ is depicted in Fig. 5.2 and the robot considered in the simulation is the same used in the experimental tests described in Sec. 5.4, i.e. a Franka Emika Panda robot. As a 7-DOF robot with a redundant kinematic structure, the inverse kinematics is not available analytically and must be computed numerically (in particular, we used the inverse kinematics solver from the Matlab Robotics System Toolbox). Three different scenarios are considered: a) the constraints (5.3), generating the timing law $s_t = \gamma_{s,t}$; b) the constraints (5.5), generating the timing law $s_t = \gamma_{q,t}$; c) both the constraints (5.3) and (5.5), generating the timing law $s_t = \gamma_{qs,t}$. The obtained results are compared in Fig. 5.3 with the original timing law γ_t . The figure clearly shows that the execution time is significantly reduced compared to the demonstrated trajectory: $\gamma_{s,t}$ and $\gamma_{qs,t}$ result in an execution time $T = 1.1$ s, denoting a reduction of 64.5% with respect to γ_t , while $\gamma_{q,t}$ results in an execution time of $T = 0.6$ s, denoting a reduction of 80.6% with respect to γ_t . Indeed, one can conclude that the constraints (5.3) are more restrictive than (5.5).

The normalized profiles $\dot{\bar{\gamma}}_s = \dot{\gamma}_{s,t}/\dot{s}_{\text{MAX}}$ and $\ddot{\bar{\gamma}}_s = \ddot{\gamma}_{s,t}/\ddot{s}_{\text{MAX}}$ displayed in Fig. 5.4(a) confirm the optimality of the timing law $\gamma_{s,t}$, since the boundary of at

¹The practical definition of the constraints (5.5) is obtained by computing the joint configurations $\mathbf{q}_k$ from $(\eta \mathbf{r}_{\Delta,k}, \mathbf{R}_k)$, for $k = 0, \dots, M$, using the IK algorithm. Then, the parametric function $\mathbf{q}^*(s)$ is deduced by applying the same algorithm used for $\mu(s)$ in (4.39). Finally, the constraints are defined as a function of the state $\mathbf{s} = [s_t, \dot{s}_t, \ddot{s}_t]^\top$ and of $\mathbf{q}^*(s)$, $\mathbf{q}^{*'}(s)$ which are computed analytically from $\mathbf{q}^*(s)$. Therefore, the two inequalities become

$$\begin{aligned} -\dot{\mathbf{q}}_{\text{MAX}} &\leq \mathbf{q}^{*'}(s_t) \dot{s}_t &\leq \dot{\mathbf{q}}_{\text{MAX}}, \\ -\ddot{\mathbf{q}}_{\text{MAX}} &\leq \mathbf{q}^{*''}(s_t) \dot{s}_t^2 + \mathbf{q}^{*'}(s_t) \ddot{s}_t &\leq \ddot{\mathbf{q}}_{\text{MAX}}. \end{aligned}$$

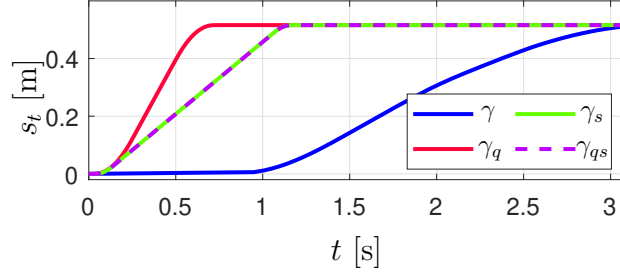


Figure 5.3: Phase profiles obtained as a solution of the optimization problem (5.2) subject to constraints (5.3) and (5.5).

least one constraint in (5.3) is reached at every time instant. The same conclusion can be drawn for the timing law $\gamma_{q,t}$ by referring to Fig. 5.4(b), presenting the normalized profiles $\dot{\tilde{q}} = \dot{q}(\gamma_{q,t})/\dot{q}_{\text{MAX}}$ and $\ddot{\tilde{q}} = \ddot{q}(\gamma_{q,t})/\ddot{q}_{\text{MAX}}$. In this case, the limiting factors are the velocity of joint #7 and the acceleration of joint #3.

5.2.1 Comparison with other DMPs

To demonstrate the effectiveness of the GDMP framework shown in Fig. 5.1, we applied it to solve the optimization problem (5.2), focusing on the task-space constraints described in (5.4). To provide a comprehensive evaluation, we compared its performance with several approaches from the literature, including the classical DMP (Ijspeert et al., 2013), Temporal Coupling DMP (TCDMP) (Dahlin & Karayiannidis, 2021), the reversible DMP (RDMP) (Sidiropoulos & Doulgeri, 2021), and offline optimal DMP (DMP_P^{*}) (Sidiropoulos et al., 2023). These methods were applied to the same demonstrated trajectory, depicted in black in Fig. 5.5(a).

For what concerns GDMP, the trajectory was processed, and the optimization problem (5.2) was solved under the kinematic constraints (5.3) and (5.4). The optimization yielded an optimal task duration of $T^* = 4.11\text{s}$. For consistency, the durations of the other methods were scaled to match T^* , enabling a direct comparison of their adherence to the constraints and trajectory approximation.

The classical DMP achieves time scaling by modifying the temporal parameter τ , preserving the original curve shape, as shown in purple in Fig. 5.5(a). However, this approach does not inherently account for velocity or acceleration constraints, leading to violations of these constraints even when the desired task duration is achieved, see Fig. 5.5(b). In contrast, TCDMP leverages temporal coupling to enforce kinematic constraints by appropriately adjusting τ . While this ensures that the resulting trajectory stays within the constraint boundaries, it fails to meet the optimal duration T^* , as the coupling mechanism indirectly

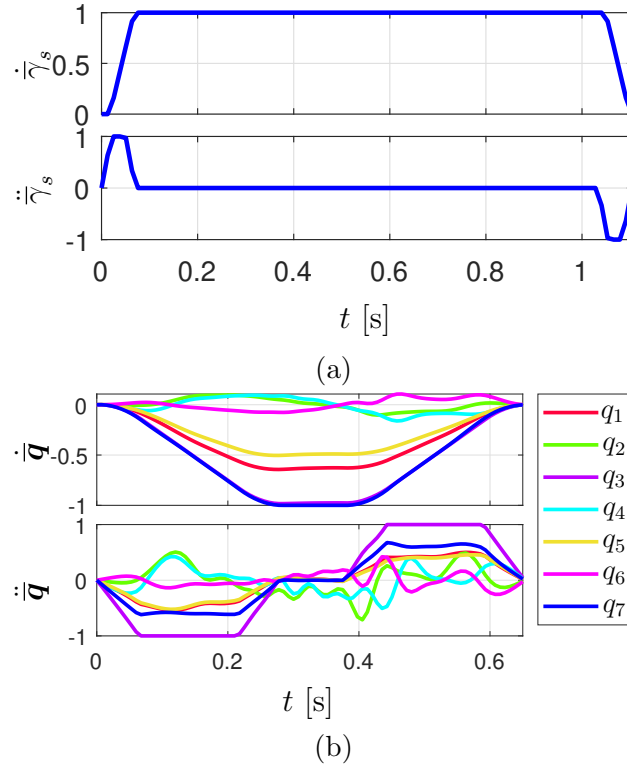


Figure 5.4: Normalized results of the optimization problem (5.2) subject to (a) constraints (5.3) and (b) constraints (5.5).

affects task timing. Similarly, the RDMP structure allows addressing the optimization problem (5.2) via s_t^* but fails to achieve $T^* = 4.11s$ due to its reliance on time-parametrized demonstrations, which encode speed variations and pauses.

Finally, DMP_P^* addresses these limitations by directly optimizing the weights ω in (4.40), ensuring both compliance with the constraints and the desired task duration. However, strict kinematic constraints can lead to a compromise in the curve approximation, as the optimization prioritizes constraint satisfaction over precise interpolation of the demonstrated trajectory, which can be observed in Fig. 5.5(a).

The limitations of the existing approaches are summarized in the table of Fig. 5.5(c), where the terms “Trajectory Reproduction”, “Constraints Satisfaction”, and “Optimal Execution (min T)” indicate the ability to reproduce the demonstrated trajectory, satisfy kinematic constraints, and achieve the optimal duration T^* , respectively. While each existing DMP solution has specific drawbacks, the proposed GDMP, as highlighted in green in Fig. 5.5(c), successfully fulfills all task requirements.

This success stems from the arc-length parametrization provided by the SS algorithm, which decouples the trajectory’s path and velocity information. This

decoupling allows the velocity profile to be computed independently, ensuring compliance with kinematic constraints while maintaining the desired trajectory shape and task duration. Consequently, GDMP offers a robust solution for tasks requiring high precision, constraint adherence, and time optimization.

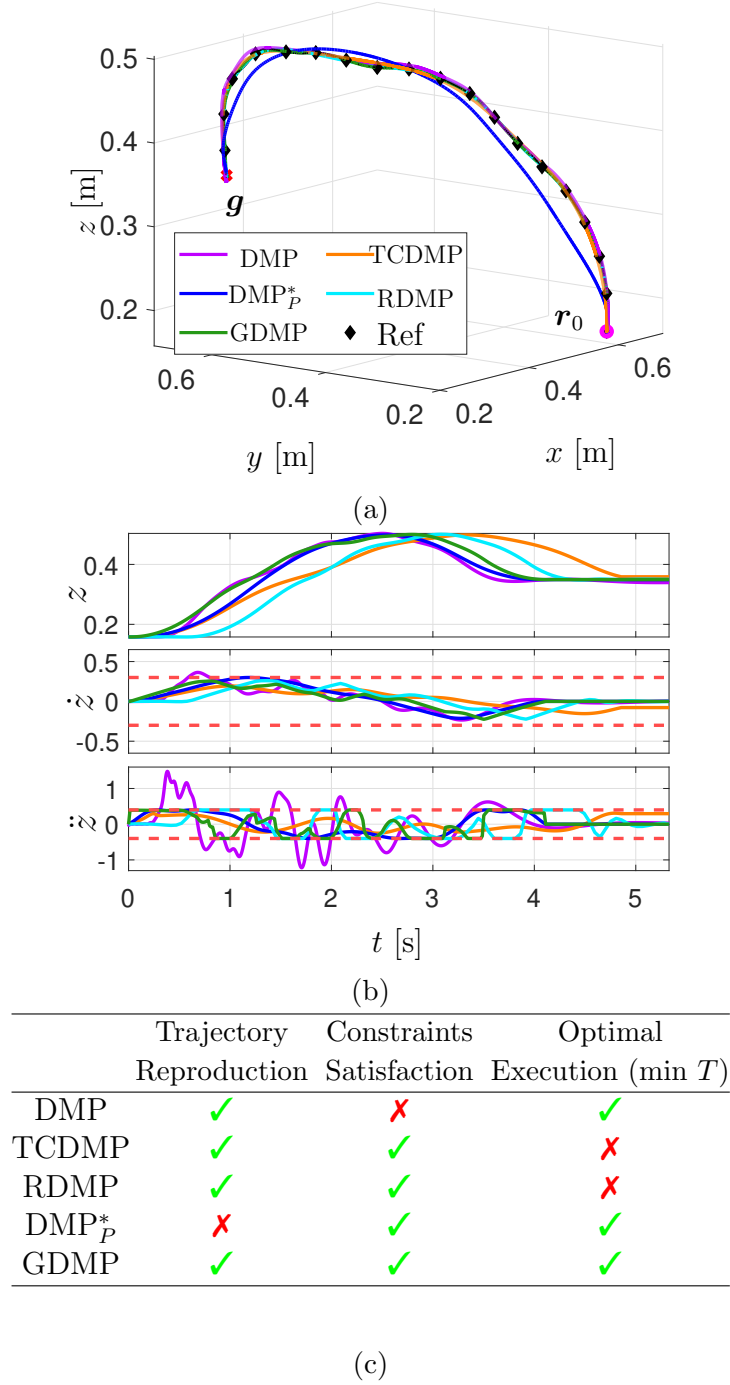


Figure 5.5: Comparison of GDMP with other DMP solutions in addressing the minimum time optimization problem. (a) 3D path recordings; (b) z -axis' position, velocity and acceleration trajectories for each DMP model; (c) final evaluation table.

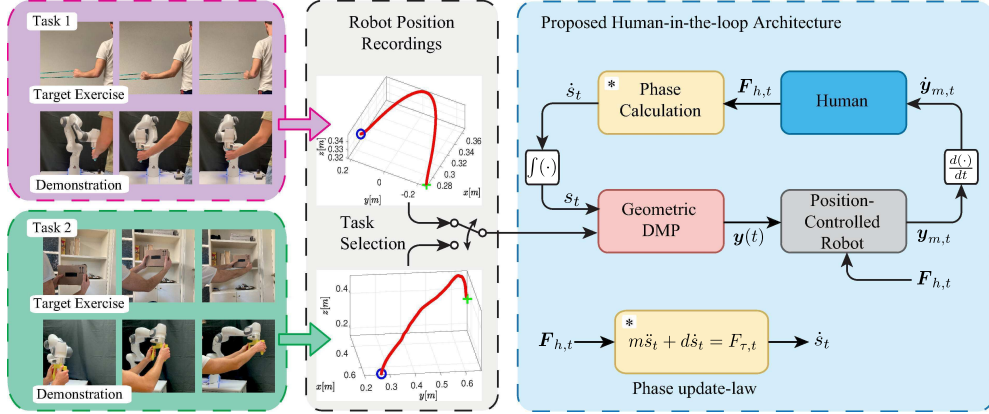


Figure 5.6: Human-robot interaction through kinesthetic guidance along the trajectories encoded by GDMP.

5.3 Human-in-the-loop Geometric DMP

The considered human-robot interaction framework employing GDMP is schematically depicted in Fig. 5.6. The phase variable s_t is determined as follows:

$$m \ddot{s}_t + d \dot{s}_t = F_{\tau,t} = \boldsymbol{\mu}'_\eta(s_t)^\top \mathbf{F}_{h,t}, \quad (5.6)$$

where $\boldsymbol{\mu}_\eta(s) = \mathbf{E}\boldsymbol{\mu}(s_t)$ is a generalization of (4.42) to the multidimensional case, with $\mathbf{E}$ being defined in eqs.(4.38) and (4.44). Equation (5.6) represents the dynamics of a mass m with damping coefficient d , influenced by the external force $\mathbf{F}_{h,t}$ applied to the robot end-effector by a human operator. The presence of the scaling factor $\mathbf{E}$ in (5.6) brings the following two features:

- 1) It enables a consistent human-robot interaction, as further discussed in the remainder of this section;
- 2) It ensures the passivity of the human-robot interaction framework in Fig. 5.6, as proven in Sec. 5.3.1.

The force $F_{\tau,t}$ in (5.6) denotes the projection of the human force vector $\mathbf{F}_{h,t}$ onto the tangential direction of the reference trajectory $\boldsymbol{\mu}_\eta(s) = \mathbf{E}\boldsymbol{\mu}(s_t)$, where $\boldsymbol{\mu}(s_t)$ is the parameterized trajectory as depicted in the GDMP schematic of Fig. 5.1. From (3.7), it follows that $\|\boldsymbol{\mu}'_\eta(s_t)\| \neq 0$, meaning that the projection in (5.6) can always be successfully computed (Biagiotti & Melchiorri, 2008; Simonic et al., 2021). This presents an alternative solution to the approach proposed in (Nemec et al., 2019), where a minimum threshold is employed instead.

With reference to the aforementioned feature 1), the inclusion of matrix $\mathbf{E}$ ensures that the tangential direction remains consistent with the current orientation of the new curve (see Fig. 5.7). This feature proves valuable as it guarantees an

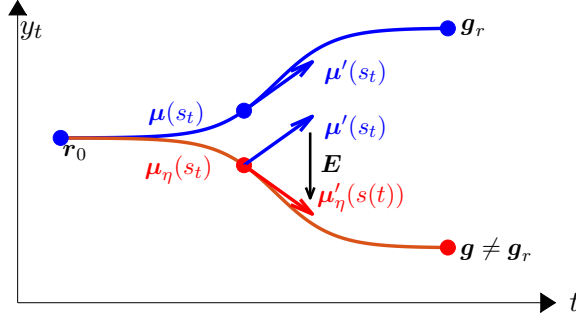


Figure 5.7: An example showcasing the impact of the scaling matrix $\mathbf{E}$ in (5.6) on the projection of the force $\mathbf{F}_{h,t}$.

accurate projection of the force $\mathbf{F}_{h,t}$ in (5.6) without the necessity of a new DMP parameterization (Koutras & Doulgeri, 2020b; Sidiropoulos & Doulgeri, 2022).

As shown in the framework of Fig. 5.6, an additional term $\mathbf{F}_{h,t}$ is inserted in Eqs. (4.41) and (4.43), leading to the following DMP formulation:

$$\ddot{\mathbf{y}}_t + \mathbf{A}\dot{\mathbf{y}}_t + \mathbf{A}\mathbf{B}\mathbf{y}_t - \mathbf{A}\mathbf{B}\mathbf{g} = \mathbf{E}[\ddot{\boldsymbol{\mu}}(s_t) + \mathbf{A}\dot{\boldsymbol{\mu}}(s_t) + \mathbf{A}\mathbf{B}\boldsymbol{\mu}(s_t) - \mathbf{A}\mathbf{B}\mathbf{g}_r] + \mathbf{F}_{h,t}, \quad (5.7)$$

where the additional term $\mathbf{F}_{h,t}$ ensures the passivity of the human-robot interaction framework of Fig. 5.6, as proven in Sec. 5.3.1. The motion trajectory $\mathbf{y}_t$ is then fed into a robot manipulator equipped with high-gain position control, which essentially implements an admittance control system for the physical human-robot interaction (Keemink et al., 2018). In this setup, the admittance model restricts the user's movement to follow the trajectories defined by the GDMP, allowing forward or backward motion. As shown in (Keemink et al., 2018), although the human user might exhibit a non-passive behavior, a feasible approach for ensuring a safe human-robot interaction is to verify the passivity of the system with respect to the energetic port $\langle \dot{\mathbf{y}}_t, \mathbf{F}_{h,t} \rangle$ describing the interaction between the human and the controlled robot.

5.3.1 Passivity Analysis

Definition 5.3.1 (*Passivity*). Let $P_{Tot,t} = \sum_{i=1}^n P_{i,t} = \mathbf{u}_t^\top \mathbf{z}_t$ be the total power flowing through the n energetic ports of a dynamical system, where $\mathbf{u}_t$ and $\mathbf{z}_t$ are the input and output vectors. Such a system is said to be passive if there exists a non-negative storage function $S(\mathbf{x}_t)$ of the state vector $\mathbf{x}_t$ such that, for any time interval $[t_0, t_1]$ and for any initial state vector $\mathbf{x}_0$, and being $\mathbf{x}_1$ the state vector at time t_1 , the following inequality holds true (Colgate & Hogan, 1989; Khalil,

2002):

$$S(\mathbf{x}_1) - S(\mathbf{x}_0) \leq \int_{t_0}^{t_1} P_{\text{Tot},t} dt \leftrightarrow \dot{S}(x_t) \leq P_{\text{Tot},t}. \quad (5.8)$$

Proposition 5.3.1 (*Passivity of the proposed framework*). *The human-robot interaction framework of Fig. 5.6 is passive with respect to the energetic port $\langle \dot{\mathbf{y}}_{m,t}, \mathbf{F}_{h,t} \rangle$, where $\dot{\mathbf{y}}_{m,t}$ is the manipulator velocity.*

Proof. Let variables $\tilde{\mathbf{y}}$, $\dot{\tilde{\mathbf{y}}}$ and $\ddot{\tilde{\mathbf{y}}}$ be defined as:

$$\begin{aligned} \tilde{\mathbf{y}}_t &= \mathbf{y}_t - \mathbf{g} - \mathbf{E}\boldsymbol{\mu}(s_t) + \mathbf{E}\mathbf{g}_r, \\ \dot{\tilde{\mathbf{y}}}_t &= \frac{d\tilde{\mathbf{y}}_t}{dt} = \dot{\mathbf{y}}_t - \mathbf{E}\dot{\boldsymbol{\mu}}(s_t), \\ \ddot{\tilde{\mathbf{y}}}_t &= \frac{d\dot{\tilde{\mathbf{y}}}_t}{dt} = \ddot{\mathbf{y}}_t - \mathbf{E}\ddot{\boldsymbol{\mu}}(s_t), \end{aligned} \quad (5.9)$$

where $\mathbf{g}$ and $\mathbf{g}_r$ are the goals of the desired and parameterized demonstrated trajectories $\mathbf{y}_t$ and $\boldsymbol{\mu}(s_t)$, respectively. Using (5.9), Eq. (5.7) can be rewritten as:

$$\ddot{\tilde{\mathbf{y}}}_t + \mathbf{A}\dot{\tilde{\mathbf{y}}}_t + \mathbf{A}\mathbf{B}\tilde{\mathbf{y}}_t = \mathbf{F}_{h,t}, \quad (5.10)$$

The following storage function $S(\tilde{\mathbf{y}}_t, \dot{\tilde{\mathbf{y}}}_t)$ for the system in (5.6) and (5.10) can be considered:

$$S(\tilde{\mathbf{y}}_t, \dot{\tilde{\mathbf{y}}}_t) = \underbrace{\frac{1}{2}\dot{\tilde{\mathbf{y}}}_t^\top \dot{\tilde{\mathbf{y}}}_t}_{E_{K_1}} + \underbrace{\frac{1}{2}\tilde{\mathbf{y}}_t^\top \mathbf{A}\mathbf{B}\tilde{\mathbf{y}}_t}_{E_U} + \underbrace{\frac{1}{2}m\dot{s}_t^2}_{E_{K_2}}, \quad (5.11)$$

where E_{K_1} and E_U represent the kinetic and potential energies, respectively, stored in the Geometric DMP of Fig. 5.6, while E_{K_2} is the kinetic energy of the Phase Calculation block of Fig. 5.6. Differentiating the storage function $S(\tilde{\mathbf{y}}_t, \dot{\tilde{\mathbf{y}}}_t)$ in (5.11) with respect to time yields:

$$\dot{S}_t = \dot{\tilde{\mathbf{y}}}_t^\top [\ddot{\tilde{\mathbf{y}}}_t + \mathbf{A}\mathbf{B}\tilde{\mathbf{y}}_t] + m\dot{s}_t\ddot{s}_t. \quad (5.12)$$

Substituting $\ddot{\tilde{\mathbf{y}}}_t$ from (5.10) in (5.12) yields:

$$\begin{aligned} \dot{S}_t &= \dot{\tilde{\mathbf{y}}}_t^\top [-\mathbf{A}\dot{\tilde{\mathbf{y}}}_t - \mathbf{A}\mathbf{B}\tilde{\mathbf{y}}_t + \mathbf{F}_{h,t} + \mathbf{A}\mathbf{B}\tilde{\mathbf{y}}_t] + \\ &\quad + m\dot{s}_t\ddot{s}_t. \end{aligned} \quad (5.13)$$

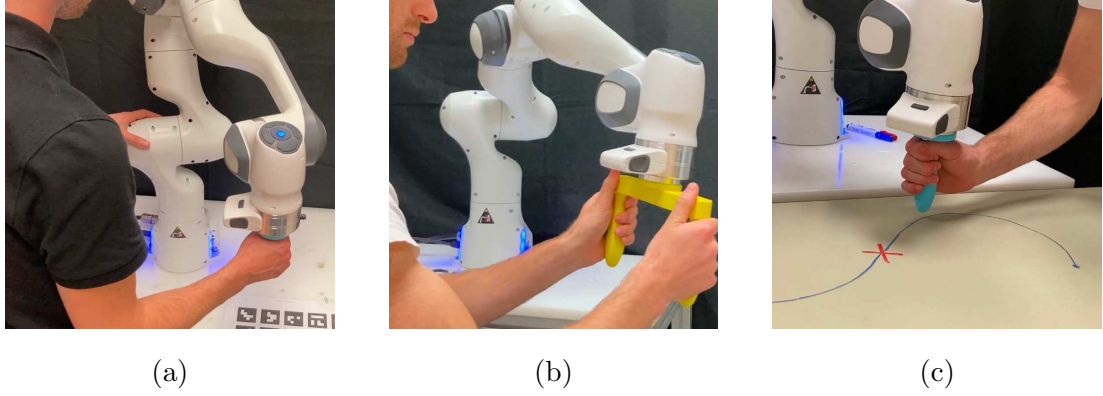


Figure 5.8: The considered experimental tests: (a) Internal/External rotation of the shoulder, (b) Bi-manual grabbing, (c) Stop at a point.

Substituting $\ddot{s}_t$ from (5.6) in (5.13) yields:

$$\begin{aligned} \dot{S}_t = & -\dot{\tilde{\mathbf{y}}}_t^\top \mathbf{A} \dot{\tilde{\mathbf{y}}}_t + \dot{\tilde{\mathbf{y}}}_t^\top \mathbf{F}_{h,t} + \\ & -d\dot{s}_t^2 + \dot{s}_t \boldsymbol{\mu}'(s_t)^\top \mathbf{E} \mathbf{F}_{h,t}. \end{aligned} \quad (5.14)$$

Finally, substituting $\dot{\tilde{\mathbf{y}}}_t$ from (5.9) in (5.14) and recalling that $\boldsymbol{\mu}'(s_t)\dot{s}_t = \dot{\boldsymbol{\mu}}(s_t)$ from (3.3) yields:

$$\begin{aligned} \dot{S}_t = & -\dot{\tilde{\mathbf{y}}}_t^\top \mathbf{A} \dot{\tilde{\mathbf{y}}}_t + [\dot{\mathbf{y}}_t - \mathbf{E} \dot{\boldsymbol{\mu}}(s_t)]^\top \mathbf{F}_{h,t} + \\ & -d\dot{s}_t^2 + \dot{\mathbf{y}}^{*T}(s_t) \mathbf{E} \mathbf{F}_{h,t}, \\ = & -\dot{\tilde{\mathbf{y}}}_t^\top \mathbf{A} \dot{\tilde{\mathbf{y}}}_t + \dot{\mathbf{y}}_t^\top \mathbf{F}_{h,t} - d\dot{s}_t^2. \end{aligned} \quad (5.15)$$

Considering the power $P_T = \dot{\mathbf{y}}_{m,t}^\top \mathbf{F}_{h,t}$ flowing through the energetic port $\langle \dot{\mathbf{y}}_{m,t}, \mathbf{F}_{h,t} \rangle$ and using (5.15), the passivity condition (5.8) is fulfilled. Indeed, assuming perfect tracking by the robot of the reference trajectory generated by the GDMP, i.e. $\mathbf{y}_{m,t} = \mathbf{y}_t$ and consequently $\dot{\mathbf{y}}_{m,t} = \dot{\mathbf{y}}_t$, the inequality

$$-\dot{\tilde{\mathbf{y}}}_t^\top \mathbf{A} \dot{\tilde{\mathbf{y}}}_t + \dot{\mathbf{y}}_t^\top \mathbf{F}_{h,t} - d\dot{s}_t^2 \leq \dot{\mathbf{y}}_{m,t}^\top \mathbf{F}_{h,t} \quad (5.16)$$

transforms into $-\dot{\tilde{\mathbf{y}}}_t^\top \mathbf{A} \dot{\tilde{\mathbf{y}}}_t - d\dot{s}_t^2 \leq 0$, which is always satisfied for $\mathbf{A} > 0$ and $b > 0$.

5.4 Experiments and Results

This section presents and discusses the experimental tests conducted to validate the human-robot interaction framework of Fig. 5.6.

Section 5.4.1 outlines the utilized setup and details the experimental tests performed for the tuning of parameters m and d in (5.6) through the practical stability analysis of Sec. 5.4.2. The results have shown the emergence of instability phenomena such as vibrations and oscillations despite the passivity proof of Sec. 5.3.1, which assumes perfect tracking of the GDMP trajectory. This motivates the introduction of the stability analysis performed in Sec. 5.4.2. Finally, Sec. 5.4.3 addresses the comparison of GDMP with other DMP solutions with reference to an insertion task, showing the enhanced performance of GDMP.

5.4.1 Experimental Setup

The experimental setup involves the use of a Franka Emika Panda robot in the GDMP framework of Fig. 5.6 for co-manipulation tasks. Three distinct experimental tests have been designed as illustrated in Fig. 5.8, requiring the user to manipulate the robot end-effector along the position trajectory $\mathbf{y}_t$ generated by the GDMP. The manipulator position control is developed in the MATLAB/Simulink environment. The experiments involved 12 participants of different gender, aged between 20 and 60, out of whom only three possessed prior expertise in robotics. In the test of Fig. 5.8(a), the user is required to perform an internal and external rotation of the shoulder in the upright position. In the test of Fig. 5.8(b), the user is required to lead the end-effector from the initial to the goal position using a bi-manual grabbing in a seated position. Finally, in the experiment of Fig. 5.8(c), the user is required to move the end-effector from the initial position to a given point located along the demonstrated trajectory. For each task, an expert user demonstrated the reference position trajectory $\mathbf{r}_t$ through kinesthetic guidance. The parameterized position trajectory $\boldsymbol{\mu}(s_t)$ has then been generated as depicted in Fig. 5.1.

5.4.2 Parameters Tuning through Practical Stability Analysis

In the GDMP of Eq. (4.43), the parameters $\alpha_x = \alpha_y = \alpha_z = 40$ and $\beta_x = \beta_y = \beta_z = 10$ have been assigned to the matrices $\mathbf{A}$ and $\mathbf{B}$, in order to ensure a sufficiently high stiffness in (5.10) guaranteeing $\mathbf{y}_t \approx \boldsymbol{\mu}(s_t)$. The choice of m and d in (5.6) is pivotal in practical applications. In order to extract first-attempt values for m and d , the Franka Emika Panda specifications have been considered: $\dot{\mathbf{y}}_{\text{MAX}} = 1.7 \text{ m/s}$, $\ddot{\mathbf{y}}_{\text{MAX}} = 13.0 \text{ m/s}^2$ and a maximum payload of 3 kg, which implies a force limit of $F_{\text{MAX}} = 30N$. Under the nominal condition

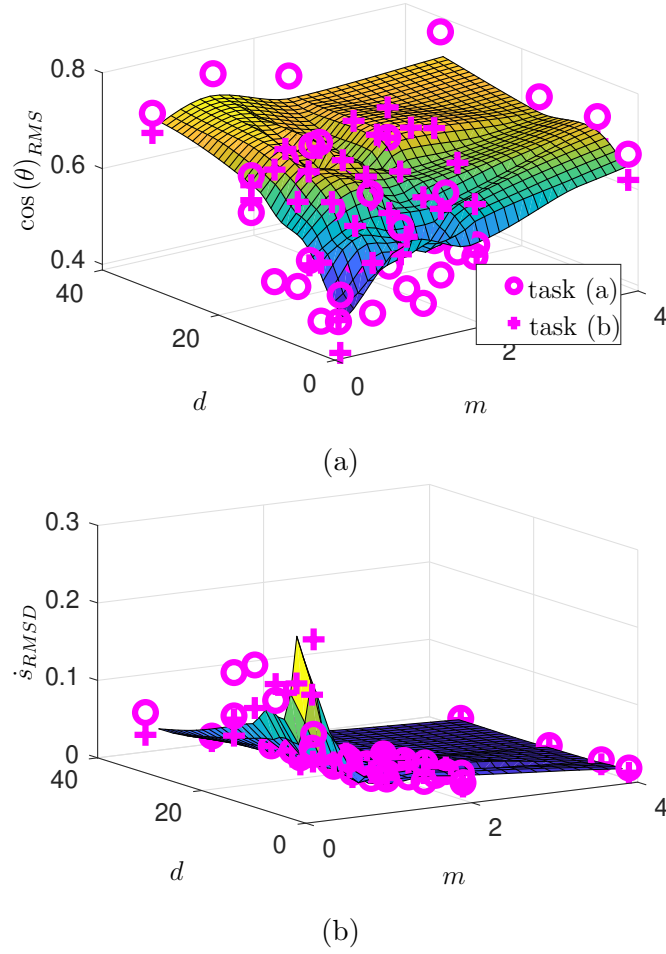


Figure 5.9: Trend of (a) external force application angle $\cos(\theta)_{RMS}$ and (b) peak-to-peak oscillations $\dot{s}_{RMSD}$ on the phase velocity.

$\mathbf{E} = \mathbf{I}_3$ and recalling (3.7), it follows that $\|\dot{\boldsymbol{\mu}}(s_t)\| = |\dot{s}_t|$ and, if the curvature is negligible, $\|\ddot{\boldsymbol{\mu}}(s_t)\| \approx |\ddot{s}_t|$ from (3.3). Consequently, the following specifications can be derived:

$$\dot{s}_{MAX} = 1.7 \text{ m/s} \quad \text{and} \quad \ddot{s}_{MAX} = 13.0 \text{ m/s}^2. \quad (5.17)$$

When the velocity $\dot{s}_t$ is sufficiently low, the following values of values of m and d can be determined from Eq. (5.6) and (5.17):

$$m = \frac{F_{MAX}}{\ddot{s}_{MAX}} \approx 2 \text{ kg} \quad \text{and} \quad d = \frac{\dot{s}_{MAX}}{\ddot{s}_{MAX}} \approx 17 \text{ N s/m}.$$

Starting from (5.18), the following ranges have been experimentally investigated:

$$\mathcal{N}_m = [0.2, 4] \text{ kg} \quad \text{and} \quad \mathcal{N}_d = [1.7, 34] \text{ N} \cdot \text{s/m}. \quad (5.18)$$

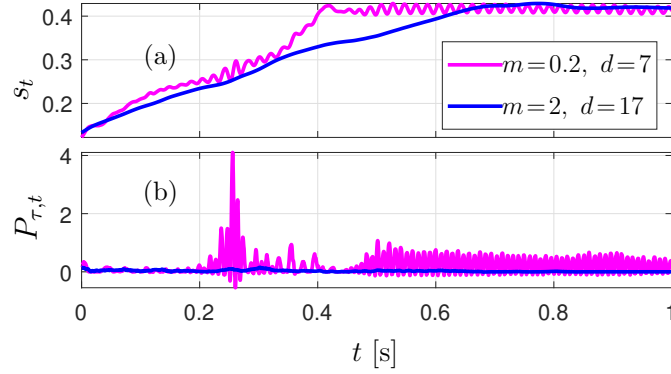


Figure 5.10: (a) Phase variable s_t and (b) power $P_{\tau,t} = F_{\tau,t}\dot{s}_t$ exchanged between human and the robot.

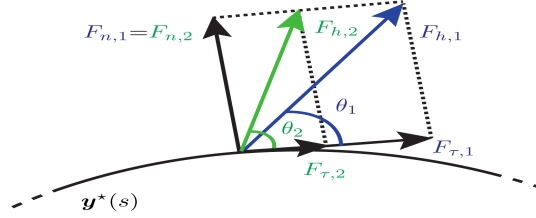


Figure 5.11: Variation of θ with respect to different forces $\mathbf{F}_h$.

The three tests in Fig. 5.8 have been executed by each user for 60 s using m and d in (5.18). The results of the tests of Fig. 5.8(a) and Fig. 5.8(b) are shown in Fig. 5.9. The metric in Fig. 5.9(a) is defined as follows:

$$\cos(\theta)_{RMS} = \text{rms}(\cos(\theta_t)), \text{ with } \cos(\theta_t) = \frac{F_{\tau,t}}{\|\mathbf{F}_{h,t}\|},$$

giving the normalized effort that the user has to make in order to move along the desired trajectory. This can be evinced with reference to the schematic representation of Fig. 5.11 showing that, if the tangential component is reduced from $F_{\tau,1}$ to $F_{\tau,2}$ for a given normal force component $F_{n,1} = F_{n,2}$, the angle θ increases and $\cos(\theta)$ decreases. Since $\cos(\theta)_{RMS} \in [0, 1]$, this metric evaluates the tangential force $F_{\tau,t}$ independently of the magnitude $\|\mathbf{F}_{h,t}\|$ associated with each particular user. The metric in Fig. 5.9(b) is defined as follows:

$$\dot{s}_{RMSD} = \text{rms}(\dot{s}_t - \dot{s}_{AVG,t}),$$

where $\dot{s}_{AVG,t}$ is the moving average of the function $\dot{s}_t$. This metric quantifies the average magnitude of the oscillations affecting the phase velocity $\dot{s}_t$.

The surface plot of Fig. 5.9(a) suggests that, as parameters m and d decrease, the user effort required to execute the task results to be lower as expected. How-

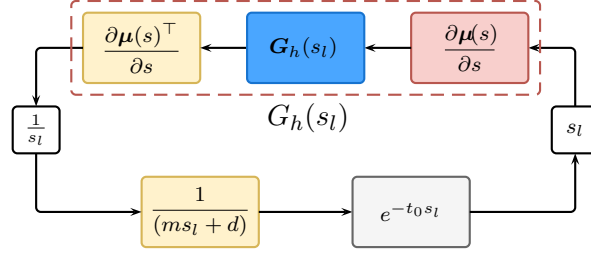


Figure 5.12: Linearized Human-robot interaction framework through kinesthetic guidance.

ever, as shown in Fig. 5.9(b), this also leads to higher oscillations affecting the phase velocity $\dot{s}_t$, which may cause the robot to stop because of violation of the safe limit constraints. These oscillations are also evident in Fig. 5.10, showing the results of the test in Fig. 5.8(c). Consequently, for lower values of m and d falling within the yellow peak in Fig. 5.9(b), the user's attempt to counteract the robot oscillations results in a larger recorded power $P_{\tau,t}$.

This oscillating behavior can be explained using the following practical stability analysis on the system of Fig. 5.6 based on the Lyapunov's indirect method (Khalil, 2002). Although the robot is typically assumed to perfectly track the desired reference, the bandwidth of the position control Fig. 5.6 is actually finite in practical applications, leading to the introduction of a finite delay t_0 between the robot position $\mathbf{y}_{m,t}$ and the DMP reference $\mathbf{y}_t$. Although providing an accurate description of the human is non-trivial (Ficuciello et al., 2014), the following simplified spring-damper model exhibiting isotropic behavior across all spatial directions is employed in this work:

$$\mathbf{F}_h = G_h(s_l) \mathbf{I}_3 \dot{\mathbf{y}}_m, \quad \text{where} \quad G_h(s_l) = \frac{B_h s_l + K_h}{s_l}$$

and where s_l is the Laplace complex variable. Assuming that the GDMP behaves as an ideal trajectory generator, i.e. $\mathbf{y}_t \approx \boldsymbol{\mu}(s_t)$, and linearizing the system of Fig. 5.6 around the equilibrium point $(s_e, \dot{s}_e) = (\bar{s}, 0)$, where $\bar{s}$ represents the stop position, the scheme illustrated in Fig. 5.12 can be derived, which is characterized by the following input-output transfer function:

$$G_{IO}(s_l) = \frac{G_{LG}(s_l)}{1 + G_{LG}(s_l)}, \quad \text{where} \quad G_{LG}(s_l) = \frac{G_h(s_l) e^{-t_0 s_l}}{m s_l + d} \quad (5.19)$$

is the loop-gain function of the system. The qualitative Nyquist diagram of $G_{LG}(s_l)$ is shown in Fig. 5.13. The Nyquist criterion states that the closed-

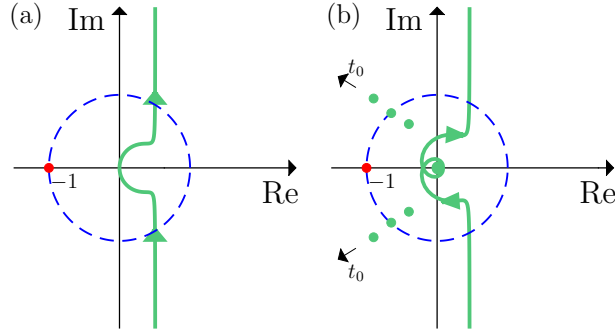


Figure 5.13: Nyquist diagram of the loop-gain function $G_{LG}(s_l)$ without (a) and with (b) finite delay t_0 .

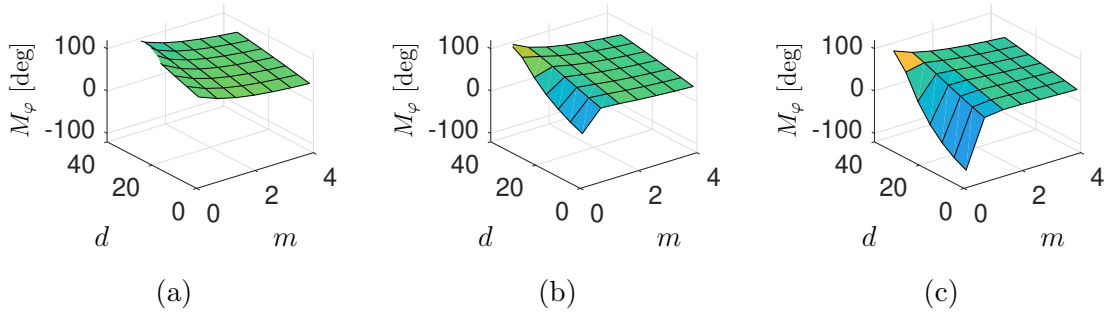


Figure 5.14: Phase margin M_φ of the loop-gain function $G_{LG}(s_l)$ as a function of m , d for different values of the delay t_0 : $t_0=0$ s (a), $t_0=0.01$ s (b), and $t_0=0.02$ s (c).

loop system is asymptotically stable as long as the Nyquist contour of $G_{LG}(s_l)$ does not touch or encircle the critical point $-1 + j0$ (Ogata & Yang, 2002). Based on this, Fig. 5.13 clearly shows that a large finite delay t_0 may lead to an unstable system. The stability margin M_φ (Ogata & Yang, 2002) of the closed-loop human-robot interaction framework is shown in Fig. 5.14. Fig. 5.14(a) shows that $M_\varphi > 0$ is verified $\forall m, d$ when $t_0 = 0$, proving the system stability as predicted in the passivity proof of Sec. 5.3.1. As t_0 increases, Fig. 5.14(b) and Fig. 5.14(c) show that the system becomes unstable for small values of m and d because $M_\varphi < 0$. This well agrees with the oscillations affecting the phase variable s_t in the experimental results of Fig. 5.9(b) and Fig. 5.10. It can be concluded that the parameters m and d in the phase calculation equation (5.6) play an important role in determining the effort perceived by the user when moving along the desired trajectory, and they need to be sufficiently large in order ensure the stability of the considered human-in-the-loop system.

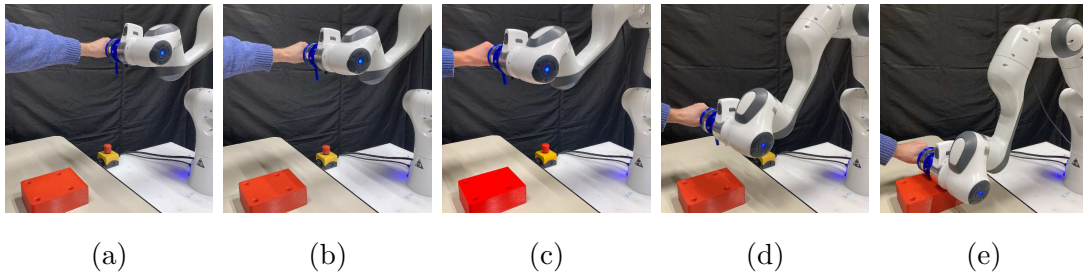


Figure 5.15: Snapshots of the insertion task. During the demonstration by kinesthetic teaching, the user moves horizontally the robot over the target hole from its start position (a)-(b), then the user stops for few seconds (c) and moves vertically the end-effector tip toward the hole in the red box to complete the insertion task (d)-(e).

5.4.3 Insertion Task

To further evaluate the GDMP architecture in Fig. 5.1, a final co-manipulation task was conducted using $m = 2$ and $d = 17$ for robustness, as discussed previously. The task involved the guidance of the end-effector handle in order to insert its tip into one of the four holes in the red box shown in Fig. 5.15.

The insertion task was initially demonstrated via kinesthetic guidance. Starting from the position in Fig. 5.15(a), the robot was guided above the target hole (Fig. 5.15(c)), paused for few seconds, and then moved into the hole to complete the task (Fig. 5.15(d)-(e)). During the execution, the goal position of the DMP architecture was varied in order to select each of the four holes of the box. The GDMP architecture was compared with three other DMP methodologies: reversible DMP (RDMP) (Sidiropoulos & Doulergi, 2021), speed-scaled DMP (SSDMP) (Nemec, Likar, et al., 2018), and arc-length DMP (ALDMP) (Gaspar et al., 2018). Each approach was tested three times for all four holes of the box, and the results are presented in Fig. 5.16- 5.17.

Figure 5.16 shows the 3D plots of the task execution for each DMP architecture. The position trajectories are color-coded based on the magnitude of the measured force norm $\|\mathbf{F}_h\|$ during the execution, where the force data have been acquired using a Schunk FT-AXIA force/torque sensor. To allow the comparison across the different DMP methods, all executions were normalized to their maximum measured force. A colorbar for the force magnitude is provided in Fig. 5.16(e).

For what concerns the RDMP case shown in Fig. 5.16(a), a high force demand is evident due to the lack of arc-length parametrization, which prevents proper tangential projection of the forces along the trajectory. This issue becomes critical near the stopping interval recorded during the demonstration, where the tangen-

tial component is undefined, making it impossible for the user to complete the insertion task. In the SSDMP case shown in Fig. 5.16(b), although a higher force demand is observed, the user successfully completes the task. This is because the SSDMP autonomously progresses its phase, actively guiding the user. However, the tests have shown that the SSDMP's phase velocity adjustment based on the user's feedback lacks reactivity, leading to difficulty switching between active and passive guidance, and resulting in a higher force demands during the execution.

Significantly lower force measurements are observed for the ALDMP case shown in Fig. 5.16(c). This is because the arc-length parametrization used by ALDMP allows a consistent computation of the tangential component of the desired curve at each instant, ensuring the correct projection of the measured force $\mathbf{F}_h$. However, the color code shows frequent force variations, as the ALDMP algorithm does not maintain a constant tangential norm, unlike the proposed SS algorithm. In the GDMP case shown in Fig. 5.16(d), the SS algorithm ensures that the tangential component of the geometric curve has unitary norm, as guaranteed by the filtering action obtained in property (3.7) which ensures the regularity of the curve. This results in consistent projection of $\mathbf{F}_h$ throughout the task, reducing the user's effort when moving the robot, as evident from the trajectory color code.

The task execution is further analyzed in the results of Fig. 5.17. The statistical results are reported in Fig. 5.17(a) based on: (i) the norm of the displacement error $\|\mathbf{e}\|$, where $\mathbf{e}_t = \mathbf{y}_t - \mathbf{y}_{m,t}$; and (ii) the difference $\|\mathbf{F}_h\| - |F_\tau|$, representing the force used to keep the end-effector near the trajectory $\mathbf{y}$ provided by the DMP solution. The proposed GDMP achieves more accurate position tracking, as shown in Fig. 5.16, thanks to its more effortless guidance compared to other DMP methods. Additionally, the lower difference $\|\mathbf{F}_h\| - |F_\tau|$ indicates that, like ALDMP, the GDMP provides more intuitive guidance. This is because a greater portion of the applied force is projected tangentially to the curve, making the robot guidance easier to follow.

The limitations of the analyzed DMP approaches when compared with GDMP are summarized in the table in Fig. 5.17(b), where “Reversibility”, “Insertion Task Completion”, and “Managing Pausing Intervals” refer to the DMP's ability to be reversible, complete the insertion task, and handle pausing intervals recorded during the demonstration, respectively. The RDMP fails to accomplish the task due to its reliance on time parametrization, which prevents it from managing pausing intervals. The SSDMP and ALDMP lack reversibility, making them unsuitable for co-manipulation tasks. While the SSDMP can handle pausing

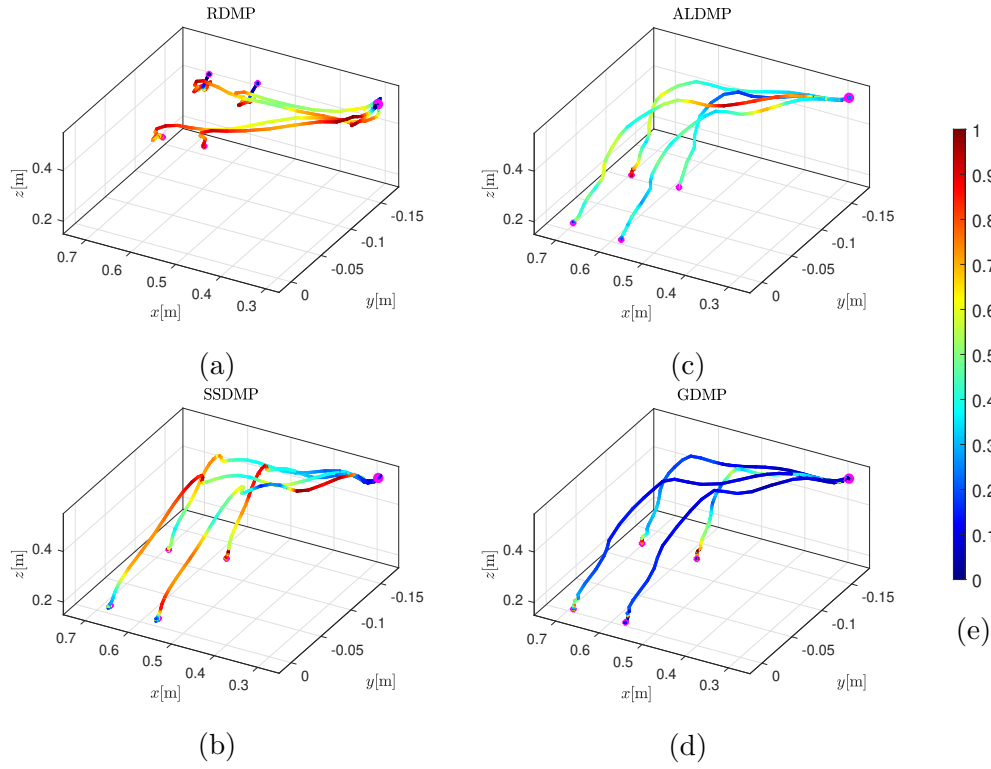
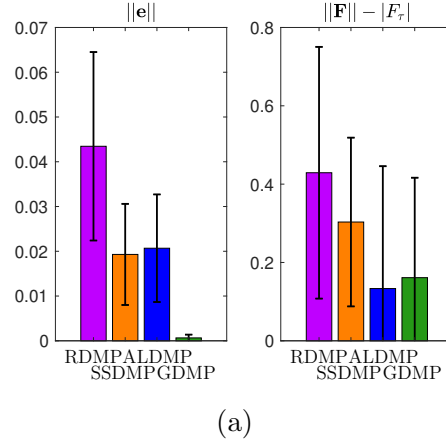


Figure 5.16: Plots of the recorded trajectories when varying the goal position. Each position trajectory has been reported with varying color, which encodes the measured force norm $\|\mathbf{F}_h\|$ normalized to the maximum measured component inside the same experimental trial.

intervals by actively guiding the user and resuming movement after the pause, its execution remains time-dependent on the demonstration, posing limitations. In contrast, the proposed GDMP satisfies all the requirements, demonstrating its enhanced performance over the other approaches.

5.4.4 Discussion

A potential application for reversible DMP is its use as a virtual fixture (Amor et al., 2014; Raiola, Lamy, & Stulp, 2015; Restrepo et al., 2017). The latter represents a predefined path constraining the robot movement without a specified timing law. The phase variable s_t can be adjusted arbitrarily, allowing the system to evolve along the predefined paths. However, the DMP architecture imposes some limitations as the recorded reference trajectories are affected by the time dependence observed during the task demonstrations (Billard et al., 2008). The same issue emerges when multiple demonstrations of the same task are recorded with different timing laws. In this case, techniques such as Dynamic Time Warping (DWT) (Berndt & Clifford, 1994) allow to align the different



(a)

	Reversibility	Insertion Task Completion	Managing Pausing Intervals
RDMP	✓	✗	✗
SSDMP	✗	✓	✓
ALDMP	✗	✓	✓
GDMP	✓	✓	✓

(b)

Figure 5.17: Comparison of GDMP with other DMP solutions in addressing the insertion task. (a) Obtained results in term of average norm of the error displacement $\|e\|$, representing the spatial accuracy, and the average norm of the non-tangential force component $\|\mathbf{F}_h\| - |F_\tau|$, which reflects the undesired forces orthogonal to the task direction. (b) final evaluation table.

demonstrations and compute a unique reference motion, such that the related skill is extracted (Calinon et al., 2007; Hovland et al., 1996).

The Canonical System of the DMP provides information about the evolution of the forcing term which, in sequence, shapes the attractor landscape of the Transformation system to get the desired output dynamics. While time modulation properties are ensured by the standard DMP formulation as mentioned in Chapter 4.3- 4.3.2, in the literature we found no unifying framework that gathers in a compact way multiply phase profiles without considerably change the standard DMP formulation. In this Chapter, such problem is addressed by elaborating GDMP, which leverages a spatial sampling algorithm to decouple the learned curve from its timing law imposed during the teaching phase. This is ensured by a geometric interpretation of the phase variable given by the arc-length parametrization of the task curve.

Such framework was considered, as a show case, in a simulated environment in

Section 5.2. Here we calculate the optimal jerk command of the phase variable in order to minimize the task execution period subject to constraint on the maximum allowed speed and acceleration, respectively on the joint and work space of the robot. The optimal phase profile can thus be launched as an input to the forcing term, guaranteeing a fast execution of the intended task while respecting the imposed constraints.

By the design of a custom phase law, many real-world experiments have been conducted using the values of parameters m and d of (5.6) as knot variables to vary the execution of a target task, which evolution is modulated directly by the user. As highlighted by the results of Section 5.4, while the proposed architecture guarantees passivity on paper, the assumptions made for the model might introduce approximation errors, which in turns do not ensure stability during the task execution.

This was proven qualitatively from the measurements recorded with different users and different values combinations of m and d . Therefore we quantified, through an analytical study, a procedure to choose properly those parameters to ensure stability and passivity of the human-in-the-loop architecture. This could additionally be seen as an insightful procedure to enclose the concepts of safety and stability, which is non-trivial as the human is unpredictable, and useful in human-robot collaborative tasks.

Enhancing human-robot interaction: a minimum-jerk approach

In many guiding applications, allowing compliance with respect to the virtual fixture (VF) curve $\boldsymbol{\mu}$ ensures a more natural and intuitive interaction with the user (Amirabdollahian et al., 2002; Raiola, Lamy, & Stulp, 2015; Sciavicco & Siciliano, 2012). A common approach in such cases, is to adjust the reference position on $\boldsymbol{\mu}$ to the point that minimizes the displacement with the actual position of the robot. In this way, the robot is able to follow the user movements, whilst imposing the virtual constraint (Selvaggio et al., 2018). However, this procedure has not yet been extended to positions in the workspace that share the same distance to multiple points on the VF, here called *Euclidean distance singularities* (EDSs) (Braglia, Calinon, & Biagiotti, 2024). This problem is illustrated in Fig. 6.1.

This chapter deals with techniques where the time-law s_t is obtained as the solution of an optimization problem, i.e. by solving the minimum distance problem (Reynoso-Mora et al., 2013; Verscheure et al., 2009). In particular, we observed that many virtual fixture applications rely on distance-based methods, selecting the point on the curve $\boldsymbol{\mu}(s)$ that minimizes the Euclidean distance from the robot's end effector (EE) position $\mathbf{y}$ (Bischof et al., 2016; Girgin et al., 2023; Selvaggio et al., 2018; L. Zhang et al., 2020a). Finding an analytical solution to calculate the minimum distance point $\boldsymbol{\mu}(s^*)$ is generally non-trivial. Nevertheless, the presence of techniques such as the Gauss-Newton (GN) algorithm, simplifies the problem to the computation of the phase s_t^* that minimizes the residual $\|\mathbf{y}_t - \boldsymbol{\mu}(s_t^*)\|$ (Y. Wang, 2012).

The GN algorithm is known to be computationally efficient and was applied in various robotic applications (Calinon, 2023b; Khatib et al., 2022). However, its

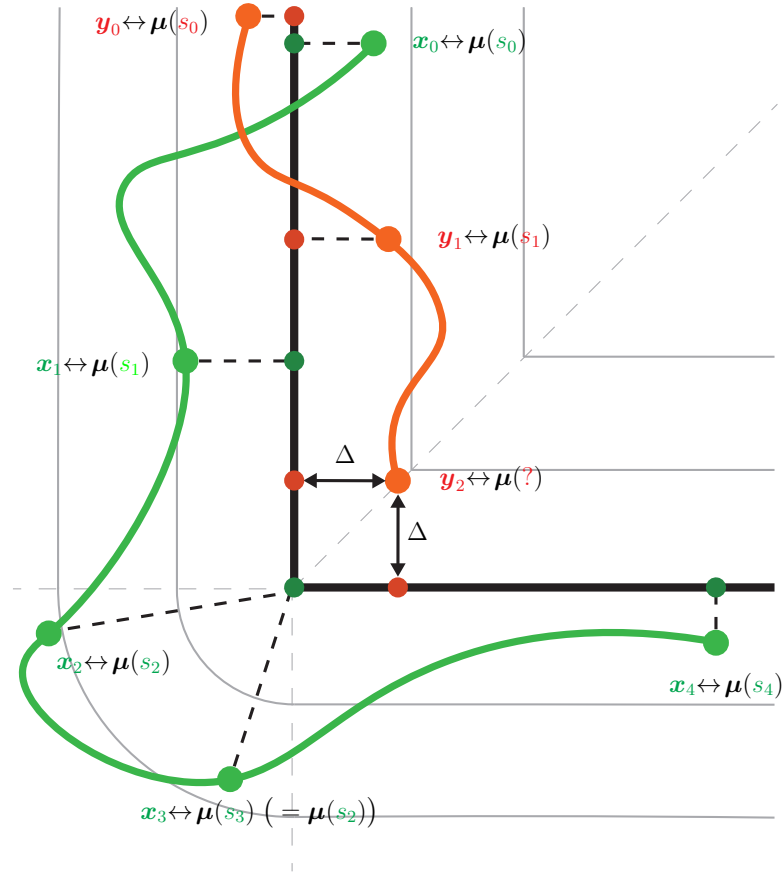


Figure 6.1: 2D visualization of an Euclidean distance singularity. Continuous gray lines represent the iso-lines with respect to the constraint path $\mu(s)$ depicted in black. While a distance-based method could correctly update the reference position $\mu(s_t)$ for the green trajectory, it fails to find a stable solution for the red trajectory, as y_2 has the same distance Δ from 2 points in $\mu(s)$.

use to update s_t may not lead to a convex formulation for the minimum distance problem (Luenberger, Ye, et al., 1984). This condition is referred as EDS and it can be proven both analytically and experimentally that approaching an EDS might produce abrupt changes in the phase velocity $\dot{s}_t$.

Controlling the phase velocity is common in planning algorithms, and the outcome solution usually produce a time-optimal trajectory that exploits the maximum accelerations/torques given the constraints of the robot actuators (Verschueren et al., 2009). This could potentially result in acceleration profiles with a high rate of change, which can stress the actuators leading to long term damage (Palleschi et al., 2019). In these cases, enhancing smoothness is a good compromise between minimum time task execution, and acceleration noise reduction. To achieve so, it is a common practice to minimize the jerk of the planned trajectories (Bianco & Ghilardelli, 2017; Piazzini & Visioli, 2000). Minimum jerk

trajectories have been widely investigated also in the reproduction of human arm movements (Biess et al., 2007; Sharkawy, 2021). As demonstrated in (Todorov & Jordan, 1998), minimum jerk trajectories can faithfully reproduce the speed profile of arm movements in reaching tasks or curve tracking, making them suitable for human-robot interaction scenarios (Amirabdollahian et al., 2002). This motivates the interest in proposing a novel methodology for the online phase update in guiding VFs, exploiting a jerk-controlled framework based on minimum distance, while ensuring robustness against EDSs.

6.1 Preliminaries

The aim is to constrain the robot over a predefined reference path $\boldsymbol{\mu}$ in the task space. Once established, the operator can use the robot's guidance to navigate along this path. Details of this mechanism are briefly introduced in the following paragraphs as preliminaries to better understand the next sections.

6.1.1 Definition of the virtual constraint

To define the constraint $\boldsymbol{\mu}$, we first kinesthetically move the robot end effector to demonstrate the target curve. The recordings are then filtered using the spatial sampling (SS) algorithm introduced in Chapter 3.2, to extract the geometrical path information regardless of the timing introduced during the demonstration. This associates the recordings $\mathbf{r}_t = [r_{x,t}, r_{y,t}, r_{z,t}]^\top$ with their filtered counterparts $\mathbf{r}_{\Delta,k} = [r_{x,\Delta,k}, r_{y,\Delta,k}, r_{z,\Delta,k}]^\top$ and arc-length parameter $s_k \in [0, L]$, with $k = 1, \dots, M$ and $L = \Delta M$ representing the curve length $\mathbf{r}_\Delta$.

Then, $\boldsymbol{\mu}(s)$ is expressed using basis functions approximation as introduced in Chapter 4.1, representing the signal $\mathbf{r}_{\Delta,k}$ as a weighted sum of nonlinear terms, namely (Calinon, 2020b)

$$\boldsymbol{\mu}(s_t) = \sum_{i=1}^N \omega_i \phi_i(s_t) = \boldsymbol{\omega}^\top \boldsymbol{\phi}(s_t). \quad (6.1)$$

In (6.1), the constraint curve is formed by summing N basis functions ϕ_i , each weighted by ω_i , found via a Least-Squares solution (Ijspeert et al., 2013). The basis functions $\phi_i(s_t)$ can be chosen in different ways, in this case we used Bernstein polynomials previously presented in Chapter 4.1.2. Basis function approximation provides an analytical expression of the demonstrated path and a prior knowledge of its continuity (thus continuity about its derivatives), which was helpful

in the formulation of the framework introduced in this chapter. Again, the phase variable is denoted by s_t , but thanks to the SS algorithm it carries an additional meaning of being the arc-length parameter of the curve $\boldsymbol{\mu}$. Note that, because of (3.7), $\boldsymbol{\mu}'(s_t)$ will always be well-defined and different from zero (Biagiotti & Melchiorri, 2008; Sciavicco & Siciliano, 2012).

6.1.2 Tracking problem statement

Let $\boldsymbol{\mu}(s) \in \mathbb{R}^3$ be the path constraint parametrized with respect to the arc-length parameter s , and $\mathbf{y}_t \in \mathbb{R}^3$ be the Cartesian position of the robot's end effector (EE). The objective of limiting the EE movement along the curve $\boldsymbol{\mu}(s)$ can be formulated as a tracking problem, that is

$$s_t^* = \arg \min_{s_t \in [0, M\Delta]} \|\mathbf{y}_t - \boldsymbol{\mu}(s_t)\|^2. \quad (6.2)$$

The optimal problem in (6.2) equals to update s_t at every instant t in order to find the closest point $\boldsymbol{\mu}(s_t^*)$ to $\mathbf{y}_t$. From (Luenberger, Ye, et al., 1984), the existence of a solution is guaranteed if:

$$(\mathbf{y}_t - \boldsymbol{\mu}(s_t^*))^\top \boldsymbol{\mu}'(s_t^*) = 0, \quad \text{and} \quad (6.3a)$$

$$\|\boldsymbol{\mu}'(s_t^*)\|^2 - (\mathbf{y}_t - \boldsymbol{\mu}(s_t^*))^\top \boldsymbol{\mu}''(s_t^*) > 0. \quad (6.3b)$$

The conditions in (6.3) leads to the following geometrical intuition. On one hand, the first order necessary condition in (6.3a) states that the optimal s_t^* lies on the point $\boldsymbol{\mu}(s_t^*)$ whose tangential component $\boldsymbol{\mu}'(s_t^*)$ is orthogonal to the EE position $\mathbf{y}_t$. From (3.7), $\boldsymbol{\mu}'(s_t)$ represents the tangential unit vector to the curve. On the other hand, the second-order sufficient condition in (6.3b) guarantees the convexity of (6.2) if the Hessian of the cost function $\|\mathbf{y}_t - \boldsymbol{\mu}(s_t)\|^2$ is positive definite. In particular, given the constraint $\boldsymbol{\mu}(s)$ for $s \in [0, L]$, at every instant t a feasible subset $\boldsymbol{\chi}_t = \{\bar{\mathbf{y}}_t \in \mathbb{R}^3 : (6.3b) \text{ holds}\}$ can be deduced (Bischof et al., 2016). Equation (6.3) implies $\boldsymbol{\mu} \in C^2$, which can be ensured by a proper choice of the basis functions in (6.1) (Calinon, 2020b).

6.2 Updating Phase Variable with Gauss-Newton algorithm

As proposed in (Calinon, 2023b), one common technique for solution of the optimization problem in (6.2) comes from the Gauss-Newton (GN) algorithm (Y. Wang, 2012). In this case, the evolution of the arc-length parameter s_t is given by $s_{t+1} = s_t + \Delta s$, with the term Δs being the GN update. Defining the residual $\mathbf{f}(s_t) = \mathbf{y}_t - \boldsymbol{\mu}(s_t)$, the GN algorithm provides an analytical solution for the computation of Δs , that is

$$\begin{aligned} \Delta s_t &= -\mathbf{H}(s_t)^{-1} \mathbf{g}(s_t), \quad \text{with} \\ \mathbf{g}(s_t) &= 2\mathbf{J}_f^\top \mathbf{f}(s_t), \quad \mathbf{H}(s_t) \approx 2\mathbf{J}_f^\top \mathbf{J}_f. \end{aligned} \quad (6.4)$$

In (6.4), $\mathbf{J}_f \in \mathbb{R}^{3 \times D}$ defines the Jacobian matrix for $\mathbf{f}(s_t) \in \mathbb{R}^3$, while $\mathbf{g} \in \mathbb{R}^D$ and $\mathbf{H} \in \mathbb{R}^{D \times D}$ are the gradient vector and the Hessian matrix of the cost function with respect to the residual $\mathbf{f}(s_t)$, respectively. For our tracking problem we can assume $\mathbf{J}_f(s_t) = -\boldsymbol{\mu}'(s_t)$. The advantage of the GN algorithm is that, despite being a second order technique, it converges in few steps. This is possible as the Hessian can be easily estimated from the Jacobian in (6.4), which guarantees its positive definiteness (Calinon, 2023b; Y. Wang, 2012).

Note that the GN algorithm neglects the second-order derivative term of the Hessian in (6.3b). However, when this term cannot be ignored, the GN algorithm may experience slow convergence. From (3.7) one can write (6.3b) as

$$1 - (\mathbf{y}_t - \boldsymbol{\mu}(s_t))^\top \boldsymbol{\mu}''(s_t) > 0. \quad (6.5)$$

Given the geometry of curves, we have that $\boldsymbol{\mu}''(s_t) = \mathbf{n}(s_t)\kappa(s_t)$, where $\mathbf{n}$ and κ define the normal versor and the curvature, respectively, calculated at s_t (Sciavicco & Siciliano, 2012; Toponogov, 2006). In particular, observing that $r(s_t) = 1/\kappa(s_t)$ expresses the radius of the osculating circle at the point $\boldsymbol{\mu}(s_t)$ (see Appendix A.2), the condition in (6.5) can be written as

$$(\mathbf{y}_t - \boldsymbol{\mu}(s_t))^\top \mathbf{n}(s_t) < r(s_t). \quad (6.6)$$

Equation (6.6) provides the following geometric intuition: given the reference $\boldsymbol{\mu}(s_t)$, the center of the osculating sphere defines a boundary for the deviation of the EE position $\mathbf{y}_t$, out of which the second order condition for the optimality in (6.5) is violated. Therefore, the optimal problem in (6.2) no longer admits a

unique solution, as it loses convexity due to the fact that the Hessian matrix represented in (6.3b) ceases to be positive definite (Calinon, 2023b; Luenberger, Ye, et al., 1984).

Notably, in (Bischof et al., 2016) authors demonstrate that the time derivative of the optimal s_t^* in (6.2) can be written as

$$\dot{s}_t^* = \frac{\boldsymbol{\mu}'(s_t^*)^\top \dot{\mathbf{y}}_t}{1 - (\mathbf{y}_t - \boldsymbol{\mu}(s_t^*))^\top \boldsymbol{\mu}''(s_t^*)}. \quad (6.7)$$

Given the condition in (6.5), it can be evinced from (6.7) that the limit case $(\mathbf{y}_t - \boldsymbol{\mu}(s_t))^top \boldsymbol{\mu}''(s_t) \rightarrow 1$ implies $\dot{s}_t^* \rightarrow \infty$. Hence, small values of $\dot{\mathbf{y}}_t$ results in large derivatives of $\dot{s}_t^*$ when close to the center of the osculating circle calculated at $\boldsymbol{\mu}(s_t^*)$. In our case, as we allow compliance in the directions outside the path constraint $\boldsymbol{\mu}(s)$, this can compromise the contact with the robot and the human. These considerations provide an analytical explanation of the constraint imposed in (L. Zhang et al., 2020a) and lead to the definition of Euclidean distance singularity.

Remark 1. *Given the optimal problem in (6.2), with $\bar{s}$ being arc-length parameter of the curve $\boldsymbol{\mu}(s) \in \mathbf{R}^n$, we denote with Euclidean distance singularity (EDS) the set:*

$$EDS = \left\{ \mathbf{y} \in \mathbf{R}^n : \begin{array}{l} (\mathbf{y} - \boldsymbol{\mu}(\bar{s}))^\top \boldsymbol{\mu}'(\bar{s}) = 0, \\ (\mathbf{y} - \boldsymbol{\mu}(\bar{s}))^\top \mathbf{n}(\bar{s}) \geq r(\bar{s}) \end{array} \right\}. \quad (6.8)$$

The geometric intuition is that if we move along the normal direction $\mathbf{n}(\bar{s})$ by a distance greater than the radius of the osculating circle at $\boldsymbol{\mu}(\bar{s})$ (as shown in Fig. 6.2), the optimization problem in (6.2) degenerates into multiple local solutions.

EDS: Simulation

Figure 6.3 provides a 2D example. The left subplot shows the osculating circle γ at $\boldsymbol{\mu}(s_{\bar{t}})$ in dashed gray. The right subplot depicts the cost functions $c_{\mathbf{p}_1}$, $c_{\mathbf{p}_2}$, and $c_{\mathbf{p}_3}$ in (6.2) concerning the EE position $\mathbf{y}_{\bar{t}}$ relative to points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. Note that the cost function loses local convexity once the deviation reaches the center of the osculating circle $\mathbf{p}_3$. Note that, for curves with sharp corners, using a basis function approximation as in (6.1) ensures the continuity of the geometrical path and the Jacobian in (6.4) (Biagiotti & Melchiorri, 2008).

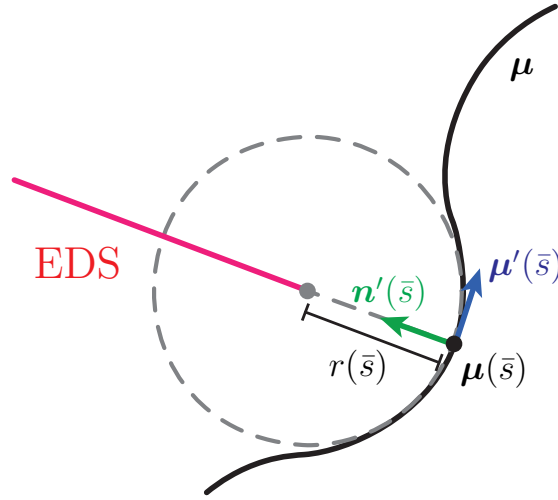


Figure 6.2: EDS visualization

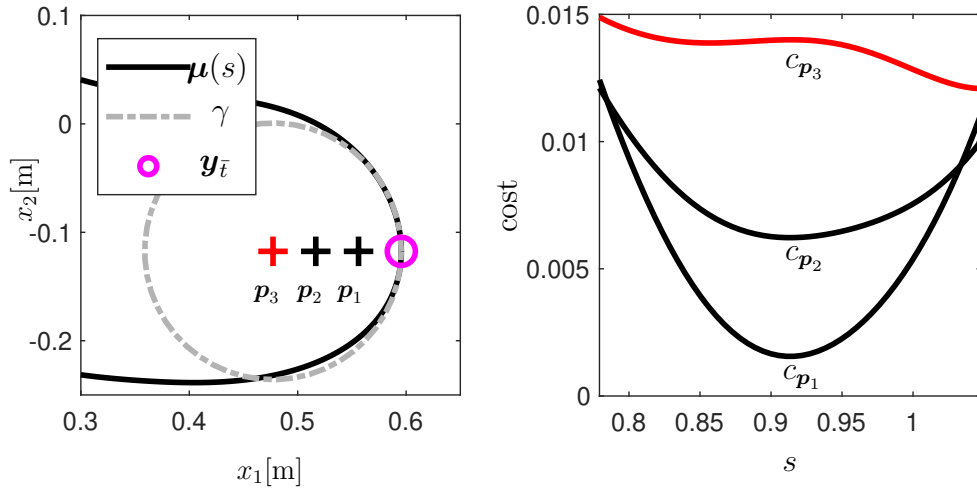


Figure 6.3: Trend of the cost in (6.2) for different path deviations.

EDS: Experimental proof

The existence of EDS was investigated using a simple experiment with the manipulator, as shown in Fig. 6.4. The encoded reference, depicted as dashed black in Fig. 6.5(a), includes a smooth corner region where the osculating circle radius decreases, increasing the likelihood of encountering an EDS. Experiments revealed that this proximity induces abrupt oscillations in the robot, compromising safe and natural guiding during the co-manipulation exercise.

These oscillation intervals, highlighted in red, are clearly visible in Fig. 6.5(b), which illustrates the phase variable s_t behavior throughout the exercise. To address this, a suitable controller should mitigate these oscillations by smoothing the jerk of the generated reference $\mu(s_t)$, as detailed in the following sections.

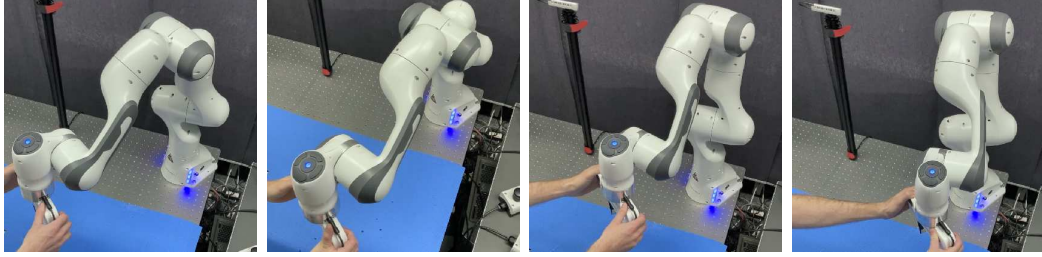


Figure 6.4: Snapshots of experimental results recorded on the Panda by Franka Emika robot on a virtual guiding task updating the phase variable with Gauss-Newton algorithm.

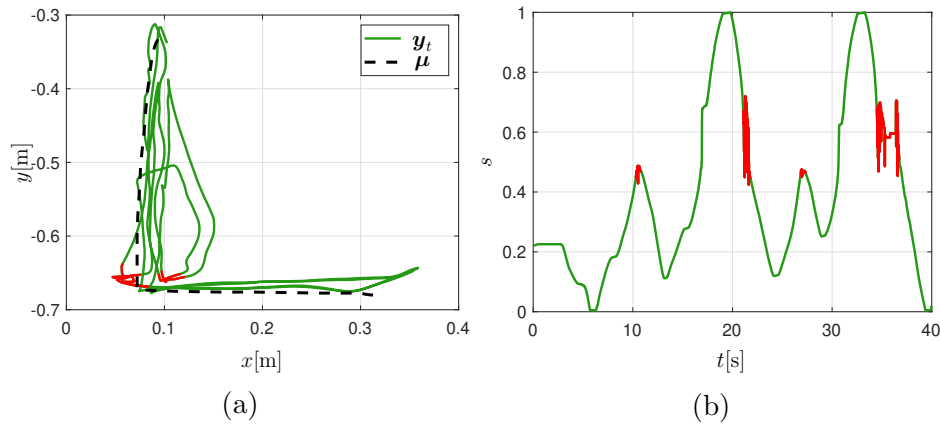


Figure 6.5: Robot experimental results: (a) the recorded robot position is shown in green and in red to highlight the intervals where undesired vibrations appeared; (b) the corresponding phase profile is plotted with the same color code of (a).

6.3 A minimum-jerk approach to handle EDS

With reference to equation (6.7), the proximity of the EE position $\mathbf{y}_t$ to EDSs could lead to high velocities $\dot{s}_t$. One way to mitigate such undesired behaviour, is to formulate the optimal problem in (6.2) such that we can control the damping on the velocity term. This is possible by considering minimum-jerk trajectories. Moreover, many results in literature indicate that this model is well-suited for human-robot co-manipulation tasks (Amirabdollahian et al., 2002; H. Chen et al., 2024; Todorov & Jordan, 1998).

Minimum-jerk approaches highlight that the velocity profile of hand movements is established by minimizing its squared jerk (Biess et al., 2007). Authors in (Meirovitch et al., 2016) extend the minimum jerk formulation to the so-called *jerk-accuracy* (JA) model, where the tracking precision and the control action is modulated with the help of a Lagrange multiplier. We here formulate the JA model as a linear quadratic tracking (LQT) problem for discrete-time systems, following the consideration made in Chapter 4.4. In doing so, the control com-

mand is fed to a chain of three discrete-time integrators, i.e. $u_t = \ddot{s}_t$, such that the reference $\boldsymbol{\mu}(s_t)$ stays aligned with the EE position $\mathbf{y}_t$. The problem becomes

$$\min_{u_t} \mathbf{f}(\mathbf{y}_t, \mathbf{s}_t)^\top \mathbf{Q} \mathbf{f}(\mathbf{y}_t, \mathbf{s}_t) + u_t^\top R u_t, \quad (6.9)$$

with $\mathbf{f}(\mathbf{y}_t, \mathbf{s}_t) = [\mathbf{e}_t^\top, \dot{\mathbf{e}}_t^\top, \ddot{\mathbf{e}}_t^\top]^\top$, $\mathbf{e}_t = \mathbf{y}_t - \boldsymbol{\mu}(s_t)$, subject to

$$\underbrace{\begin{bmatrix} s_{t+1} \\ \dot{s}_{t+1} \\ \ddot{s}_{t+1} \end{bmatrix}}_{\mathbf{s}_{t+1}} = \underbrace{\begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} s_t \\ \dot{s}_t \\ \ddot{s}_t \end{bmatrix}}_{\mathbf{s}_t} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \Delta t \end{bmatrix}}_{\mathbf{B}} u_t. \quad (6.10)$$

In (6.9), $\mathbf{s}_t = [s_t, \dot{s}_t, \ddot{s}_t]^\top$, $s_t \in [0, \Delta M]$, Δt is the sampling time, $\mathbf{Q} = \text{diag}(\mathbf{c}_1, \mathbf{c}_2, c_3)$ is called precision matrix with constant parameters $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^3$, $c_3 \in \mathbb{R}$, and the constant R is the weight control coefficient. The addition of $\ddot{s}_t$ to the non-linear term $\mathbf{f}$ guarantees that the computation of the Jacobian $\mathbf{J}_t = \partial \mathbf{f}_t / \partial \mathbf{s}_t$ is not rank deficient. Let the robot EE position be at $\bar{\mathbf{y}}_t$. We apply the Newton's method to minimize the cost $c(\mathbf{s}_t, u_t) = \mathbf{f}(\mathbf{s}_t)^\top \mathbf{Q} \mathbf{f}(\mathbf{s}_t) + u_t^\top R u_t$. This can be done by carrying out a second order Taylor expansion around the point $(\mathbf{s}_t, u_t)$, that is (Calinon, 2023b):

$$\begin{aligned} \Delta c(\Delta \mathbf{s}_t, \Delta u_t) \approx & 2\Delta \mathbf{s}_t^\top \mathbf{J}(\mathbf{s}_t)^\top \mathbf{Q} \mathbf{f}(\mathbf{s}_t) + 2\Delta u_t^\top R u_t + \\ & + \Delta \mathbf{s}_t^\top \mathbf{J}(\mathbf{s}_t)^\top \mathbf{Q} \mathbf{J}(\mathbf{s}_t) \Delta \mathbf{s}_t + \Delta u_t^\top R \Delta u_t. \end{aligned} \quad (6.11)$$

Therefore, the optimization problem in (6.9) is linearized using the cost function in (6.11), i.e.

$$\min_{\Delta \mathbf{u}} \Delta c(\Delta \mathbf{s}, \Delta \mathbf{u}) \quad \text{s.t.} \quad \Delta \mathbf{s} = \mathbf{S}_s \Delta \mathbf{u}, \quad (6.12)$$

with $\Delta \mathbf{s} = [\Delta \mathbf{s}_1, \dots, \Delta \mathbf{s}_T]^\top$, $\Delta \mathbf{u} = [\Delta u_1, \dots, \Delta u_{T-1}]^\top$, and $\mathbf{S}_s$ and $\mathbf{S}_u$ characterizing the state and control transfer matrices of the system's dynamics in (6.10) expressed at trajectory level for $t = 1, 2, \dots, T$, namely $\mathbf{s} = \mathbf{S}_s \mathbf{s}_1 + \mathbf{S}_u \mathbf{u}$ (Calinon, 2023b). The computation of $\mathbf{S}_s$ and $\mathbf{S}_u$ is here omitted to improve readability, but can be found in Appendix D.1. By differentiating (6.12) with respect to $\Delta \mathbf{u}$ and equating to zero, one can compute at each iteration the optimal control command:

$$\Delta \mathbf{u}^* = (\mathbf{S}_u^\top \mathbf{J}(\mathbf{s})^\top \mathbf{Q} \mathbf{J}(\mathbf{s}) \mathbf{S}_u + \mathbf{R})^{-1} (-\mathbf{S}_u^\top \mathbf{J}(\mathbf{s})^\top \mathbf{Q} \mathbf{f}(\mathbf{s}) - \mathbf{R} \mathbf{u}). \quad (6.13)$$

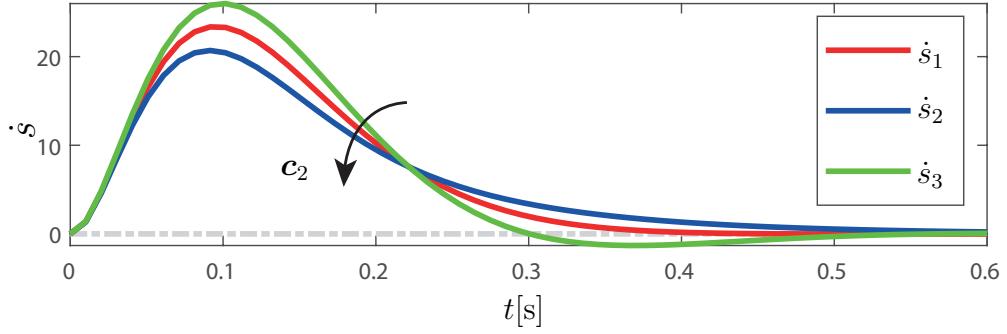


Figure 6.6: Behaviour of the curvilinear parameter velocity $\dot{s}$ with varying velocity weight c_2 .

With the proposed LQT formulation, the quadratic cost function in (6.9) offers intuitive and interpretable settings for parameters $\mathbf{Q}$ and R . Additionally, a second-order approximation reliably guides the optimization steps towards the minimum of the local estimation of the cost function (6.11) (Calinon, 2023b; Luenberger, Ye, et al., 1984). While higher-order methods may provide a more accurate approximation, they require additional algorithms to find their minimum, thereby increasing computational effort. The LQT algorithm described earlier was initially simulated in a reaching task. The velocity profiles $\dot{s}$ are displayed in Fig. 6.6. The figure demonstrates that the proposed framework achieves the characteristic bell-shaped profile which characterizes human reaching tasks (Todorov & Jordan, 1998). Moreover, by reducing c_2 in the precision matrix $\mathbf{Q}$, the velocity error $\dot{\mathbf{e}} = \dot{\mathbf{y}}_t - \dot{\boldsymbol{\mu}}(s_t)$ becomes less damped, thus the controller more reactive. As a consequence, the velocity profile changes its sign before reaching zero, as for s_3 depicted in green. This behaviour is common in rapid arm movements, where a corrective movement is actuated to refine the reaching precision (Meirovitch et al., 2016).

The implementation of the LQT algorithm is resumed in Algorithm 3. In its real-time implementation, the LQT algorithm is computed at each iteration with reduced time steps $t = 1, 2, \dots, T_W$, using a Model Predictive Control approach. This allows for faster computation of the optimal control $\hat{u}_{t+1}$ by minimizing the time window T_W . Additionally, the control $\Delta \mathbf{u}^*$ is iteratively computed until its norm reaches a lower bound Δ_{min} , with a maximum number of epochs I_{MAX} set to prevent unwanted latency. The initial conditions for each iteration are set to the last commanded pair (s_{t-1}, u_{t-1}) , providing a warm start for faster convergence.

Algorithm 3 Iterative LQT algorithm**Input:** $\mathbf{y}_{\bar{t}}$, $\boldsymbol{\mu}(s_{\bar{t}})$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$, R , in (6.9)- (6.10), I_{MAX} .**Output:** $\mathbf{s}_{\bar{t}+1}^*$.

Initialization : for $t = \bar{t}, \bar{t}+1, \dots, \bar{t}+T_W-1$ compute the transfer matrices $\mathbf{S}_s$, $\mathbf{S}_u$ and initialize $\mathbf{s}_{1,0} = \mathbf{s}_{\bar{t}}$, $\mathbf{u} = [0_{\bar{t}}, \dots, 0_{\bar{t}+T_W-1}]$.

- 1: **for** $i = 1$ to I_{MAX} **do**
- 2: Compute the dynamics $\mathbf{s} = \mathbf{S}_s \mathbf{s}_{1,i-1} + \mathbf{S}_u \mathbf{u}$
- 3: Calculate the residual $\mathbf{f}(\mathbf{y}_{\bar{t}}, \mathbf{s})$ and the Jacobian $\mathbf{J}(\mathbf{s})$ in (6.9)- (6.12)
- 4: Compute $\Delta \mathbf{u}^*$ as in (6.13)
- 5: **if** ($\|\Delta \mathbf{u}^*\| < \Delta_{min}$) **then**
- 6: Local minimum reached, exit **for** loop
- 7: **end if**
- 8: Update control $\mathbf{u} = \mathbf{u} + \Delta \mathbf{u}^*$
- 9: Update initial state $\mathbf{s}_{1,i} = \mathbf{A} \mathbf{s}_{1,i-1} + \mathbf{B} \mathbf{u}_1$
- 10: Re-define the control as $\mathbf{u} = [u_2, u_3, \dots, u_{T_W}, u_{T_W}]^\top$
- 11: **end for**
- 12: Save the state $\mathbf{s}_{\bar{t}+1}^* = \mathbf{s}_{1,i} = [s_{1,i}, \dot{s}_{1,i}, \ddot{s}_{1,i}]^\top$
- 13: **return** $\mathbf{s}_{\bar{t}+1}^*$

6.4 Experiments and Results

We compare the proposed linear quadratic tracking (LQT) algorithm, together with the Gauss-Newton (GN), a virtual mechanism (VM) approach (Raiola, Lamy, & Stulp, 2015) and a simple control in gravity compensation (GC). The validation consists in two different experimental scenarios which are detailed in the following sections.

6.4.1 Experimental Setup

Experiments involved a Franka Emika Panda robot together with a Schunk FT-AXIA force/torque sensor. The robot controller has been implemented in C++, while the codes for the optimization of s_t were developed in MATLAB; finally, each node was connected using Robot Operative System (ROS). The tested algorithms imposes that the generated reference point $\boldsymbol{\mu}(s_t)$ follows the EE displacement caused by the user (Sciavicco & Siciliano, 2012). To minimize the kinesthetic effort while preventing the user from sensing the natural manipulator's inertia, the robot has been endowed with an admittance control. Specifically, the translation dynamics in the Cartesian space have been chosen as:

$$\mathbf{M} \ddot{\hat{\boldsymbol{\mu}}}_t + \mathbf{D} \dot{\hat{\boldsymbol{\mu}}}_t + \mathbf{K} (\hat{\boldsymbol{\mu}}_t - \boldsymbol{\mu}(s_t)) = \mathbf{F}_t, \quad (6.14)$$

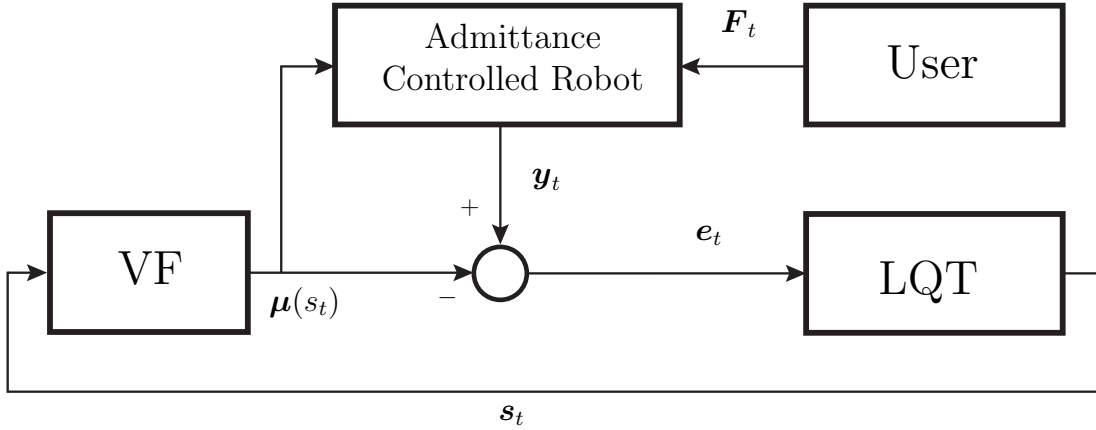


Figure 6.7: Controller framework. Here VF denotes the virtual fixture constraint.

where $\mathbf{M} = \text{diag}(m, m, m)$, $\mathbf{D} = \text{diag}(d, d, d)$, $\mathbf{K} = \text{diag}(k, k, k)$ are the simulated inertia, damping and stiffness coefficient, respectively, while $\mathbf{F}_t$ represents the measured force at the EE. To achieve good tracking and low manipulator inertia, we empirically choose $m = 1.5[\text{kg}]$, $d = 15[\text{N} \cdot \text{s}/\text{m}]$ and $k = 200[\text{N} \cdot \text{m}]$. The position $\hat{\boldsymbol{\mu}}_t$ computed from the admittance model (6.14) is sent to the robot and used as a reference signal for inverse dynamics position control in Cartesian space (Sciavicco & Siciliano, 2012). During the experiments, the orientation of the EE was kept fixed. The adopted framework is schematized in Fig. 6.7.

Finally we consider the values in (6.9) for control weight R and the precision matrix $\mathbf{Q} = \text{diag}(\mathbf{c}_1, \mathbf{c}_2, c_3)$, given $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^3$ and $c_3 \in \mathbb{R}$. The initial values were selected based on the specific path $\boldsymbol{\mu}$ needed for the task. Specifically, we employed an inverse LQR approach as in (Klemm et al., 2019; Priess et al., 2014) for computing first attempt values. The procedure is detailed in Appendix D.2.

6.4.2 Center-reaching task

In Section 6.2 we made some theoretical considerations about the behaviour of the GN algorithm close to EDS. To validate so, an experiment consisting in moving the EE towards the center of the osculating circle $\boldsymbol{\gamma}(\hat{s})$ has been proposed. The performed task refers to Fig. 6.8.

During the experiments three users were involved, two of them with no prior expertise in robotics. Each execution was recorded for approximately one minute. The numerical values chosen for the precision matrix $\mathbf{Q}$ were $\mathbf{c}_1 = 47.8 \cdot \mathbf{1}_3$, $\mathbf{c}_2 = 0.02 \cdot \mathbf{1}_3$, $c_3 = 0.01$ and the control weight was set to $R = 1e-5$. The constraint curve $\boldsymbol{\mu}$ was obtained by kinesthetic demonstration over the curve of

Fig. 6.8(a).

In Fig. 6.8 the red and blue colors refer to the GN and LQT algorithms, respectively. Figure 6.8(b) plots the recorded EE position $\mathbf{x}_{GN}$ and $\mathbf{x}_{LQT}$ extrapolated from an interval of 6s; Fig. 6.8(c) plots the variations of the force modulus $|\mathbf{F}|$ and its argument $\angle \mathbf{F}$; finally, Fig. 6.9(a)-(b) summarise the statistics of all the task executions in terms of the rate of change of the force modulus and argument, denoted with $d|\mathbf{F}|/dt$ and $d\angle \mathbf{F}/dt$.

The experiments underlined that the proximity to the center of the osculating circle $\gamma(\hat{s})$ induces abrupt directional changes when using the GN algorithm. Precisely, Fig. 6.8(b) demonstrates that reaching the center of $\gamma(\hat{s})$ is not problematic in the LQT case, while it becomes complicated for the GN case. A major explanation of this phenomena is given in Fig. 6.8(c). Despite the modulus of the force $|\mathbf{F}|$ being in the same range for both GN and LQT, the former displays sharp variations for the argument $\angle \mathbf{F}$, induced by the proximity with an EDS. This behaviour was observed throughout all the center-reaching task as revealed in the boxplots of Fig. 6.9(a)-(b). On the left, one can observe that the rate of change of $|\mathbf{F}|$ is similar for both algorithms while, on the right, the GN case exhibits larger variations of the rate of change of $\angle \mathbf{F}$.

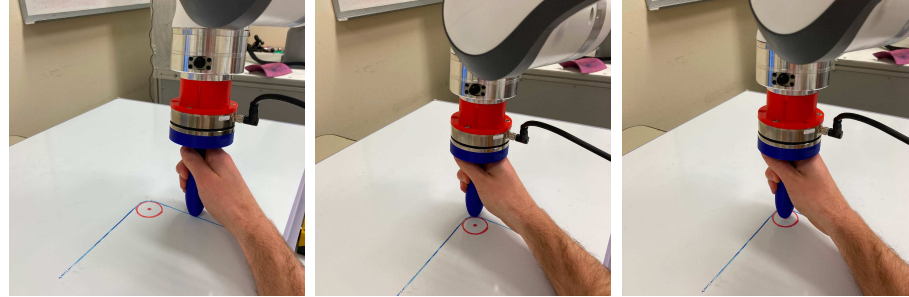
6.4.3 Target following task

In this experiment, the objective is to ensure that the EE's tip remains close to the moving point displayed on a second PC, as shown in Fig. 6.10(a). The aim is to prove that the LQT algorithm proposed in 3 is applicable to a real-time human-robot interaction scenario, and compare it with the GN, VM and GC performances. The VM methodology for phase calculation was introduced in (Raiola, Lamy, & Stulp, 2015). The authors consider a virtual mechanism (VM) connected to the end effector of the robot and characterized by a spring-damper system, namely:

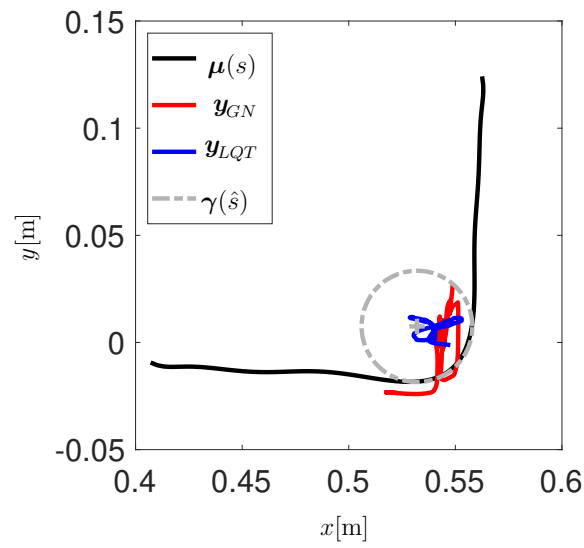
$$\mathbf{F}_t = \mathbf{K}(\mathbf{y}_t - \boldsymbol{\mu}(s_t)) + \mathbf{D}(\dot{\mathbf{y}}_t - \dot{\boldsymbol{\mu}}(s_t)), \quad (6.15)$$

where $\mathbf{K}$ and $\mathbf{D}$ are chosen to be the same as in (6.14). The force exerted in the VM is always orthogonal to its velocity, that is $\boldsymbol{\mu}'(s_t)^\top \mathbf{F}_t = 0$. By plugging (6.15) into this condition, one obtains:

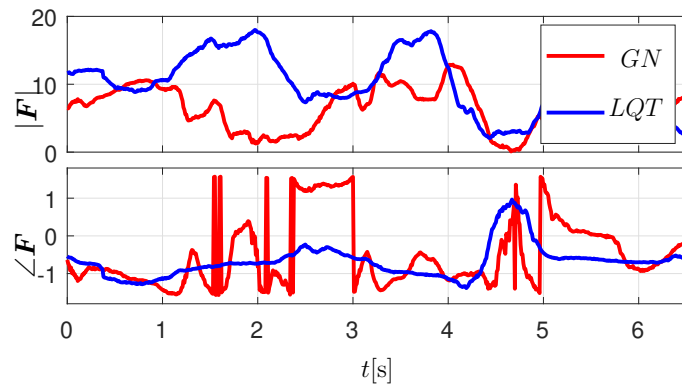
$$\boldsymbol{\mu}'(s_t)^\top (\mathbf{K}(\mathbf{y}_t - \boldsymbol{\mu}(s_t)) + \mathbf{D}(\dot{\mathbf{y}}_t - \dot{\boldsymbol{\mu}}(s_t)\dot{s}_t)) = 0, \quad (6.16)$$



(a)



(b)



(c)

Figure 6.8: Experimental results of the center-reaching task.

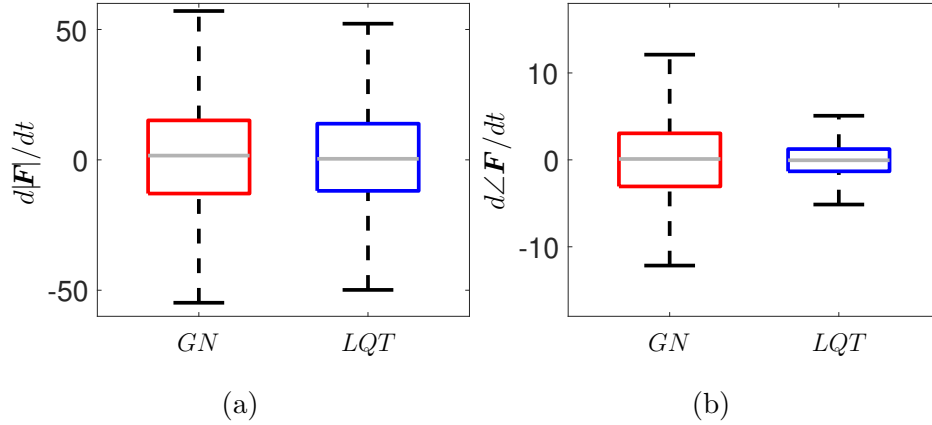


Figure 6.9: Box plots for the center-reaching task.

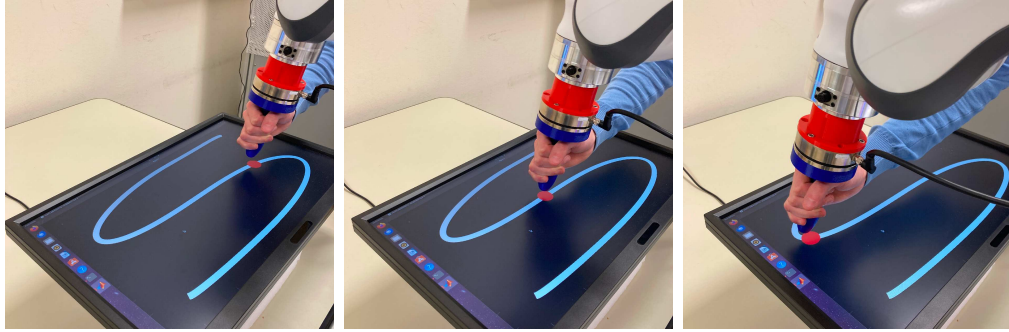
from which the following update law holds

$$\dot{\mathbf{s}}_t = (\boldsymbol{\mu}'(s_t)^\top \mathbf{D} \boldsymbol{\mu}'(s_t))^{-1} \boldsymbol{\mu}'(s_t)^\top (\mathbf{K}(\mathbf{y}_t - \boldsymbol{\mu}(s_t)) + \mathbf{D} \dot{\mathbf{y}}_t). \quad (6.17)$$

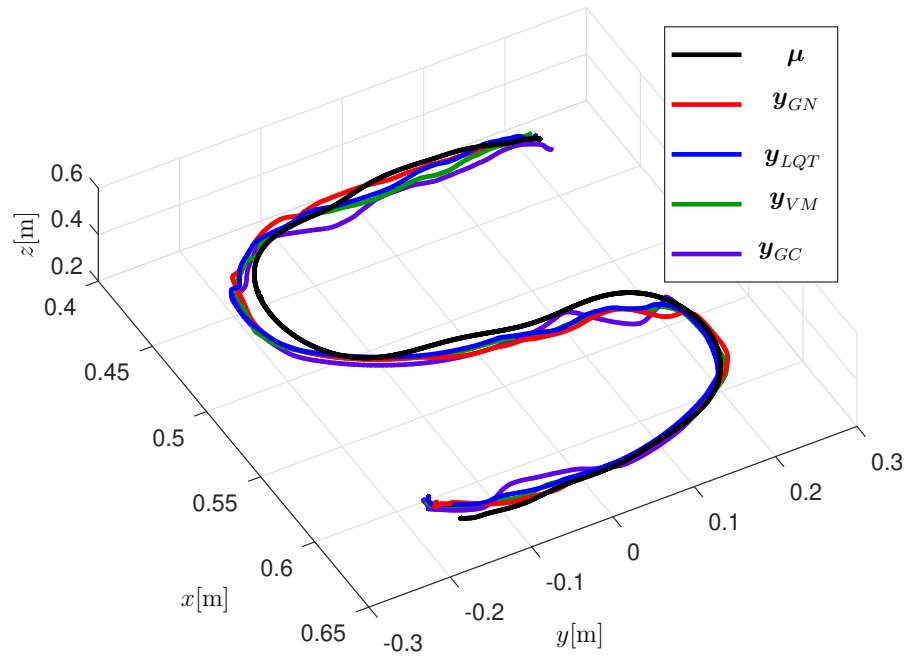
During the experiments ten users were involved, four of them with no prior expertise in robotics. Every user had to execute the target following task eight times, alternating the right and left hand for each of the three tested algorithms. Each task execution lasted 50 seconds. The numerical values chosen for the precision matrix $\mathbf{Q}$ were $\mathbf{c}_1 = 400.0 \cdot \mathbf{1}_3$, $\mathbf{c}_2 = 0.14 \cdot \mathbf{1}_3$, with $\mathbf{1}_3 = [1, 1, 1]^\top$, $\mathbf{c}_3 = 0.01$ and the control weight was set to $R = 1e-5$. The constraint curve $\boldsymbol{\mu}$ was obtained by kinesthetic demonstration over the second PC monitor.

The results are summarized in Fig. 6.10- 6.11. Figure 6.10(b) plots the recorded EE positions $\mathbf{y} = [x_t, y_t, z_t]^\top$ for the GN (red), LQT (blue), VM (green) and GC (violet) cases, while the constraint curve $\boldsymbol{\mu} = \boldsymbol{\mu}(s)$ is depicted in black. Given the target position $\boldsymbol{\mu}_t = [\mu_{x,t}, \mu_{y,t}, \mu_{z,t}]^\top$, it is straightforward to see that in the four cases the robot succeeded to follow $\boldsymbol{\mu}_t$ with a limited error $\mathbf{e}$. Numerical values for the term $\|\mathbf{e}\| = \|\mathbf{y} - \boldsymbol{\mu}(s)\|$ are resumed in Table 6.1 with their mean and standard deviation.

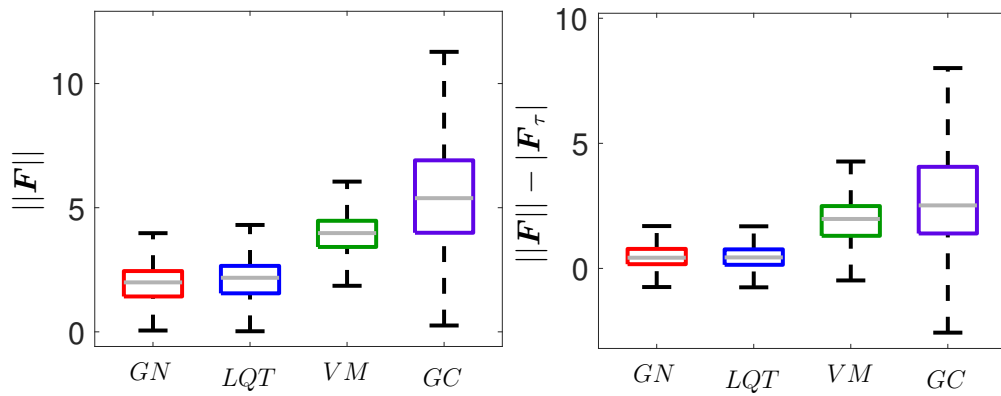
Interesting results come when analyzing the measurements of the forces applied to the end effector $\mathbf{F} = [F_x, F_y, F_z]^\top$. It is reasonable to assume that the virtual constraint $\boldsymbol{\mu}$ simplifies the target following task, as it guides the robot EE along with point movements. For the GC case, the user is not constrained to the curve $\boldsymbol{\mu}$, making it more difficult to perform the task. The outcome, on average, is a significant larger effort required for the GC case, described in Fig. 6.10(c) with the norm of the acquired force measurements. If we define $F_{\tau,t} = \boldsymbol{\mu}'_t \cdot \mathbf{F}_t$ as the force component $\mathbf{F}_t$ projected to the tangential direction of the curve



(a)



(b)



(c)

(d)

Figure 6.10: Experimental results of the target following task.

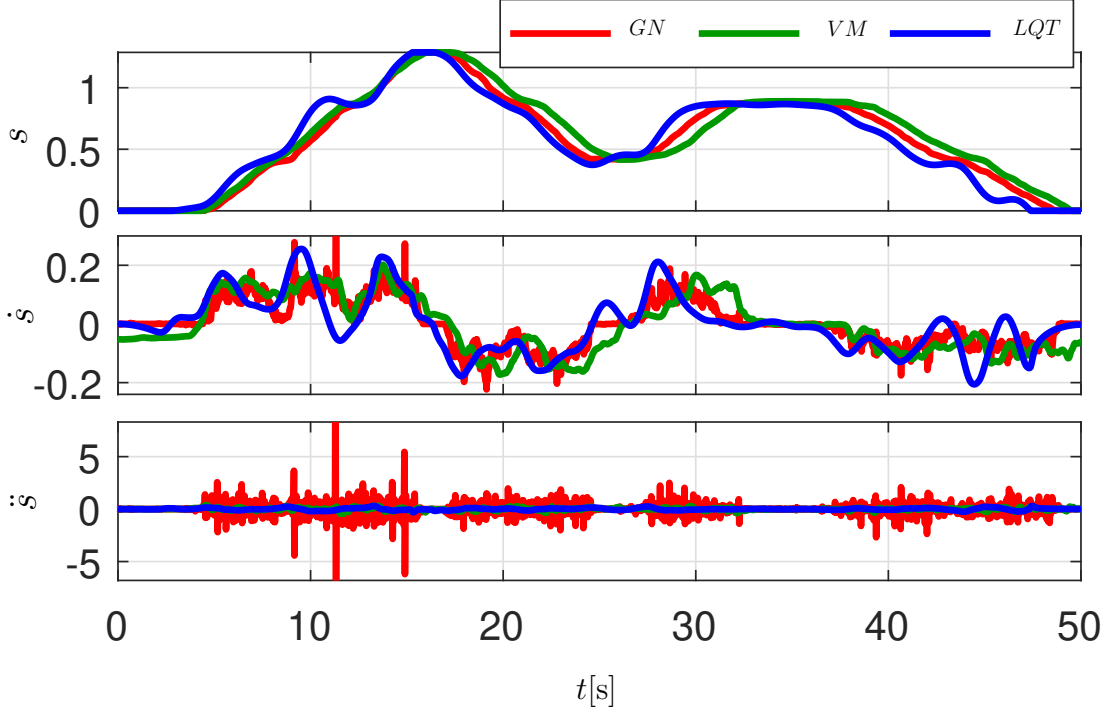


Figure 6.11: Phase and its derivatives profile for the target following task.

$\mu'_t = \partial \mu(s_t) / \partial s_t$, the difference $\|\mathbf{F}_t\| - |\mathbf{F}_{\tau,t}|$ reported in Fig. 6.10(d) quantifies the amount of force which is spent in holding the EE close to the curve μ . Again, as the GN, LQT and VM algorithms impose a virtual constraint, they exhibit substantial lower effort requirements if compared to the GC case, which makes the GC less suitable for this kind of application.

While GN, LQT and VM performances appear comparable, differences emerged when comparing the generated trajectories. Indeed, the GN algorithm computes a velocity command with no smoothness cost on $\dot{s}$ and $\ddot{s}$, which affect the reference trajectories $\mu(s_t)$, $\dot{\mu}(s_t)$ and $\ddot{\mu}(s_t)$. Conversely, the VM algorithm in (6.17) showed slight vibration, especially in regions with high curvature. This may be attributed to the choice of $\mathbf{D}$ and $\mathbf{K}$ parameters in (6.14), which is critical for human-in-the-loop applications (Braglia et al., 2025; Ferraguti et al., 2019). This is exemplified by the experimental measurements reported in Fig. 6.11. Here we can appreciate that the LQT algorithm, by minimizing a jerk command, computes smoother derivatives $\dot{s}$ and $\ddot{s}$, demonstrating robustness against the difficulties encountered in the GN and VM cases. The smoothness has been quantitatively evaluated with the dimensionless squared jerk (DSJ) (H. Chen et al., 2024):

$$DSJ(s) = \tau \sum_{t=t_0}^T \ddot{s}_t^2, \quad DSJ(\ddot{\mathbf{y}}) = \tau \sum_{t=t_0}^T \|\ddot{\mathbf{y}}_t\|^2, \quad (6.18)$$

Table 6.1: Errors between EE and target position and DSJ.

	$\ e\ [\text{cm}]$	DSJ(s)	DSJ($\ddot{\mathbf{y}}$)
GN	2.5 ± 1.16	$(1.02 \pm 3.446)e+14$	$(5.44 \pm 5.570)e+16$
LQT	2.3 ± 1.31	$(3.61 \pm 5.771)e+9$	$(7.40 \pm 0.071)e+15$
VM	2.3 ± 1.26	$(1.22 \pm 3.446)e+11$	$(1.53 \pm 1.129)e+16$
GC	2.2 ± 1.14	$\times$	$(1.18 \pm 1.347)e+17$

with $\tau = (T - t_0)^5/L^2$. Parameters $T = 50s$ and $L = 3.5m$ represent the task duration and the path length, respectively. The metric in 6.18 was applied both to the phase variable s computed by the algorithms (absent for the GC case), and the actual end effector position $\mathbf{y}$. To reduce noise from computing derivatives, data for $DSJ(\ddot{\mathbf{y}})$ were pre-processed using a moving average filter with a window size of $w = 20$. The results in Table 6.1 show an improvement in smoothness with the proposed LQT algorithm.

6.4.4 Discussion

Results from the target following task confirm the previous statement, with higher force demand when no virtual constraint is implemented. Indeed, this increases cognitive stress for the user, who must track the target point while ensuring the EE follows the specified curve. Results also highlights that incorporating the proposed LQT framework has the advantage, over a GN and VM implementations, of generating smoother trajectories as evinced by the results in Fig. 6.11 and Table 6.1. By specifying the precision matrix $\mathbf{Q}$ and the control weight R , one can robustly contain unwanted vibrations by minimizing a jerk command and guaranteed a smooth execution of the proposed guiding tasks (Sharkawy, 2021; Todorov & Jordan, 1998).

Nevertheless, a high cost on the jerk command can reduce the algorithm tracking precision, as it penalizes high acceleration profiles (Meirovitch et al., 2016). In the target following task, this effect could alter the perceived inertia of the robot when moving along the task curve. Although lower accelerations may impact the robot's responsiveness (Palleschi et al., 2019), we demonstrate in the center-reaching task the importance of imposing a smoothness cost. Precisely, when reaching EDSs, the solution of the optimization cost in (6.2) degenerates (Luenberger, Ye, et al., 1984). Using the GN algorithm, experiments underlined the presence of sharp changes in the force direction exerted to the EE. This is caused by the generation of abrupt velocity commands in (6.4), as opposed to

the proposed algorithm which, instead, successfully prevents this condition.

Representing tasks on different domains

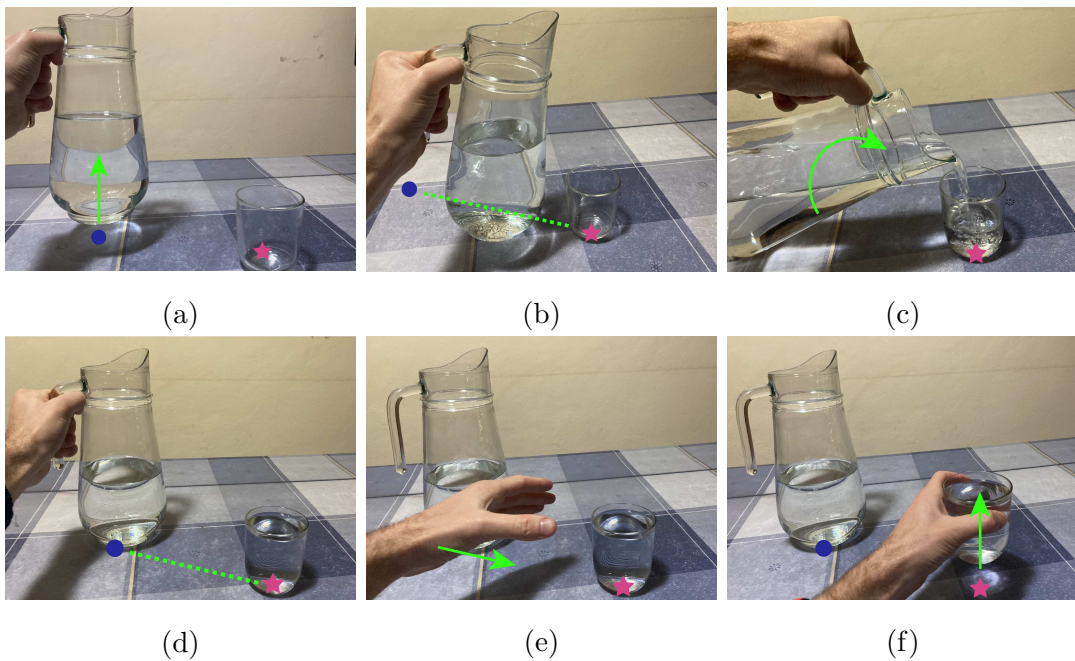


Figure 7.1: A simple example of movement primitives in water pouring: the task could be resumed by three fundamental movements combined together, i.e., grab an object, move it from one point to another, pouring.

When we learn a certain task, we typically follow the instruction of someone with a major expertise on that field. Learning involves mainly observation as a first step, repetition to acquire proficiency in performing a certain task, then generalization to extend the acquired knowledge to other situations. In all these three steps our brain relies on a synthesized representation of such task in order to recognize in which context what we learn could be reproduced, and how to actuate it.

In robotics, this approach is commonly used in the context of motion rep-

resentation, relying on the concept of *movement primitives* (MPs) (Ijspeert et al., 2002; Pastor et al., 2009). A MP could be imagined as a building brick, a fundamental movement unit that, composed with other MPs, could represent complex and more sophisticated movements. An example of this concept is given in Fig. 7.1. This sequence of actions represent a trivial task of pouring water in a glass. From Fig. 7.1(a) to 7.1(f), the subject *pick* the water jug (a), *move* it toward the glass (b), *pour* some water inside (c), *move* the water jug back to its initial point (d), *move* the hand towards the glass (e), *pick* the glass (f). In this example we can recognize at least three fundamental motion commands, or MPs, like *pick*, *move* and *pour* that, combined together, enables to perform the pouring task. One key aspect of these MPs, is that they represent a basis for a broader set of task, meaning that they can be re-used in different tasks, or in variations of the pouring task itself.

Indeed, motion commands differs in the way they are represented and transferred to the robot. The pouring task in Fig. 7.1, for example, is represented as a sequence of high level actions; nevertheless, the same task can be represented as pose trajectories of the hand in the snapshots. In robotics, depending on how the task is represented, we can distinguish between *symbolic* and *trajectory* domains of representation (Calinon, 2009).

At symbolic level, the task is represented through sequential or hierarchical information to establish a discrete set of movements with predefined rules, like the figure above. An example of techniques used for this type of representation are Hidden Markov Models (HMM) (Jurafsky & Martin, n.d.). HMM are based on a Markov chain, which describes the probabilities of sequences of random variables, called *states*, that take values from a defined set. In many applications, such as gesture and speech recognition (Pentland & Liu, 1999; Roveda et al., 2021), HMM are used to model human behavior, with the aim of identifying gestures or predicting the most likely pattern of subsequent control states. At trajectory level, instead, the task is represented at a lower level of abstraction by means of non-linear functions that describes continuous signals that change over time. Commanding pose or joint trajectories, possibly with constraints on their derivatives, is an example of this kind of representation (Biagiotti & Melchiorri, 2008; Bianco & Ghilardelli, 2017).

Both domains of representation present advantages and limitations. On one hand, symbolic representation provides a more intuitive and natural way of programming tasks but lacks direct control over the robot states, sometimes leading to non-exact or unclear execution of the commands. On the other hand, at the

trajectory level, it is easier to have control over the robot state variables and ensure robustness of the controller, but this comes at the cost of losing interpretability and increasing the level of expertise required by those programming the robot.

While discussing the integration of symbolic and trajectory-based representations, it is important to clarify the distinction between symbolic and continuous variables, as well as the potential overlap between symbolic and discrete variables. In this chapter, symbolic variables refer to abstract representations of motion primitives (MPs) that convey human-interpretable meanings, such as *pick*, *move*, and *pour* (Calinon, 2009; Chaquet et al., 2013). These symbols provide a structured way to describe tasks at a higher level of abstraction. In contrast, discrete variables are numerical or categorical states that belong to a predefined finite set, without necessarily carrying human-like semantics. For instance, in Hidden Markov Models (HMMs), discrete states can emerge from self-organized clustering methods that segment motion into probabilistic states, which may not always align with intuitive human task segmentation (Calinon et al., 2007; Rabiner, 1989). Additionally, while continuous variables in trajectory-based representations are often associated with open-loop control, e.g. defining pose, velocity, or acceleration profiles, it is also possible to interpret them in the context of feedback control laws (Schaal, 1999; Sciavicco & Siciliano, 2012). This distinction will be made explicit throughout the chapter, ensuring a clearer understanding of how both symbolic and trajectory-based representations contribute to motion programming for the target application.

This chapter aims to introduce some early attempts at merging the two domains of representation by proposing a novel framework that combines the modification of the geometric path at trajectory level with a speed adaptation mechanism at symbolical one (Braglia et al., 2023). The goal here is to provide evidence, through an obstacle avoidance application, that task performance can benefit from the merging of these two domains. In particular, the objective is:

1. Design an online controller that can fit generic tasks and smoothly avoid collisions with dynamic obstacles.
2. Include the possibility to restore the original task whenever the robot is not prone to any collision.
3. Exploit a probabilistic framework to gather information about the obstacle and modify the robot's velocity accordingly.

4. Combine trajectory and symbolic domains in a unified framework.

7.1 Task Encoding

The simplest way to define a robotic task is by specifying its trajectory, i.e. a mapping between time and space that the end effector or joints must follow. Alternatively, a task can be represented as a sequence of action units or elements governed by loops or rules, as exemplified in (Hovland et al., 1996; Roveda et al., 2021).

This work aims to integrate these two domains for an obstacle avoidance application. The approach employs a control system that adjusts the nominal robot path, defined by B-spline functions (Biagiotti & Melchiorri, 2008), while dynamically altering the task phase and velocity using a Hidden Markov Model (HMM) (Pentland & Liu, 1999). The following paragraphs summarize the theoretical foundations of these methods.

7.1.1 Spatial Encoding based on B-Splines

Spline functions are widely used for generating smooth and optimal time trajectories in robotic applications (Biagiotti & Melchiorri, 2013; Jankowski et al., 2023; Sayols et al., 2020). B-splines, a specific type of spline curve, are defined as a linear combination of N basis functions (Biagiotti & Melchiorri, 2013), expressed as:

$$\mathbf{p}(s) = \sum_{i=1}^N \mathbf{p}_i \beta_i^d(s), \quad 0 \leq s \leq L, \quad (7.1)$$

with L being the length of the curve $\mathbf{p}$, with $\beta_i^d(s)$ being the basis functions of degree d , depending on the phase variable s , and $\mathbf{p}_i$ being the control points that determine the geometric shape of the curve. As illustrated in Figure 7.2, a B-spline is essentially a smooth approximation of the control polygon defined by the control points. These control points are typically determined by imposing interpolation conditions $\mathbf{p}(s_i) = \mathbf{p}_i$, $i = 1, \dots, N$, based on the desired task. A noteworthy property of the basis functions $\beta_i^d(s)$ is that their value is zero everywhere except in the interval defined by knots $[s_i, s_{i+d+1}]$. As a consequence, a change in the i -th control point $\mathbf{p}_i$ only influences the B-spline in this interval, allowing for local modifications in the trajectory described. Another useful property of B-splines is time-scaling, i.e., the possibility of changing the velocity along the curve by Moreover, by simply applying a transformation of the type $\hat{s}_t = \alpha s_t$,

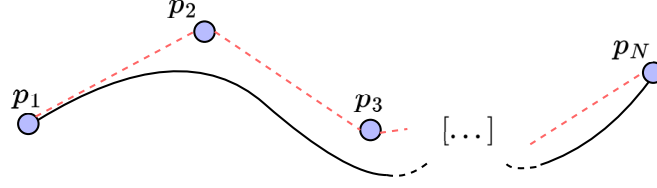


Figure 7.2: Example of B-spline trajectory: $\mathbf{p}(s)$ (black line) is an approximation of the control polygon (red dashed line) defined by the control points $\mathbf{p}_i$.

one can easily vary the velocity $\dot{\mathbf{p}}(s_t)$ according to the variation of $\alpha \in \mathbb{R}$.

7.1.2 Temporal and Sequential Encoding Based on Hidden Markov Models

Hidden Markov Models (HMMs) are based on a Markov chain, describing the probabilities of sequences of random variables, referred to as *states*, that take values from a predefined set (Martin, 2009). HMMs are widely used in applications like gesture and speech recognition (Pentland & Liu, 1999; Roveda et al., 2021) to model human behavior by identifying gestures or predicting the most likely sequence of subsequent control states.

Specifically, given a set of hidden states $H = \{h_1, h_2, \dots, h_M\}$ and a sequence of T observations $O = \{o_1, o_2, \dots, o_T\}$, HMMs analyze event sequences through a reduced parameter set $\boldsymbol{\lambda} = [\boldsymbol{\Pi}, \mathbf{A}, \mathbf{B}]$, where:

- $\boldsymbol{\Pi}$ is the *prior distribution*, indicating the probability of starting a sequence in state h_i ;
- $\mathbf{A}$ is the *transition probability matrix*, where each element $a_{i,j}$ represents the probability of transitioning from state i to state j ;
- $\mathbf{B}$ contains the *observation likelihoods* (emission probabilities), where each element $b_i(o_t)$ expresses the probability of an observation o_t being generated from state i .

In the standard HMM definition, $\boldsymbol{\lambda}$ describes a model with discrete observation likelihoods (Mongillo & Deneve, 2008). However, for the proposed framework, which deals with continuous observation spaces, such as trajectories in Cartesian space, the HMM is represented as:

$$\boldsymbol{\lambda} = [\boldsymbol{\Pi}, \mathbf{A}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i] \quad i = 1, \dots, K \quad (7.2)$$

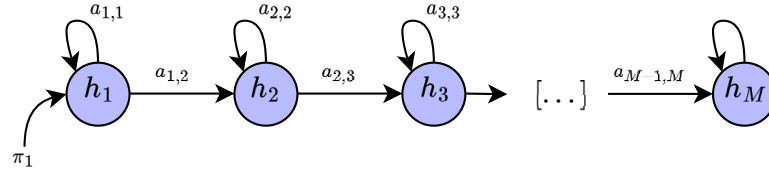


Figure 7.3: Example of forward-chained left-right HMM: being in state h_i , it is only possible to move to the next state h_{i+1} or remain in the current state.

where K is the number of components characterized by the mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ of a multivariate Gaussian Mixture Model (GMM). Using this representation, the $\boldsymbol{\lambda}$ parameters can be trained to encode the desired robot task, enabling the regression of the model to reproduce the desired end-effector path (Vasquez et al., 2009).

The proposed methodology can be explained as follows: using control points $\mathbf{p}_i$ as hidden states h_i , a forward-chained left-right HMM is adopted (see Fig.7.3) to account for the evolution of the trajectory (Walter et al., 1999). With this model, in nominal conditions (i.e. no obstacles in the workspace), $\pi_1 = 1$ and transition probabilities $a_{i,i+1} \approx 1$, meaning that the robot evolves from one control point to the other with a probability close to one. In particular, let us analyze the transition matrix $\mathbf{A}$ for the HMM in Figure 7.3:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & 0 & 0 \\ 0 & a_{22} & a_{23} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{K-1,K-1} & a_{K-1,K} \\ 0 & 0 & 0 & \cdots & 0 & a_{KK} \end{bmatrix} \quad (7.3)$$

Here, the elements of each row of $\mathbf{A}$ sum up to 1. Parameter $a_{i,i}$ is the probability of staying in state h_i , while $a_{i,i+1}$ is the probability of moving from state h_i to state h_{i+1} . Thus, the transition probabilities $a_{i,i+1}$ can be used to slow down or eventually stop the robot every time their values warn the system about a potential collision, as is explained in the next section.

7.2 Proposed Method

The proposed approach, which integrates B-splines and Hidden Markov Models (HMMs) for real-time motion adaptation in human-robot collaboration, differs significantly from traditional scheduling mechanisms such as Behavior Trees (BTs) and AND/OR graphs. BTs, commonly used in robotics and artificial

intelligence, provide a hierarchical and modular framework for task execution, where nodes represent actions and conditions that dictate robot behavior in a structured manner (Colledanchise & Ogren, 2018). Similarly, AND/OR graphs, widely employed in automated planning and decision-making, encode task decomposition and dependencies, allowing for systematic exploration of possible execution paths (Dechter & Mateescu, 2007). While these methods are well-suited for high-level task scheduling and decision-making, they typically lack inherent mechanisms for continuous real-time adaptation to dynamic environments. In contrast, the proposed framework directly modifies the geometric trajectory using B-splines while adjusting motion velocity probabilistically through HMMs, enabling smooth and reactive obstacle avoidance without predefined task transitions. Unlike BTs or AND/OR graphs, which require explicit rule definitions, the HMM-based approach learns motion patterns statistically, providing a probabilistic representation of task execution that adapts dynamically based on environmental conditions. This distinction highlights the advantage of integrating trajectory and symbolic control within a unified framework, enhancing the robot's ability to operate safely and efficiently in shared workspaces.

We consider a typical industrial scenario in which a robot is required to perform a specific task within a monitored workspace, using a fixed camera. When an obstacle enters the workspace, such as the hand of a human operator, its actual position is tracked, and sent to the controller that implements the proposed framework in order to prevent possible collisions. The functional blocks of the entire algorithm are displayed in the scheme shown in Figure 7.4, while the working principles are detailed below.

7.2.1 Spatial Modulation with Dynamical Control Points

Given the presence of moving obstacles, the task is modified only when the obstacles interfere with the nominal trajectory. If the obstacles move away from the robot, the task reverts to its original trajectory. To implement this, B-spline functions are used to induce a reversible deformation of the trajectory. Specifically, a dynamical system is associated with each control point $\mathbf{p}_i$ (Chiaravalli et al., 2018):

$$\mathbf{M}_i \ddot{\tilde{\mathbf{p}}}_i + \mathbf{D}_i \dot{\tilde{\mathbf{p}}}_i + \mathbf{K}_i \tilde{\mathbf{p}}_i = \mathbf{F}_{rep,i}, \quad (7.4)$$

where $\tilde{\mathbf{p}}_i = \bar{\mathbf{p}}_i - \mathbf{p}_i$ represents the displacement of the actual position $\bar{\mathbf{p}}_i$ of the i -th control point from its original location $\mathbf{p}_i$. The matrices $\mathbf{M}_i = \text{diag}\{m_i\}$, $\mathbf{D}_i = \text{diag}\{d_i\}$, and $\mathbf{K}_i = \text{diag}\{k_i\}$ are 3-by-3 diagonal matrices, where m_i , d_i ,

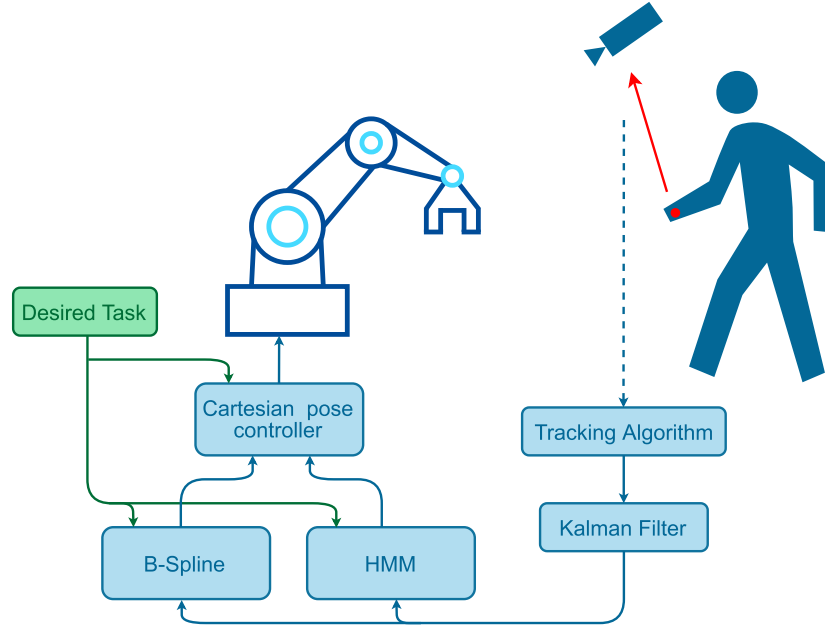


Figure 7.4: Block scheme representation of the proposed framework for collision avoidance.

and k_i are positive constants ensuring the system in (7.4) is critically damped. This can be achieved by setting $m_i = \hat{m}_i$ and $k_i = \hat{k}_i$, then computing d_i as $\hat{b}_i = 2\sqrt{\hat{m}_i \cdot \hat{k}_i}$. Finally, $\mathbf{F}_{rep,i}$ is a virtual repulsive force directed from the obstacle to the i -th control point.

Figure 7.5 illustrates this concept. The magnitude of the repulsive force $\mathbf{F}_{rep,i}$ acting on the i -th control point does not directly depend on its Cartesian distance from the obstacle. Instead, it varies based on the likelihood of the human hand colliding with the obstacle. To compute this, the Mahalanobis distance is used:

$$D_M(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}, \quad (7.5)$$

where $\mathbf{x}$ is a generic state (e.g., hand position $\in \mathbb{R}^3$), and $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are the mean and covariance of a Gaussian distribution. In this application, (7.5) quantifies the obstacle's occupation probability, modeled as $\hat{\mathbf{p}}_{obs} \sim \mathcal{N}(\boldsymbol{\mu}_{obs}, \boldsymbol{\Sigma}_{obs})$, incorporating uncertainties in obstacle position estimation (Kanazawa et al., 2019; Meuter et al., 2008).

The magnitude of the repulsive force is then computed as (Kanazawa et al., 2019):

$$|F_{rep,i}| = \frac{\chi_i}{D_M(\hat{\mathbf{p}}_i; \boldsymbol{\mu}_{obs}, \boldsymbol{\Sigma}_{obs})}, \quad (7.6)$$

where χ_i is a tunable parameter to adjust the intensity of the repulsive force field.

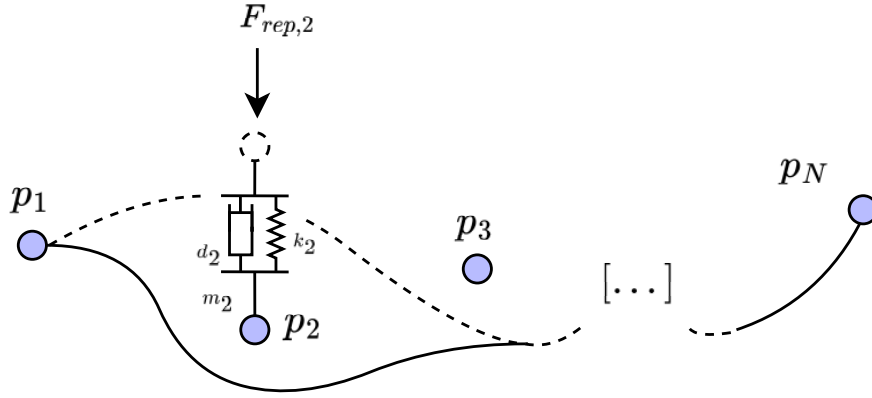


Figure 7.5: B-spline with associated dynamical systems to control points p_i : when an external force $F_{rep,2}$ acts on p_2 , it locally changes the desired path.

Finally, the path is encoded using a third-order B-spline, ensuring continuity of derivatives up to acceleration, with limited jerk. This careful design is critical as, in practical scenarios, a poorly designed trajectory could cause joint acceleration discontinuities, leading to errors in the robot's internal inverse kinematics algorithm during position control (Biagiotti & Melchiorri, 2013).

7.2.2 Temporal Modulation with Varying Transition Probabilities in HMM

In our framework, the task path is represented by $\mathbf{p}(s)$ in (7.1), while the velocity is modulated through the HMM parameters $\boldsymbol{\lambda}$, enabling temporal adjustments. The connection between the B-spline representation and the HMM domain is established via the continuous observation likelihoods $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = b_i(o_t)$, $i = 1, \dots, M$, as described in (7.2). Specifically, $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ are components of a GMM that encode the likelihood of a point $\mathbf{r} \in \mathbb{R}^3$ being described by the HMM model $\boldsymbol{\lambda}$:

$$\mathbf{B} \sim P(\mathbf{r}) = \sum_{i=1}^M \mathcal{W}_i(\mathbf{r}) \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (7.7)$$

where $\mathcal{W}_i$ is the mixing coefficient satisfying $\sum_{i=1}^M \mathcal{W}_i = 1$, representing the influence of the i -th component (Calinon et al., 2007). Gaussian distributions are encoded such that their regression corresponds to the nominal trajectory $\mathbf{p}(s)$ (Wiest et al., 2012). In (7.7), $\mathbf{B}$ captures the variability introduced by the movement of control points, thereby providing a stochastic and topological representation of the task. Figure 7.6 depicts a graphical representation of this approach.

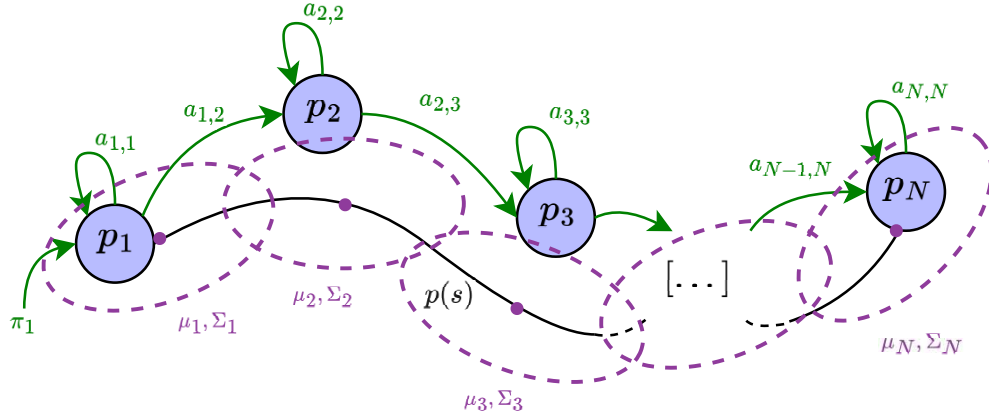


Figure 7.6: B-spline and HMM combined together: while the former describes the nominal path $\mathbf{p}(s)$ in the Cartesian domain, the latter provides a probabilistic map of it as described in (7.7), together with transition probabilities $a_{i,j}$ among control points $\mathbf{p}_i$ computed such that a left-right HMM is obtained, thus ensuring always going from a start position to a goal one.

It is important to note that the number of HMM states M generally differs from the number of control points N that define the B-spline. For instance, a single HMM state can encode a trajectory segment corresponding to a subtask involving multiple control points. However, for simplicity, it is convenient to assume $M = N$.

As explained in Section 7.1.1, the evolution of the B-spline trajectory can be modified by applying the transformation $\hat{s} = \alpha s$ to the knot vector of $\mathbf{p}(s)$. This scales the desired velocity $\dot{s}$ by a factor $\alpha \in [0, 1]$, reducing it accordingly. To implement a mechanism that slows down the robot and stops it in case of collision risk, the transition probabilities of the HMM are mapped into appropriate values of α , as described below.

Assume the robot's trajectory is in state h_i (associated with the control point $\mathbf{p}_i$), and consider the transition probability $a_{i,i+1}$ to the next state h_{i+1} . The following conditions are imposed:

- If $a_{i,i+1} = 1.0$, the robot moves at nominal velocity.
- If $0.0 < a_{i,i+1} < 1.0$, the robot slows down.
- If $a_{i,i+1} = 0.0$, the robot stops.

The transition coefficient is computed as a Sigmoid function ϕ based on the Mahalanobis distance $D_M(\hat{\mathbf{p}}_{obs}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ between the obstacle position and the HMM observation space:

$$a_{i,i+1} = \phi(D_M) = \frac{1}{1 + e^{-(D_M + \gamma)}}, \quad (7.8)$$

where γ is a free parameter that adjusts the robot's sensitivity to obstacles. Positive γ values shift the Sigmoid function $\phi(D_M)$ to the right, allowing fine-tuning of the distance D_M at which $a_{i,i+1}$ decays to zero.

If the scaling factor α is set as

$$\alpha = a_{i,i+1}, \quad (7.9)$$

the nominal velocity along the B-spline trajectory is modulated according to the specified conditions.

This approach reinterprets $a_{i,i+1}$ as the probability of transitioning from control point $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$ without collision. The symbolic control is structured as follows: the path is encoded as a B-spline, which combines basis functions weighted by control points. Each basis function locally encodes a segment of the trajectory in conjunction with its corresponding control point. Hence, the control points serve as the hidden states of the HMM.

Access to information about a hidden state relies on the parameter $\mathbf{B}$, as detailed in Section 7.1.2 and initialized as a GMM in (7.7). In this framework, $\mathbf{B}$ provides a probability mapping that identifies the control point $\mathbf{p}_i$ responsible for the trajectory segment closest to the obstacle position $\hat{\mathbf{p}}_{obs}$. With this information, the transition probability between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ can be adjusted using (7.8), and the nominal velocity is modulated via α as defined in (7.9). Consequently, as $\alpha \rightarrow 0$, the robot halts, and when the obstacle moves out of the task space, $\alpha \rightarrow 1$ restores the nominal velocity. Figure 7.7 illustrates an example of 2D encoding, integrating the techniques described in the preceding sections.

It is important to note that the phase-law scaling $\hat{s} = \alpha s$ could, in principle, be applied to any encoding technique, including the GDMP proposed in Chapter 5 and the LQT formulation of Chapter 6. While their combination has not yet been explored, the B-Spline framework was chosen due to its dynamic control point configuration, as described in (7.4), which offers a more intuitive and straightforward implementation.

This, however, does not imply that the same approach would not be feasible with alternative solutions. For instance, in Chapter 4.5, we demonstrated how an optimal control formulation of DMP can accommodate dynamic via-points. This could be further extended by integrating a Hidden Markov Model (HMM) framework, thereby creating a spatio-temporal representation capable of handling both trajectory and symbolic domains. Indeed, temporal modulation handled by HMM could possibly be included into the DMP framework through the scaling factor

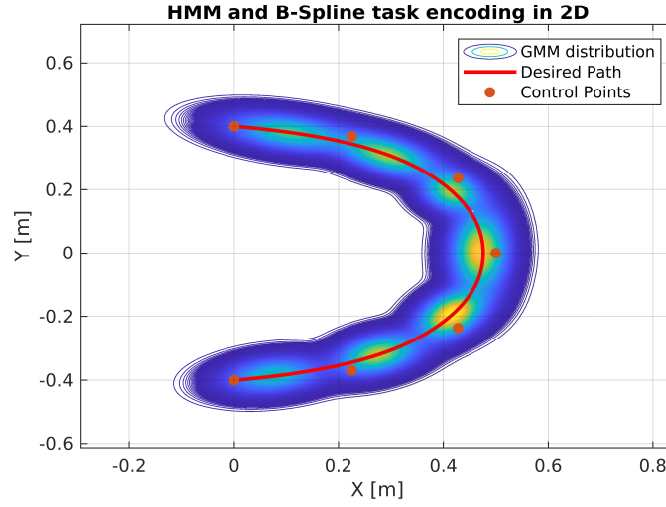


Figure 7.7: Control at both trajectory and symbolical level using B-splines and HMM ($\xi = 0.5$). A planar task, also used in the experiments, is considered for illustration purposes. It is encoded using a B-spline exploiting 7 control points (with fixed height $z = 0.3$ m), and an HMM with observation space described by a GMM with 7 components.

τ in (4.27). However, this would loose the association between control points and hidden states that was assumed here, requiring a novel connection between symbolic and trajectory domains to be established. The primary objective of this chapter is to provide theoretical insights and empirical evidence demonstrating how high- and low-level representations of a task can be effectively integrated within a unified framework.

7.2.3 Switching between Trajectory and Symbolic Domains

The proposed motion planner combines two different control domains. However, there could be situations where operating at the trajectory level is preferred over the symbolic one, or vice versa. The algorithm is provided with a knob parameter, $\xi \in [0, 1]$, so the user can choose the influence of each controller. Specifically, we expect the following behaviors:

- $\xi = 0.0 \implies$ only the time-scaling mechanism based on HMM is active;
- $0.0 < \xi < 1.0 \implies$ B-spline and HMM cooperate with different proportions;
- $\xi = 1.0 \implies$ only the B-spline modification algorithm is applied.

To achieve this, the parameters (m_i, d_i, k_i) in (7.4) for trajectory control, and α in (7.9) for symbolic control, are modified according to the following laws:

$$(m_i, d_i, k_i) = \begin{cases} \frac{(\hat{m}_i, \hat{d}_i, \hat{k}_i)}{2\xi} & \text{if } 0 \leq \xi < 0.5 \\ (\hat{m}_i, \hat{d}_i, \hat{k}_i) & \text{if } 0.5 \leq \xi \leq 1 \end{cases} \quad (7.10a)$$

$$\alpha = \begin{cases} \hat{\alpha} & \text{if } 0 \leq \xi \leq 0.5 \\ (2\hat{\alpha} - 1)(1 - \xi) + \xi & \text{if } 0.5 < \xi \leq 1 \end{cases} \quad (7.10b)$$

where $(\hat{m}_i, \hat{d}_i, \hat{k}_i)$ are the default values assigned in (7.4), and $\hat{\alpha}$ is the result of (7.9). In this way, as ξ approaches 0, the values of the parameters (m_i, d_i, k_i) increase rapidly (in practical experiments an upper bound on the three parameters is imposed), making the system in (7.4) extremely rigid and thus insensitive to external (virtual) forces. As a consequence, no spatial modifications of the trajectory are observed. On the other hand, as ξ approaches 1, the spatial modification mechanism is restored, but in this case, $\alpha \rightarrow 1$ and therefore no changes in the velocity profile occur. Finally, when $\xi \approx 0.5$, both the trajectory and symbolic level controllers work together, producing spatial and temporal modulation of the trajectory.

7.3 Experiments and Results

For the experimental tests, we used a Franka Emika Panda collaborative robot controlled through a Cartesian pose controller developed in C++. The controller ran on a standard PC equipped with an Intel i7 8-core CPU and 8 GB RAM. The trajectory/symbolic motion control was implemented in Matlab/Simulink on the same PC. The C++ controller updated the robot states at a frequency of 1 kHz. To handle sharp trajectory deviations and potential discontinuities during inverse kinematics computation (Gaz et al., 2019), we used a second-order filter to calculate the acceleration profile. The filtered acceleration profile was then double-integrated to update the end effector (EE) position. For the target object, a human's hand represented the moving obstacle, sharing the same workspace with the robot. The hand's Cartesian position was continuously tracked by a fixed camera, specifically an Intel RealSense D435. The camera's output was processed on a second PC with similar characteristics to the first one. All software components were connected using ROS to facilitate communication among

them (Quigley et al., 2015b). In particular, to simplify the computation of the hand's pose from the camera data, we used an ArUco marker directly placed on the operator's hand (Garrido-Jurado et al., 2014). The position coordinates derived from the camera images were sent through a multistep Kalman filter to predict the future positions of the hand (Kohler et al., 1997; Meuter et al., 2008). This step proved valuable when the camera's view of the scene was obstructed. The Kalman filter output, denoted as $\hat{\mathbf{x}}$, was associated with a uniformly distributed random variable (Julier & Uhlmann, 1997), i.e., $\hat{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})$, which justified the use of the Mahalanobis distance in modifying the task (as shown in (7.6)) (Kanazawa et al., 2019).

Given the focus on collaborative tasks where manipulator speed is constrained, the encoded trajectory was designed using position and velocity data rather than explicitly accounting for manipulator dynamics. Motor torques and accelerations were not incorporated into the proposed planning and control framework. Instead, parameters such as the dynamics of the spline control points and the GMM covariance matrix were chosen to ensure feasible collision avoidance, considering the typical motion velocity of a human (Hovland et al., 1996; Kanazawa et al., 2019; Merckaert et al., 2022). For testing, a cubic B-spline with $N = 7$ control points in the xyz plane was employed at the trajectory level. We selected (m_i, d_i, k_i) in (7.4) to ensure high stiffness at the first and last control points to maintain their positions constantly. Specifically, we empirically chose the parameters (m_i, d_i, k_i) to be

- $m_i = 20 \text{ [kg]}, d_i = 310 \left[\frac{\text{Ns}}{\text{m}} \right], k_i = 1200 \left[\frac{\text{N}}{\text{m}} \right]$ for $i \in \{1; N\}$;
- $m_i = 5 \text{ [kg]}, d_i = 60 \left[\frac{\text{Ns}}{\text{m}} \right], k_i = 180 \left[\frac{\text{N}}{\text{m}} \right]$ for $i \in 2, 3, \dots, N - 1$;

At the symbolic level, the task was encoded into a Gaussian Mixture Model (GMM) with 7 components (i.e., $M = N$) in 3D space. These components served as observation likelihood parameters in (7.2) to construct a left-right hidden Markov model (HMM). We adjusted (7.8) with $\gamma = 10$ to increase the sensitivity of the symbolic controller even for higher Mahalanobis distance values computed in (7.5).

During the experiments, the robot followed a desired trajectory while an operator moved his hand close to it to simulate collisions. We varied the parameter ξ in (7.10) to individually analyze the B-spline control, the HMM, and the combination of both controllers. This allowed us to demonstrate the improvements achieved through the integration of the two techniques and validate our novel

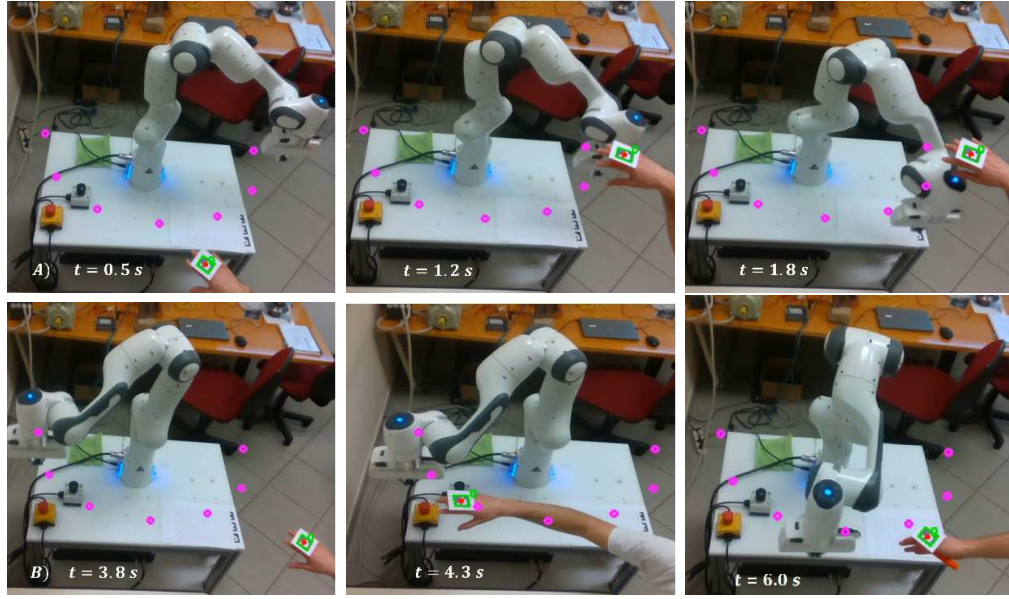


Figure 7.8: Snapshots from experiments conducted on the B-spline–HMM-based controller. The control points corresponding to the nominal path are displayed in magenta. In sequence (A), the hand holding the marker moves slightly above the end effector, inducing the robot to slow down and modify its path to pass under the hand. In sequence (B), the hand suddenly moves in front of the end effector, giving the robot no time to modify its path. Therefore, the robot controller decides to stop and wait until the hand changes position.

planning methodology for obstacle avoidance. We evaluated the performance using four parameters:

1. The average computation time for a single iteration T_{comp} .
2. The normalized average path deviation ΔL , calculated as the mean deviation of the traveled distance normalized over the nominal path.
3. The average stop time, T_{stop} , calculated as the average time taken for parameter α in (7.9) to reach zero (i.e., stop the robot) once the hand collision was perceived.
4. The success rate, ρ , defined as the percentage of cases where the distance did not go below a given safety threshold (equal to 10 cm in our experiments) compared to the total number of situations where an incipient collision was detected.

Figure 7.8 displays snapshots of the real experiments conducted on the robot to test the B-spline–HMM-based controller.

In Fig.7.9 and Table 7.1, we report the results of the experiments that are discussed further in the following paragraphs, while in Fig.7.10, we present a

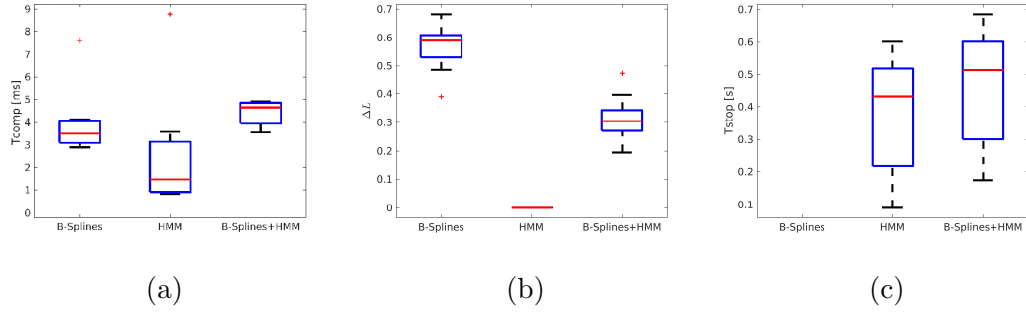


Figure 7.9: Box plots of the values reported in Table 7.1; in plots (a)-(c) the values of T_{comp} , ΔL , and T_{stop} are displayed, respectively.

Table 7.1: Validation tests results in terms of average computation time for single iteration T_{comp} , normalized average path deviation ΔL , average stop time T_{stop} , and success rate ρ .

	T_{comp} (ms)	ΔL	T_{stop} (s)	ρ
B-spline	3.515 ± 1.53	0.590 ± 0.08	/	77.97%
HMM	1.480 ± 2.70	0.000 ± 0.00	0.431 ± 0.01	89.66%
B-spline+HMM	4.649 ± 0.55	0.304 ± 0.07	0.514 ± 0.01	94.41%

comparison of the trajectory of the hand recorded and virtually reproduced in three different tests using B-spline, HMM, and B-spline–HMM-based controllers. The XY projections on the right of Fig.7.10 show path modifications, while temporal modulation along the time axis is observed. To assess the performance of the controllers under different scenarios, several experiments were conducted where the robot executed the same task repeatedly. These experiments involved varying parameters such as ξ (balance between trajectory and symbolic control), task execution speed, and acceleration.

Controller at Trajectory Level Only ($\xi = 1.0$)

In this test, the controller operated solely at the trajectory level, with $\xi = 1.0$. The average computation time for a single iteration was $T_{comp} = 3.515$ ms. Across six different experiments, the robot encountered a total of 59 possible collision situations. The success rate (ρ) achieved was 77.97%, and the normalized average path deviation (ΔL) was 0.590. Since the controller only focused on spatial control without temporal modulation, no value of T_{stop} was reported as the robot never stops. In Fig. 7.10(a) one can observe that the presence of an obstacle results in path modifications but no change in task duration, thus the nominal and modified trajectories were aligned in time.

Controller at Symbolic Level Only ($\xi = 0.0$)

In the second test, the controller operated exclusively at the symbolic level, with $\xi = 0.0$. It employed an HMM with an observation space described by a GMM with seven components. The average computation time for a single iteration was $T_{comp} = 1.480$ ms. Across five different experiments, the robot faced 58 possible collision situations. The success rate (ρ) achieved was 89.66%, with an average stop time (T_{stop}) of 0.431 s and $\Delta L = 0.0$ since no spatial controller was implemented. In Fig. 7.10(b) one can observe that the end effector trajectory slowed down when close to the obstacle, but no path modification was observed as the XY projections of the nominal and modified trajectories perfectly overlapped.

Controller at Both Trajectory and Symbolic Levels ($\xi = 0.5$)

In the final validation test, the task was encoded using a control at both the trajectory and symbolic levels, with $\xi = 0.5$. Similar configuration parameters as the previous tests were maintained. The average computation time for a single iteration was $T_{comp} = 4.649$ ms, which was slightly higher than the previous cases but still met real-time requirements within 16 ms (H. Liu et al., 2022; Merckaert et al., 2022). Across 10 different experiments, the robot encountered 143 possible collision situations. The success rate (ρ) achieved was 94.41%, with a normalized average path deviation (ΔL) of 0.304 m and an average stop time (T_{stop}) of 0.514 s. In Fig. 7.10(c) the two controllers lead the task of the robot to be modified both in terms of speed and geometry of the executed trajectory.

Performance Analysis

In the first scenario with only B-spline modification as the control ($\xi = 1.0$), the performance was relatively poor ($\rho = 77.97\%$). The controller lacked velocity control, leading to difficulties in rapidly modifying the robot's path to avoid collisions. Additionally, tuning (m_i, d_i, k_i) parameters could tighten or relax ΔL , but the controller's overall performance was unsatisfactory due to the absence of temporal modulation. Similarly, in the controller employing only HMM control ($\xi = 0.0$), performance was improved compared to B-spline control ($\rho = 89.66\%$). However, this controller's behavior was inadequate when the operator's hand moved towards the robot. By integrating the B-spline and HMM controllers, the combined B-spline–HMM controller achieved a higher success rate ($\rho = 94.41\%$). Although the average stop time (T_{stop}) was higher than the HMM controller alone, real-time requirements were still satisfied (H. Liu et al., 2022; Merckaert

et al., 2022) and the B-spline-HMM controller consistently slowed down and modified the robot’s path until the safety distance was maintained before stopping completely. The normalized average path deviation (ΔL) was lower compared to the trajectory level only controller, as the robot now reduced its velocity towards the hand and stopped before any collisions. This feature was critical in avoiding potential singular configurations that could occur with the B-spline controller alone.

7.4 Discussion of limitations and challenges

The proposed framework introduces a real-time planning algorithm designed to enhance the motion planning of robots using B-splines and Hidden Markov Models (HMMs). This approach allows for both spatial and temporal modulation of a robot’s trajectory, offering the ability to modify path geometry and adjust velocity in response to dynamic environments. While the framework demonstrates promising results in its specific context, it is essential to critically examine its limitations to guide future improvements and contextualize its applicability within broader robotic systems.

One of the primary concerns lies in the framework’s capacity to ensure human safety in shared workspaces. By focusing exclusively on modifying the EE trajectory in Cartesian space, the approach leaves the configuration of the robot’s arm largely unaddressed. This limitation is particularly significant for redundant robotic systems, where multiple configurations can achieve the same EE position. For instance, scenarios in which a human approaches the robot from an angle interacting with the robot’s body, rather than the EE’s path, are not adequately addressed by the proposed method. This could lead to potential collisions, which highlights the need for an approach that considers the robot’s entire geometry and proximity to obstacles. A more holistic approach would involve planning at the joint level, ensuring that both the arm’s configuration and EE trajectory contribute to overall safety.

Another important aspect to consider is the applicability of the proposed framework to industrial tasks. While B-splines provide an efficient way to generate smooth paths, their utility may be limited in scenarios requiring strict adherence to predefined trajectories, such as welding or assembly operations. These tasks often demand a level of precision and flexibility that may not be achievable through B-spline representations alone. Additionally, the method’s reliance on adjusting control points in response to obstacles could introduce unintended

behaviors. For example, excessive displacement of control points might cause the trajectory to flip direction or exhibit discontinuities, raising concerns about its reliability in high-stakes industrial environments.

The mathematical formulation of the framework also warrants closer examination, particularly regarding its stability. The system employs nonlinear excitation forces to adjust the control points which, while effective in certain conditions, may introduce instability under others. Sudden changes in the obstacle positions or system parameters could result in abrupt trajectory adjustments, compromising both smoothness and predictability. Ensuring stability in such a nonlinear system would require a more rigorous analysis, potentially incorporating constraints or safeguards to mitigate the risks of discontinuities or erratic behavior.

The novelty and comparative performance of the framework are also areas where further clarification and validation would strengthen its contribution, for example by offering a detailed comparison with established methods such as potential fields (Khansari-Zadeh & Khatib, 2017), Rapidly Exploring Random Trees (RRTs) (Rodriguez et al., 2006), or Covariant Hamiltonian Optimization for Motion Planning (CHOMP) (Zucker et al., 2013). Without such benchmarks, it is difficult to grasp the advantages of the proposed method relative to widely used alternatives. Finally, it is essential to consider broader implications for robotic systems, particularly in terms of joint limit violations and self-collision, which are not explicitly addressed by the framework. These factors are critical in redundant robotic systems and must be integrated into any robust collision avoidance solution. In summary, while the proposed framework presents an innovative integration of B-splines and HMMs for motion planning, its limitations point to opportunities for further refinement.

7.5 Conclusions

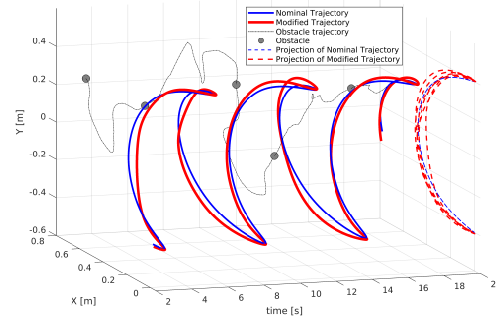
In recent years, robotics has increasingly embraced artificial intelligence (AI) techniques across a wide range of systems and applications. Learning algorithms excel in processing external data and interpreting it to support decision-making processes. This capability is particularly valuable as robotics moves toward a future in which robots are expected to collaborate closely with humans. AI's strength lies in its ability to handle high-level commands and data acquisition tasks, making it indispensable in contexts that require adaptability and responsiveness. However, low-level control, including trajectory-level approaches like those discussed in this chapter, continues to play a vital role due to its robustness

and reliability in scenarios where deploying AI may be unnecessary or inefficient.

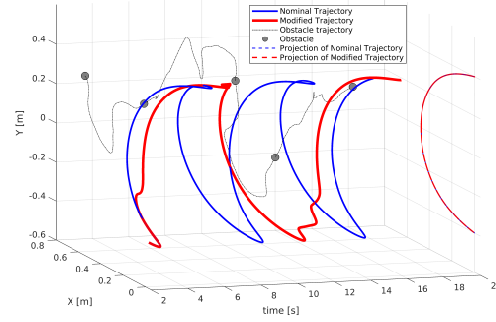
For many robotics applications, knowledge of the robot model remains a cornerstone of effective control strategies. This foundational understanding should not be replaced but rather augmented by AI, especially in situations involving unpredictable events. By integrating traditional control methods with AI-driven techniques, researchers and practitioners can create hybrid frameworks that leverage the strengths of both approaches. Such a synthesis is increasingly viewed as a promising direction for bridging the gap between theoretical advancements and real-world applications.

In this context, the framework presented in this chapter represents an initial step toward this integration. By combining trajectory-level control with elements of adaptability, it contributes to the ongoing effort to advance human-robot collaboration. Further refinement and exploration of this hybrid approach could pave the way for more versatile and effective robotic systems capable of operating safely and efficiently in dynamic environments.

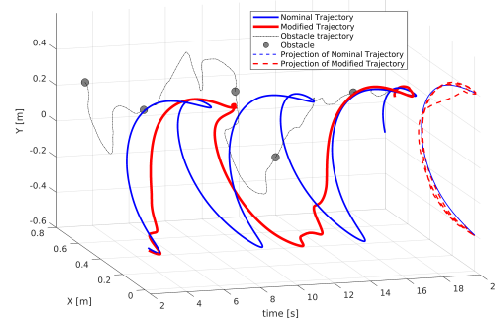
In summary, while the proposed framework presents an innovative integration of B-splines and HMMs for motion planning, its limitations point to opportunities for further refinement. Addressing these challenges through enhanced safety mechanisms, more versatile trajectory representations, and rigorous performance benchmarking can significantly enhance its applicability and impact.



(a)



(b)



(c)

Figure 7.10: Three experiments using the same obstacle trajectory (black dashed line). (a) control based on B-spline ($\xi = 1.0$); (b) control based on HMM ($\xi = 0.0$); (c) control based on both B-spline and HMM ($\xi = 0.5$). The nominal path of the robot is shown in blue and the actual one in red. The XY coordinates of the end effector were observed over time with respect to the obstacle position. The overall trajectory was projected onto the XY plane on the right-hand side for better visualization of geometric path modifications.

Conclusion

This chapter concludes the thesis with some further remarks on the obtained results. Additionally, a brief discussion over human-robot interaction is provided, together with some final words about open issues in this field that can be of interest for possible future research directions.

8.1 Discussion and limitations

Chapter 2 provides an overview of the framework serving as the foundation for the applications introduced in this work. The framework begins with a teaching phase, where demonstrations illustrate the task the robot must replicate. This requires consistent data acquisition and processing to extract the necessary information for task reproduction. The robot is then integrated into an architecture that involves human interaction, either in contact with the robot or indirectly, depending on the application.

This type of architecture is common in other applications (Ficuciello et al., 2014; Keemink et al., 2018; H. Liu et al., 2022), but its extension for more general-purpose robots demands further advancements. Experiments, such as those in Chapters 5- 7, have shown that reliable applications can be built on top of this architecture. This requires, firstly, a robust controller to manage the robot's movements and respond to human feedback, and secondly, the establishment of interaction metrics for evaluation and refinement.

The main limitation of this architecture is its restricted learning capabilities. The teaching phase is an initial step, but no subsequent learning occurs. Ideally, in LfD, a few demonstrations should suffice to generalize the task. To enhance adaptability and control policy refinement, online learning during execution should complement offline training. This would enable the system to adapt dynamically to varying requirements and scenarios.

The learning step is elaborated in Chapters 3 and 4. Chapter 4 introduces

state-of-the-art learning and encoding techniques such as GMM (Calinon et al., 2007), DMP (Ijspeert et al., 2002), and LQR (Y. Zhang et al., 2024), while Chapter 3 focuses on extracting consistent skills from the informative content of demonstrations. For example, in guiding applications, the novel SS algorithm (detailed in Chapter 3.2) is used to extract the geometric content of demonstrations, enabling (i) spatial alignment of sequences and (ii) flexibility in defining the robot’s speed profile.

The SS algorithm was incorporated into all proposed applications following the demonstration collection phase. Its significance in the framework is evident, particularly in contexts like rehabilitation, where therapists must accommodate patient limitations during demonstrations. These limitations, including pauses or speed variations, could impact task encoding and execution (Braglia, Onfiani, et al., 2024). Despite its utility, the SS algorithm presents two main limitations that suggest future research directions.

The first limitation relies on its applicability. While the SS algorithm addresses specific challenges in guidance tasks, its potential utility in broader robotic and non-robotic contexts remains unclear. For example, its alignment capability could be valuable in other domains, but not all applications benefit from filtering temporal information. As highlighted in Chapter 3.1, the DTW algorithm serves as a benchmark for evaluating the SS algorithm due to DTW’s well-established versatility. In fields like speech processing, where temporal information is critical, is clearly more convenient to use DTW rather than SS.

The second limitation concerns the algorithm’s definition. Until now, the SS algorithm has been applied to workspace trajectories in Euclidean space, where the parameter Δ is defined as a constant distance between consecutive samples. However, defining Δ in non-Euclidean manifolds, such as orientation spaces, remains unexplored. For instance, when handling end-effector poses, the orientation manifold requires consideration of rotational parameters, and arc-length parametrization may no longer apply. While this presents a challenge, it also offers an opportunity to extend the SS algorithm to other contexts, opening doors for further research.

In Chapters 4 and 5, the focus shifts to Dynamic Movement Primitives (DMP). As discussed earlier, a limitation of the adopted framework is the absence of mechanisms for online learning. However, generalization does not always necessitate new learning procedures; instead, it often involves adapting prior knowledge. This adaptability is a key reason why DMP is well-known in LfD. DMP offers several adaptive features, such as speed scaling, variations in start/end positions, and

the inclusion of coupling terms (Saveriano, Abu-Dakka, Kramberger, & Peternel, 2023). Furthermore, combining multiple DMPs allows for more complex task representations (Gaspar et al., 2018; Ruckert & d’Avella, 2013).

The extension of DMP presented in Chapter 5, termed GDMP, marks a significant advancement in applying DMP to Human-Robot Interaction (HRI) frameworks. By leveraging the SS algorithm, GDMP introduces a phase-independent version of DMP, effectively decoupling geometric and temporal information. This decoupling enables adaptation to various phase profiles. In Chapter 5.2, GDMP demonstrated its capability for offline task duration optimization, outperforming previous DMP variants. However, its adaptability to online scenarios remains unverified. For instance, changes to the initial or final target positions might render the precomputed optimal phase profile inadequate for respecting the robot’s kinematic constraints.

Similarly, the human-in-the-loop application discussed in Chapter 5.3 requires further exploration. Introducing a varying orientation of the end-effector could be addressed using orientation DMP (Koutras & Doulgeri, 2020a; Saveriano et al., 2019). However, this introduces the challenge of synchronizing orientation and position. In co-manipulation tasks, these two DMPs may evolve either synchronously or independently, depending on task requirements. Further research is needed to refine these synchronization mechanisms, particularly in human-robot co-manipulation scenarios.

These considerations can also be extended to the LQT algorithm introduced in Chapter 6. This chapter presents an approach using the LQT algorithm to regulate the robot’s end-effector position along a virtual constraint, enabling the generation of smooth trajectories while avoiding abrupt changes in the derivatives of the robot’s reference position. The LQT algorithm employs a jerk control command, providing the robot with a motion model that resembles human movement mechanisms.

However, as previously noted, orientation control remains unaddressed. It would be worthwhile to explore whether incorporating a jerk command for orientation could enhance execution, particularly by improving the co-manipulation experience. Additionally, the analysis of the LQT framework lacks subject-related evaluations, which could provide deeper insights into optimizing the model’s parameters for better performance.

Finally, Chapter 7 discusses a slightly different architecture where the robot is not in contact with the human but instead avoids collision. This shifts the focus compared to previous applications, illustrating that safety can have varying

Mental demand: how much thinking, deciding, or calculating was required to perform the task.
Physical demand: the amount and intensity of physical activity required to complete the task.
Temporal demand: the amount of time pressure involved in completing the task.
Effort: how hard does the participant have to work to maintain their level of performance?
Performance: the level of success in completing the task.
Frustration level: how insecure, discouraged, or secure or content the participant felt during the task.

Table 8.1: NASA Task Load Index

definitions depending on the context. The chapter presents results from a framework still in its early stages; for instance, the algorithm has only been applied to the end effector, while a whole-body implementation would significantly enhance safety.

Several technical aspects warrant attention. For example, relying on a single camera introduces limitations; if the user’s hand is obscured by the robot, the algorithm may fail. Additionally, the architecture lacks formalism, which would make the framework more comprehensive and mathematically robust for interested readers.

8.2 Challenges in human-robot interaction

The interest in collaborative robotics has been significantly rising, as robots are starting to work with humans rather than performing separate tasks (L. Zhang et al., 2020b). This scenario is typically referred to as Human-Robot Interaction (HRI), and a lot of research interest has been directed towards the development of novel algorithms and techniques to enhance the human-robot communication, coexistence and collaboration in various settings (Deshpande, 2022; Hogan & Sternad, 2013; Lauretti et al., 2017). An important aspect of HRI applications is represented by their accessibility to non-expert users (Calinon et al., 2007), where techniques such as Imitation Learning, or Learning from Demonstration, come into play (Schaal, 1999).

While control algorithms for robots are becoming more and more sophisticated, being able to handle complex tasks and generalize to several different scenarios (Chalvatzaki et al., 2019; Ijspeert et al., 2013), still important concerns

about safety are open. At a first glance, safety might be related to the concept of stability and robustness from control theory, but it actually differs from the latter in various aspects. Safety is objective, we know how to define specifications for a certain application to be safe for its user. If we consider for instance the obstacle avoidance application in Chapter 7, it was easy to define what was perceived as a safety constraint, i.e. avoiding collision with the human. However, safety can be subjective as well. Take as an example the guidance applications in Chapter 6 and 5: here users with different limitations/impairments might require some adaptation of the proposed framework in order to complete the task correctly.

These are just simple examples of how difficult it could be to define what safety is, to clarify its meaning, and cluster it such that proper metrics for human-robot interaction are defined (Onfiani et al., 2024; Zanchettin et al., 2015). Moreover, even if safety requirements appear to be quite defined for some applications, it is generally non-trivial to understand how to translate such safety concerns into commands that can be executed by the robots. Additionally, the unpredictability of human intentions when performing tasks increases the level of uncertainty and difficulty in elaborating a control framework that can react robustly in the presence of humans.

As shown in Chapter 5, sometimes the introduction of a human inside the loop of the controller might complicate the whole system architecture. In Sec. 5.3, the robot was oscillating in response to human-applied forces, indicating that the system was not robust at some working points. In this case, throughout several collections of data, it was possible to establish a practical stability criterion, where the human was modeled at the end as a finite delay.

While this methodology guaranteed an analytical study of the instability phenomenon, it is quite evident that it was mostly restricted to the proposed methodology and might not be effective or suitable in other scenarios. Technically speaking, a practical stability analysis requires many data to be acquired, which would require consistency in the experiments and time. Practically, this kind of analysis requires observing some instability phenomenon, which in general is better to avoid (especially if a human is involved!). This should raise some safety concerns.

Many robotic applications are starting to include subject-oriented analysis in the research. This enables the inclusion of users' opinions inside the evaluation of the proposed framework and sometimes provides an overall evaluation that is not just technical but also includes perceptions from the human side. A widely-used subject-related test is represented by the NASA task load index (NASA

TLX) (Cao et al., 2009; Hart, 1988), which is a tool for measuring and conducting a subjective mental workload (MWL) assessment. It allows you to determine the MWL of a participant while they are performing a task. It consists of asking subjects performing a certain exercise (like a robot co-manipulation task) to rate their score on an interval scale ranging from low (1) to high (20), answering the questions raised in Table 8.1. This represents a way to define a new method of evaluating robot applications, especially when safety is a major concern, suggesting that safety in human-robot interaction is something that cannot be addressed without a multidisciplinary approach, involving both technical and social aspects.

8.3 Future directions of research

Building on my doctoral research, I aim to explore several promising directions that extend the boundaries of robotic control, human-robot interaction, and system robustness. A key area of focus is the integration of machine learning with control theory to advance human-robot interaction, particularly addressing the sim-to-real transfer challenge. Learning-based techniques such as reinforcement learning (RL) offer an appealing pathway, as they bridge the gap between learning and control, enabling the development of control policies that optimize for robust, low-sensitivity trajectories in dynamic and uncertain environments. By leveraging RL frameworks, it becomes possible to craft systems that adaptively mitigate external perturbations (like contact with humans), ensuring reliability and performance even in complex scenarios (Alhousani et al., 2023; Peters & Schaal, 2008).

Safety and robustness remain pivotal in the development of human-in-the-loop robotic systems. The unpredictable nature of human behavior introduces a critical need for adaptive safety measures that extend beyond predefined constraints. One novel approach could be to explore methodologies that estimate safety constraints directly from human feedback, leveraging data-driven approaches to derive control strategies that inherently respect these learned boundaries. This approach can be further enhanced by incorporating closed-loop sensitivity measures, offering robust guarantees against the variability of human parameters, which are often difficult to quantify a priori (Giordano et al., 2018; Wasiela et al., 2023). These strategies will foster the creation of robotic systems that are both inherently safe and capable of dynamically adjusting to diverse operational contexts.

An intriguing extension of this work involves examining the bidirectional dy-

namics of human-robot interaction, particularly how robotic systems can provide meaningful feedback to humans. This feedback loop could play a transformative role in areas like intention estimation and rehabilitation robotics, where the robot's active participation enhances the overall system's effectiveness. For instance, learning-based techniques can be employed to infer safety boundaries from human demonstrations and refine control architectures that steer the interaction within those limits. This interplay not only ensures system robustness but also facilitates a more natural and intuitive human-robot collaboration.

From a geometric control perspective, significant potential lies in advancing the coupling of trajectories across different manifolds, such as those representing position and orientation (F. Abu-Dakka et al., 2024). Extending dynamical system encoding methods like Dynamic Movement Primitives (DMPs) and Gaussian Mixture Models (GMMs) to consider these couplings could lead to novel theoretical insights and practical applications (Calinon, 2020a; Paraschos et al., 2013; Saveriano, Abu-Dakka, & Kyrki, 2023). By exploring the geometric properties underlying these interactions, it may be possible to enhance precision and adaptability in robotic movements, contributing to the broader goal of creating versatile and efficient robotic systems.

Appendices

Geometry of curves

Throughout this manuscript, a lot of results leverage an arc-length parameterization of the curve, which math lies under the theory of curves in three-dimensional Euclidean space and in the plane (Sciavicco & Siciliano, 2012). This appendix section collects the main theoretical concepts from (Toponogov, 2006) which were used inside the thesis. As only theoretical results are of interest, demonstrations for the cited theorems are always omitted, but everything could be found in the book of professor V.A. Toponogov (Toponogov, 2006).

A.1 Preliminaries

Definition A.1.1 (*Homeomorphism*). If a map $\phi : M \rightarrow N$ is continuous and bijective, and if its inverse map $\psi = \phi^{-1} : N \rightarrow M$ is also continuous, then ϕ is a homeomorphism and M and N are said to be homeomorphic. In particular, given $\mathbf{x} = [x_1, x_2, \dots, x_m]^\top \in \mathbb{R}^m$, the Jacobian matrix of the differentiable map $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \phi_1}{\partial x_1} & \dots & \frac{\partial \phi_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial \phi_n}{\partial x_1} & \dots & \frac{\partial \phi_n}{\partial x_m} \end{pmatrix}.$$

If there exists a differentiable map $\psi : N \rightarrow M$ such that $\phi \circ \psi = \mathbf{I}$ and $\psi \circ \phi = \mathbf{I}$, then the differentiable map $\phi : M \rightarrow N$ is also said to be a *diffeomorphism*.

Theorem A.1.1 (Implicit function theorem). *Let $U \subset \mathbb{R}^n$ be an open set, $P \in U$ and $\phi : U \rightarrow \mathbb{R}^n$. If $\det J(P) \neq 0$ it follows that there exist neighborhoods V_P of P and $V_{\phi(P)}$ of $\phi(P)$ such that $\phi|_{V_P} : V_P \rightarrow V_{\phi(P)}$ is a diffeomorphism.*

Definition A.1.2 (*Regular curve*). A connected set $\mu \in \mathbb{R}^3$ (or $\mathbb{R}^2$) is a regular k -fold continuously differentiable curve if there is a homeomorphism $\phi : G \rightarrow \mu$, where G is a line segment $[a, b]$ or a circle of radius 1 satisfying the following

conditions:

- (1) $\phi \in \mathcal{C}^k$ ($k \geq 1$), (2) the rank of ϕ is maximal (equal to 1).

For $k = 1$ a curve μ is said to be smooth. With reference to Cartesian coordinates x, y, z in the $\mathbb{R}^3$ space, a mapping ϕ can be determined by functions $x(t), y(t), z(t)$ with $t \in [a, b]$. In this case, condition (1) above means that these functions belong to class $\mathcal{C}^k$, and the condition (2) means that the derivatives $x'(t), y'(t), z'(t)$ cannot simultaneously be zero. Note that the ' notations stands for the derivative w.r.t. t , which for the moment still maintains a generic connotation, and is not associated with time. The equations $x = x(t), y = y(t), z = z(t)$ are called parametric equations and can be diffeomorphic either to a line segment or a to circle. For the latter, being $t \in [a, b]$ the functions $x(t), y(t), z(t)$ are periodic on $\mathbb{R}$ with period $b - a$ and the curve is said to be closed.

Given the vector form of parametric equations of a curve, namely $\vec{\mu} = \vec{\mu}(t) = x_1(t)\vec{i} + y_1(t)\vec{j} + z_1(t)\vec{k}$ with unit axis versors $\vec{i}, \vec{j}, \vec{k}$, it follows from A.1.1- A.1.2 that the same curve can be given with different parameterizations, i.e.

$$\begin{aligned}\vec{\mu} &= \vec{\mu}_1(t) = x_1(t)\vec{i} + y_1(t)\vec{j} + z_1(t)\vec{k}, \quad t \in [a, b], \\ \vec{\mu} &= \vec{\mu}_2(\tau) = x_2(\tau)\vec{i} + y_2(\tau)\vec{j} + z_2(\tau)\vec{k}, \quad \tau \in [c, d].\end{aligned}$$

In this case, the vector functions $\vec{\mu}_1(t)$ and $\vec{\mu}_2(\tau)$ are related by a strictly monotonic transformation of parameters $t = t(\tau) : [c, d] \rightarrow [a, b]$ such that

- (1) $\vec{\mu}_1(t(\tau)) = \vec{\mu}_2(\tau)$,
(2) $t'(\tau) \neq 0 \quad \forall \tau \in [c, d]$.

Besides the parametric representation of a curve μ in $\mathbb{R}^3$ ($\mathbb{R}^2$), there exist also two other important representations to be mentioned.

Definition A.1.3 (*Explicit curve*). Recall a parametric representation, a curve is given in its explicit form everytimes instead of using t the variable x, y or z are employed, e.g. either $x = x, y = f_1(x), z = f_2(x)$; $x = f_1(y), y = y, z = f_2(y)$; or $x = f_1(z), y = f_2(z), z = z$.

This type of representation is especially convenient for a plane curve, since it coincides with the graph of some function f , and the curve may be written either in the form $y = f(x)$ or $x = f(y)$.

Definition A.1.4 (*Implicit curve*). Let a differentiable map be given by

$$\mathbf{f} : \mathbb{R}^3 \rightarrow \mathbb{R}^2, \quad \mathbf{f} = [f_1(x, y, z), f_2(x, y, z)].$$

If $(0, 0)$ is regular value of the map $\mathbf{f}$, that is

$$\text{rank}(\mathbf{J}) = 2 \quad \text{with Jacobian matrix } \mathbf{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \end{pmatrix},$$

then from A.1.1 it follows that each connected component of the set $T = \mathbf{f}^{-1}(0, 0)$ is a smooth regular curve in $\mathbb{R}^3$. In other words, under the above given conditions a set of points in $\mathbb{R}^3$ whose coordinates satisfy the system of equations

$$f_1(x, y, z) = 0, \quad f_2(x, y, z),$$

forms a smooth regular curve.

A.2 Tangent line and osculating plane

Let a smooth curve μ be given by the parametric equations

$$\vec{\mu} = \vec{\mu}(t) = x(t)\vec{i} + y(t)\vec{j} + z(t)\vec{k},$$

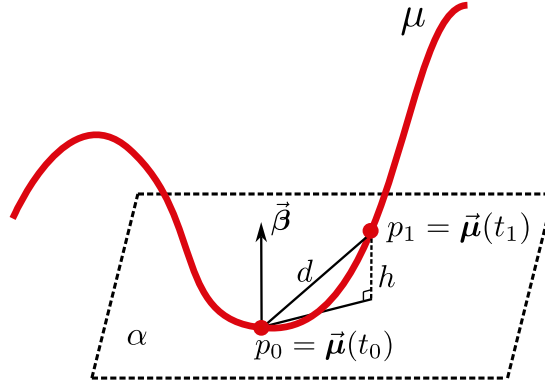
then the velocity vector of $\vec{\mu}(t)$ calculated at $t = t_0$ is the derivative $\vec{\mu}'(t_0) = x'(t_0)\vec{i} + y'(t_0)\vec{j} + z'(t_0)\vec{k}$.

Definition A.2.1 (*Tangent line*). The tangent line to a smooth curve μ at a point $p = \vec{\mu}(t_0)$ is the straight line through the point $p = \vec{\mu}(t_0) \in \mu$ in the direction of the velocity vector $\vec{\mu}'(t_0)$.

It is then convenient to give a geometric definition of the osculating plane. Let a plane α with a unit binormal $\vec{\beta}$ pass through a point $p_0 = \vec{\mu}(t_0)$ of a curve μ . Denote then by d the length of the chord of μ joining the points $p_0 = \vec{\mu}(t_0)$ and $p_1 = \vec{\mu}(t_1)$, and by h the length of the perpendicular dropped from p_1 onto the plane α .

Definition A.2.2 (*Osculating plane*). A plane α is called an osculating plane to a curve μ at a point $p_0 = \vec{\mu}(t_0)$ if

$$\lim_{d \rightarrow 0} \frac{h}{d^2} = \lim_{t_1 \rightarrow t_0} \frac{h}{d^2} = 0.$$


 Figure A.1: Osculating plane α visualization for Definition A.2.2.

Theorem A.2.1 (Existence of osculating plane). *At each point $p_0 = \vec{\mu}(t_0)$ of a regular curve μ of class $\mathcal{C}^k$ ($k \geq 2$) there is an osculating plane α , and the vectors $\vec{\mu}'(t_0)$ and $\vec{\mu}''(t_0)$ are orthogonal to its binormal vector $\vec{\beta}$.*

Now, let us analyze a generic point p_0 of μ . If the curve is regular one can always define three versors, i.e. unit vectors, at that point (Sciavicco & Siciliano, 2012). While the direction of those versors depends only curve's geometry, their sense depends on the orientation of the curve induced by the parametrization t , as discussed later in the appendix. Referring to Fig. A.2, the three versors $\vec{\tau}$, $\vec{n}$ and $\vec{\beta}$ follows from the definition

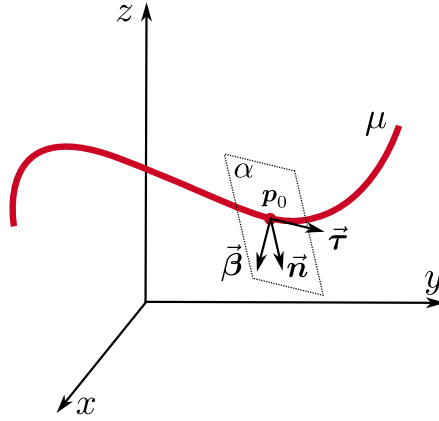
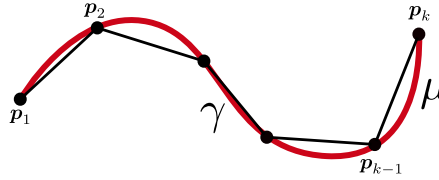
$$\vec{\tau} = \frac{\vec{\mu}'}{\|\vec{\mu}'\|}, \quad \vec{n} = \frac{\vec{\mu}''}{\|\vec{\mu}''\|}, \quad \text{and} \quad \vec{\beta} = \vec{\tau} \times \vec{n}.$$

We previously introduced the binormal versor $\vec{\beta}$ in Definition A.2.2. The tangential versor $\vec{\tau}$ lies along the line passing through p_0 and tangential to μ , with its direction defined by the orientation of the curve. Finally, the normal versor $\vec{n}$ is orthogonal to $\vec{\tau}$ and lies within the osculating plane α .

A.3 Length of a curve and arc length parametrization

The last section refers to the concept of orientation of a curve. The term orientation comes directly from the parametrization t of the curve, for which t is strictly monotonic. An important type of parametrization is the so-called arc-length parametrization, which follows from the definition of length of a curve.

Let μ be a smooth curve, i.e. $\mu \in \mathcal{C}^1$, and $\vec{\mu} = \vec{\mu}(t)$ its parametrization with $t \in [a, b]$. Let γ be a polygonal line composed of line segments passing through


 Figure A.2: Characteristic versors of a curve μ at a point $p(t)$.

 Figure A.3: Inscribed polygonal line γ in a curve μ .

adjacent points $p_1 = \vec{\mu}(t_1)$, $p_2 = \vec{\mu}(t_2)$, ..., $p_k = \vec{\mu}(t_k)$, as depicted in Fig. A.3. To each polygonal line γ corresponds its length $l(\gamma) = \sum_{i=1}^{k-1} d(p_i, p_{i+1})$, where d denotes the Euclidean distance between the points p_i and p_{i+1} . Indeed, higher the number k of segments, finer the approximation to the curve μ will be, thus leading to $l(\mu) \approx l(\gamma)$. In particular, it can be demonstrated that given a regular smooth curve its length is

$$l(\mu) = \int_a^b |\vec{\mu}'(t)| dt.$$

Now denote by $\Gamma(\mu)$ the set of all inscribed polygonal lines in a curve μ . If for any polygonal line $\gamma \in \Gamma(\mu)$ it holds $l(\gamma) < \infty$, then μ is said to be rectifiable. This property is important because, for rectifiable curves, one can define the so-called arc length parametrization.

Let μ be an oriented and rectifiable curve, where the term oriented means that μ has a pre-defined start and end. Take an arbitrary point $p_0 \in \mu$ to which associate the value $s = 0$. Taken any other point $p \in \mu$ it corresponds a value s which is equal to the arc length $l(pp_0)$, which sign is positive (+) if p follows p_0 , negative (−) if p comes before p_0 . If μ admits a smooth regular parametrization $\vec{\mu} = \vec{\mu}(t)$, then its arc length parametrization is also smooth and regular. By

taking into account the sign, one can write

$$l(\mathbf{p}\mathbf{p}_0) = s(t) = \int_0^t |\vec{\mu}'(t)| dt.$$

The function $s(t)$ is differentiable with $\frac{ds}{dt} = |\vec{\mu}'(t)| > 0$, meaning that $s(t)$ is monotonic. Therefore, there is an inverse function $t = t(s)$ such that

$$\frac{dt}{ds} = \frac{1}{|\vec{\mu}'(t(s))|}.$$

The arc length parametrization of a curve $\mu : \vec{\mu} = \vec{\mu}(s)$ is defined by the formula $\vec{\mu}(s) = \vec{\mu}(t(s))$. This parametrization is said to be unit speed, as from the differentiability of the vector function $\vec{\mu}(s)$ it holds

$$|\vec{\mu}'(s)| = \left| \vec{\mu}(t) \cdot \frac{dt}{ds} \right| = \frac{|\vec{\mu}'(t)|}{|\vec{\mu}'(t)|} = 1,$$

for which the curve is regular.

Orientation

B.1 Introduction

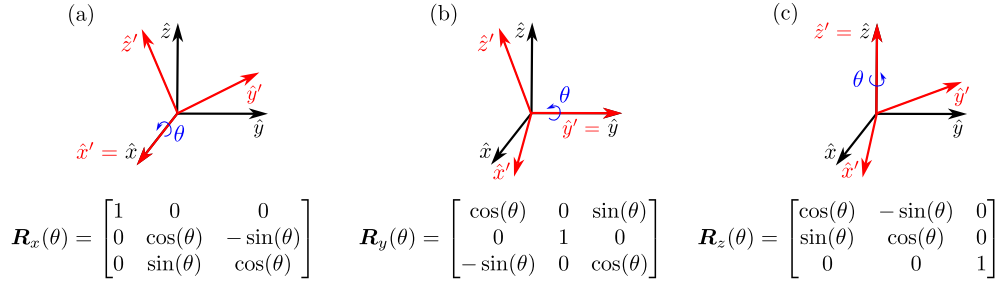


Figure B.1: Rotation matrices with respect to the x (a), y (b) and z (c) axes.

Rotations about some axis through the origin can be represented by, if considering for instance in $\mathbb{R}^3$, a 3×3 orthogonal matrix with determinant 1 (Sciavicco & Siciliano, 2012). Some summarizing and elementary examples are shown in Fig. B.1. However, the matrix representation is redundant because only four of its nine elements are independent. Additionally, the geometric interpretation of such a matrix is not immediately clear and requires several computational steps to extract the rotation axis and angle. Moreover, composing two rotations involves computing the product of their corresponding matrices, which requires twenty-seven multiplications and eighteen additions.

Quaternions provide a more efficient approach for analyzing rotations in $\mathbb{R}^3$. Represented as a 4-tuple, quaternions offer a more concise representation compared to rotation matrices. Their geometric interpretation is also straightforward, as the rotation axis and angle can be easily recovered. Furthermore, quaternion composition is computationally efficient, requiring only sixteen multiplications and twelve additions. This Appendix is meant to be a short introduction to quaternions, following the notes provided in (Jia, 2008).

B.2 Algebra of Quaternion

In $\mathbb{R}^3$ the standard orthonormal basis is given by the three unit vectors $\hat{\mathbf{i}} = (1, 0, 0)$, $\hat{\mathbf{j}} = (0, 1, 0)$ and $\hat{\mathbf{k}} = (0, 0, 1)$. These unit vector can be used to define a quaternion q , which is described as a sum of a scalar q_0 and a vector $\mathbf{q} = (q_1, q_2, q_3)$, i.e.

$$q = q_0 + \mathbf{q} = q_0 + q_1\hat{\mathbf{i}} + q_2\hat{\mathbf{j}} + q_3\hat{\mathbf{k}}.$$

Given two quaternions $q = q_0 + q_1\hat{\mathbf{i}} + q_2\hat{\mathbf{j}} + q_3\hat{\mathbf{k}}$ and $p = p_0 + p_1\hat{\mathbf{i}} + p_2\hat{\mathbf{j}} + p_3\hat{\mathbf{k}}$, it is possible to define the addition (+) and product ($\otimes$) of two quaternions as:

$$\begin{aligned} p + q &= (p_0 + q_0) + (p_1 + q_1)\hat{\mathbf{i}} + (p_2 + q_2)\hat{\mathbf{j}} + (p_3 + q_3)\hat{\mathbf{k}}, \\ p \otimes q &= p_0q_0 - \mathbf{p} \cdot \mathbf{q} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}, \end{aligned}$$

with $\mathbf{p} = (p_1, p_2, p_3)$ and $\mathbf{q} = (q_1, q_2, q_3)$ being the vector parts respectively of p and q .

The conjugate of quaternion q is denoted by q^* and is defined as

$$q^* = q_0 - \mathbf{q} = q_0 - q_1\hat{\mathbf{i}} - q_2\hat{\mathbf{j}} - q_3\hat{\mathbf{k}},$$

for which the following properties hold

$$\begin{aligned} (q^*)^* &= q_0 - (-\mathbf{q}) = q, \\ q + q^* &= 2q_0, \\ q^* \otimes q &= (q_0 - \mathbf{q}) \otimes (q_0 + \mathbf{q}) = q_0q_0 - (-\mathbf{q}) \cdot \mathbf{q} + q_0\mathbf{q} + (-\mathbf{q})q_0 + (-\mathbf{q}) \times \mathbf{q}, \\ &= q_0^2 + \mathbf{q} \cdot \mathbf{q} = q_0^2 + q_1^2 + q_2^2 + q_3^2 = q \otimes q^*. \end{aligned}$$

In particular the norm of a quaternion q is the scalar $|q| = \sqrt{q^* \otimes q}$, which is indeed equal to 1 for unit quaternions. Finally, the inverse of a quaternion is defined as

$$q^{-1} = \frac{q^*}{|q|^2},$$

that easily leads to $q^{-1} \otimes q = q \otimes q^{-1} = 1$.

B.3 Quaternion as rotation operator

A quaternion q has four components, thus is defined on $\mathbb{R}^4$, while a vector $\mathbf{v} \in \mathbb{R}^3$ can be seen as a quaternion whose real part is zero. Since q and $\mathbf{v}$ belong to different spaces, namely $\mathbb{R}^4$ and $\mathbb{R}^3$, the way q acts over $\mathbf{v}$ can be defined by the

operator

$$L_q(\mathbf{v}) \triangleq q \otimes \mathbf{v} \otimes q^* = [\dots] = (q_0^2 - \|\mathbf{q}\|^2)\mathbf{v} + 2q_0(\mathbf{q} \times \mathbf{v}) + 2(\mathbf{q} \cdot \mathbf{v})\mathbf{q}.$$

In particular, if q is a unit quaternion, it can be observed that (i) the operator does not change the length of the vector $\mathbf{v}$, namely $\|L_q(\mathbf{v})\| = \|\mathbf{v}\|$ and (ii) the direction of $\mathbf{v}$, if along $\mathbf{q}$, is not affected by the operator L_q (Jia, 2008). These considerations suggest that the operator L_q behaves like a rotation about $\mathbf{q}$, as formally stated in the following theorem.

Theorem B.3.1 (Unit quaternion rotation). *For any vector $\mathbf{v} \in \mathbb{R}^3$ and for any unit quaternion $q = q_0 + \mathbf{q} = \cos \frac{\theta}{2} + \hat{\mathbf{u}} \sin \frac{\theta}{2}$, the operator $L_q(v) = q \otimes \mathbf{v} \otimes q^*$ is equivalent to a rotation of the vector $\mathbf{v}$ through an angle θ about the axis $\hat{\mathbf{u}}$.*

For instance, given that rotations are represented as well by quaternions, rotation matrices have six degrees of redundancy more than quaternions. Indeed, given the operator $L_q(\mathbf{v})$, the corresponding rotation matrix $\mathbf{R}$ can be computed. We start by writing the operator $L_q(\mathbf{v})$ as a matrix product, namely

$$L_q(\mathbf{v}) = ((q_0^2 - \|\mathbf{q}\|^2)\mathbf{I}_3 + 2\mathbf{q}\mathbf{q}^\top + 2q_0[\mathbf{q}]_\times)\mathbf{v},$$

denoting with $\mathbf{I}_3$ the 3×3 identity matrix and with $[\mathbf{q}]_\times$ the skew symmetric matrix containing the components of $\mathbf{q}$, i.e.

$$[\mathbf{q}]_\times = \begin{pmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{pmatrix},$$

resulting in the cross product with $\mathbf{v}$. Given that $\mathbf{v}$ is an arbitrary vector, the rotation matrix $\mathbf{R}$ corresponding to the quaternion q is expressed as:

$$\mathbf{R} = (q_0^2 - \|\mathbf{q}\|^2)\mathbf{I}_3 + 2\mathbf{q}\mathbf{q}^\top + 2q_0[\mathbf{q}]_\times.$$

Finally, the rotation of $\mathbf{v}$ under the operator L_q can also be interpreted from the perspective of an observer attached to the vector $\mathbf{v}$. What the observer sees is that the coordinate frame rotates through the angle $-\theta$ about the same axis defined by the quaternion, for which the following theorem holds.

Theorem B.3.2. *For any vector $\mathbf{v} \in \mathbb{R}^3$ and for any unit quaternion $q = q_0 + \mathbf{q} = \cos \frac{\theta}{2} + \hat{\mathbf{u}} \sin \frac{\theta}{2}$, the operator $L_q(v) = q \otimes \mathbf{v} \otimes q^*$ is equivalent to a rotation of the coordinate frame of an angle θ through the axis $\hat{\mathbf{u}}$ while $\mathbf{v}$ is not rotated.*

Signal alignment metrics

This part provides a more detailed description of the metrics used in Chapter 3.3. In the explanation, we will always consider two generic series $\mathbf{x} \in \mathbf{R}^{d \times N}$ and $\mathbf{y} \in \mathbf{R}^{d \times M}$, with d being the dimension, N and M being respectively the number of samples of $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^\top$ and $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M]^\top$. While the explanation is given for the dyadic case, i.e. considering two series, the metrics can be easily extended to the multivariate case as explained in (Hudson et al., 2023).

C.1 Alignment Algorithms Comparisons Based on LASA Handwriting Dataset

C.1.1 Rho (ρ)

For the definition of ρ we used the following as reference (Richardson et al., 2012). The metrics ρ was introduced to assess group level synchrony of time-series by analyzing their phase. It does so by calculating an aggregate relative phase across multiple signals to measure how closely each signal's phase aligns with the overall phase. Note that the ρ metric was introduced for uni-dimensional series; as we are working with d -dimensional series, without loss of consistency we can calculate ρ for each individual dimension and then average its value. We will thus indicate with $\mathbf{x} = [x_1, x_2, \dots, x_N]^\top$ ($\mathbf{y} = [y_1, y_2, \dots, y_M]^\top$) a generic one-dimensional time-series selected by its d -dimensional counterpart $\mathbf{x}$ ($\mathbf{y}$). Moreover, this metric is proposed for time series of the same length, i.e. $N = M$. Some re-sampling should be introduced in case this condition is not met.

To compute ρ we start first by compute the phase time-series $\theta_{\mathbf{x}}, \theta_{\mathbf{y}} \in [-\pi, +\pi]$ from, respectively, $\mathbf{x}$ and $\mathbf{y}$. Then the group phase time-serie q_t at instant t , or cluster serie, is calculated as:

$$q_t = \text{atan2}(\hat{q}_t), \quad \text{with} \quad \hat{q}_t = \frac{1}{2} [\exp(i\theta_{\mathbf{x},t}) + \exp(i\theta_{\mathbf{y},t})], \quad (\text{C.1})$$

where i is the imaginary number and atan2 is the 2-argument arc-tangent; note that q_t is still in radian, i.e. $q_t \in [-\pi, +\pi]$.

Finally, we proceed by compute the phase cluster-displacement ϕ for each component, that is:

$$\phi_{\mathbf{x},t} = \theta_{\mathbf{x},t} - q_t \text{ and } \phi_{\mathbf{y},t} = \theta_{\mathbf{y},t} - q_t, \quad (\text{C.2})$$

then compute its mean value $\bar{\phi}$ as

$$\begin{aligned} \bar{\phi}_{\mathbf{x},t} &= \text{atan2}(\hat{\phi}_{\mathbf{x},t}), \text{ with } \hat{\phi}_{\mathbf{x},t} = \frac{1}{N} \sum_{t=1}^N \exp(i\phi_{\mathbf{x},t}), \text{ and} \\ \bar{\phi}_{\mathbf{y},t} &= \text{atan2}(\hat{\phi}_{\mathbf{y},t}), \text{ with } \hat{\phi}_{\mathbf{y},t} = \frac{1}{N} \sum_{t=1}^N \exp(i\phi_{\mathbf{y},t}). \end{aligned} \quad (\text{C.3})$$

Finally, one can obtain calculate the degree of synchronization ρ of the $(\mathbf{x}, \mathbf{y})$ dyad by first calculating its instantaneous value ρ_t as:

$$\rho_t = \left| \frac{1}{2} \left\{ \exp[i(\phi_{\mathbf{x},t} - \bar{\phi}_{\mathbf{x}})] + \exp[i(\phi_{\mathbf{y},t} - \bar{\phi}_{\mathbf{y}})] \right\} \right|, \quad (\text{C.4})$$

then calculating ρ as the average $\rho = 1/N \sum_{t=1}^N \rho_t$. Note that $\rho \in [0, 1]$ assumes higher values for a larger degree of group synchronization.

C.1.2 Entropy (H)

For the definition of H we used the following reference (Stevens & Galloway, 2014). Entropy represents the average level of unpredictability of outcomes drawn from a probability distribution. A distribution with low entropy indicates that outcomes are more predictable, while higher entropy reflects greater uncertainty and unpredictability.

Given a discrete random variable X , which takes values in the set $\mathcal{X}$ and is distributed according to $p : \mathcal{X} \rightarrow [0, 1]$ such that $p(x) := \mathbf{P}[X = x]$, the entropy H of X can be computed as:

$$H(X) = \mathbf{E}[I(X)] = \mathbf{E}[-\log p(X)], \quad (\text{C.5})$$

where $\mathbf{E}$ is the expected value operator, and $I(X)$ describes the information content of X and is itself a random variable. One can write H explicitly as

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log_b p(x), \quad (\text{C.6})$$

denoting with b the base of the used logarithm (in the experimental tests we used $b = 10$). Without loss of consistency we can use as random variable X a sample $\mathbf{x}_t$ of the serie $\mathbf{x}$, the same for $\mathbf{y}$. Moreover, as we are considering a dyad of series, one can define the conditional entropy of two variables $\mathbf{x}_t$ and $\mathbf{y}_t$ taking values from $\mathbf{x}$ and $\mathbf{y}$ respectively, as:

$$H(\mathbf{x}|\mathbf{y}) = - \sum_{i=1}^N \sum_{j=1}^M p_{\mathbf{x},\mathbf{y}}(\mathbf{x}_i, \mathbf{y}_j) \log \frac{p_{\mathbf{x},\mathbf{y}}(\mathbf{x}_i, \mathbf{y}_j)}{p_{\mathbf{y}}(\mathbf{y}_j)}, \quad (\text{C.7})$$

where $p_{\mathbf{x},\mathbf{y}}(\mathbf{x}_i, \mathbf{y}_j) := P[\mathbf{x} = \mathbf{x}_i, \mathbf{y} = \mathbf{y}_j]$ and $p_{\mathbf{y}}(\mathbf{y}_j) = P[\mathbf{y} = \mathbf{y}_j]$.

C.1.3 Sum-Normalized Cross-Spectral Density (snCSD)

For the definition of snCSD we used the following reference (Hudson et al., 2023). This metric was proposed by the authors as an alternative to the averaged coherence metric (White, 1984) when dealing with noisy signals. An important consideration for the averaged coherence metric is that it averages coherence values across frequencies, ignoring the relative amplitude at different frequencies. This can be problematic for signals with Gaussian noise, which affects all frequencies, while the actual signal may be concentrated in a limited frequency range. In such cases, noise can dominate the meaningful signal, especially with high sampling rates that increase the number of frequency components. Applying filters, like a bandpass filter, could improve the metric's reliability by removing irrelevant noise frequencies.

However, noise can significantly impact the results if no filtering is applied. To address this the metric snCSD, instead of normalizing at each frequency, postpones normalization after summing the cross-spectral density (CSD) across frequencies. This allows amplitude information in the CSD to influence the final result. Specifically, $\text{snCSD} \in [0, 1]$ and can be defined as follows (Hudson et al., 2023):

$$\text{snCSD} = \frac{\sum_n |CSD_n(\mathbf{x}, \mathbf{y})| \cdot |CSD_n(\mathbf{x}, \mathbf{y})|}{\sum_n |CSD_n(\mathbf{x}, \mathbf{x})| \cdot |CSD_n(\mathbf{y}, \mathbf{y})|}, \quad (\text{C.8})$$

where the summation is made over each n -frequency component of the CSD. Note that $\text{snCSD} \in [0, 1]$ tends to assume higher values every time $\mathbf{x}$ can accurately approximate $\mathbf{y}$, and vice-versa, therefore one can expect $\text{snCSD} \rightarrow 1$ if the signals are well synchronized.

C.1.4 ℓ_2 norm error

The ℓ_2 norm error is a measure of approximation accuracy. Given two series $\mathbf{x}$ and $\mathbf{y}$ of the same length, i.e. $N = M$ in this case, this metric computes the sum of each error $\|\mathbf{x}_t - \mathbf{y}_t\|_2$, that is:

$$\ell_2 = \sum_{t=1}^N \|\mathbf{x}_t - \mathbf{y}_t\|_2, \quad (\text{C.9})$$

where $\|\cdot\|_2$ is the Euclidean norm.

C.2 Skill Synthesis from Demonstrations

C.2.1 Hausdorff distance d_H

For the definition of d_H we used the following reference (Donoso et al., 2008). In particular, one can define d_H as:

$$d_H = \max \left\{ \sup_{\mathbf{x}_t \in \mathbf{x}} d(\mathbf{x}_t, \mathbf{y}), \sup_{\mathbf{y}_t \in \mathbf{y}} d(\mathbf{x}, \mathbf{y}_t) \right\}, \quad (\text{C.10})$$

where $d(\cdot, \cdot)$ can be defined as:

$$d(\mathbf{x}_t, \mathbf{y}) = \inf_{\mathbf{y}_t \in \mathbf{y}} \|\mathbf{x}_t - \mathbf{y}_t\| \quad \text{and} \quad d(\mathbf{x}, \mathbf{y}_t) = \inf_{\mathbf{x}_t \in \mathbf{x}} \|\mathbf{x}_t - \mathbf{y}_t\|. \quad (\text{C.11})$$

C.2.2 Dynamic Time Warping (DTW) score d_{DTW}

For the definition d_{DTW} we used the following reference (Muller, 2007). Given two generic series $\mathbf{x}$ and $\mathbf{y}$, d_{DTW} is defined as:

$$d_{DTW}(\mathbf{x}, \mathbf{y}) = \min_{\pi \in \mathcal{A}(\mathbf{x}, \mathbf{y})} \sum_{(i,k) \in \pi} d(\mathbf{x}_i, \mathbf{y}_k), \quad (\text{C.12})$$

where $\mathcal{A}(\mathbf{x}, \mathbf{y})$ is the set of all possible alignments between $\mathbf{x}$ and $\mathbf{y}$, while $d(\mathbf{x}_i, \mathbf{y}_k) = \|\mathbf{x}_i - \mathbf{y}_k\|^2$ is the squared Euclidean distance.

Additional Material to Chapter 6

D.1 Transfer matrices computation

Consider the linear system with equation

$$\mathbf{x}_{t+1} = \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t, \quad \forall t \in [1, \dots, T],$$

being $\mathbf{x}_t \in \mathbb{R}^D$ and $\mathbf{u}_t \in \mathbb{R}^d$ the state and control commands, respectively (Calion, 2023b). It is possible to express such linear system as a function of the initial state $\mathbf{x}_1$, i.e.:

$$\begin{aligned} \mathbf{x}_2 &= \mathbf{A}_1 \mathbf{x}_1 + \mathbf{B}_1 \mathbf{u}_1, \\ \mathbf{x}_3 &= \mathbf{A}_2 \mathbf{x}_2 + \mathbf{B}_2 \mathbf{u}_2 = \mathbf{A}_2 (\mathbf{A}_1 \mathbf{x}_1 + \mathbf{B}_1 \mathbf{u}_1) + \mathbf{B}_2 \mathbf{u}_2, \\ &\vdots \\ \mathbf{x}_T &= \left(\prod_{t=1}^{T-1} \mathbf{A}_{T-t} \right) \mathbf{x}_1 + \left(\prod_{t=1}^{T-2} \mathbf{A}_{T-t} \right) \mathbf{B}_1 \mathbf{u}_1 + \left(\prod_{t=1}^{T-3} \mathbf{A}_{T-t} \right) \mathbf{B}_2 \mathbf{u}_2 + \dots + \mathbf{B}_{T-1} \mathbf{u}_{T-1}, \end{aligned}$$

which can be translate in the matrix form $\mathbf{x} = \mathbf{S}_x \mathbf{x}_1 + \mathbf{S}_u \mathbf{u}$ as follows

$$\underbrace{\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A}_1 \\ \mathbf{A}_2 \mathbf{A}_1 \\ \vdots \\ \prod_{t=1}^{T-1} \mathbf{A}_{T-t} \end{bmatrix}}_{\mathbf{S}_x} \mathbf{x}_1 + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_2 \mathbf{B}_1 & \mathbf{B}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\prod_{t=1}^{T-2} \mathbf{A}_{T-t} \right) \mathbf{B}_1 & \left(\prod_{t=1}^{T-3} \mathbf{A}_{T-t} \right) \mathbf{B}_2 & \dots & \mathbf{B}_{T-1} \end{bmatrix}}_{\mathbf{S}_u} \underbrace{\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{T-1} \end{bmatrix}}_{\mathbf{u}},$$

where $\mathbf{S}_x \in \mathbb{R}^{DT \times D}$, $\mathbf{x}_1 \in \mathbb{R}^D$, $\mathbf{S}_u \in \mathbb{R}^{dT \times d(T-1)}$ and $\mathbf{u} \in \mathbb{R}^{d(T-1)}$.

D.2 Inverse LQT

The selection of $\mathbf{Q}$ and R in the experiments of Chapter 6.4, started from the computation of first attempt values by using an inverse LQR approach as specified in (Priess et al., 2014).

We started by choosing a relative low value for the input weight R (for our experiments $R = 1e - 5$). Then, given $\mathbf{Q} = \text{diag}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$, inverse LQR is used to retrieve the values of $\mathbf{c}$ once a desired behaviour for the controller $\mathbf{u}$ is known. The inverse LQR can be formulated as follows: given a controller $\mathbf{u}^*$ that optimally solves the LQR problem, find the values of $\mathbf{Q}$ and $\mathbf{R}$ that replicate such optimal solution (Priess et al., 2014). The key takeaway in our application was to use inverse LQR such that, given the similarities between arm movements and minimum jerk trajectories (Todorov & Jordan, 1998), the parameter $\mathbf{Q}$ and R can help to express the features of a recorded movement.

Given a curve $\boldsymbol{\mu}(s)$ defined in the Euclidean space $\mathbb{R}^3$, we define our jerk controller $u_t = \ddot{s}_t$ in the phase space and minimize the cost functional (Amirabdollahian et al., 2002):

$$c = \sum_{t=1}^T \|\ddot{\boldsymbol{\mu}}(s_t)\|^2 \quad (\text{D.1})$$

subject to the dynamics of $\mathbf{s}_t$ (please see (14) in the article), and along the time instants $t = 1, 2, \dots, T$. It can be observed that, for $L_t = \|\ddot{\boldsymbol{\mu}}(s_t)\|^2$, the dependence of L_t on u_t can be made explicit, i.e. (Todorov & Jordan, 1998)

$$L_t = \|\boldsymbol{\mu}'''(s_t)\dot{s}_t^3 + 3\boldsymbol{\mu}''(s_t)\dot{s}_t\ddot{s}_t + \boldsymbol{\mu}'(s_t)u_t\|^2.$$

Minimizing the cost in (D.1), one obtains the minimum jerk command u_t^* , and therefore the phase profile s_t^* , given a duration of T instants (Sharkawy, 2021).

For the computation of $\mathbf{Q} = \text{diag}(\mathbf{c}_1, \mathbf{c}_2)$, we now consider the end effector (EE) state $\mathbf{x}_t$. Without loss of consistency, we optimize the tracking problem with respect to $\mathbf{x}_t$ instead of s_t , with an acceleration command $\mathbf{u}_t$. The LQT problem can be transformed into an LQR by setting (Calinon, 2023b):

$$c = \tilde{\mathbf{x}}_T^\top \tilde{\mathbf{Q}}_T \tilde{\mathbf{x}}_T + \sum_{t=1}^{T-1} (\tilde{\mathbf{x}}_t^\top \tilde{\mathbf{Q}}_t \tilde{\mathbf{x}}_t + \mathbf{u}_t^\top \mathbf{R} \mathbf{u}_t), \quad (\text{D.2})$$

subject to the dynamical system with augmented state

$$\underbrace{\begin{bmatrix} \mathbf{x}_{t+1} \\ 1 \end{bmatrix}}_{\tilde{\mathbf{x}}_{t+1}} = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}}_{\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} \mathbf{x}_t \\ 1 \end{bmatrix}}_{\tilde{\mathbf{x}}_t} + \underbrace{\begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}}_{\tilde{\mathbf{B}}} \mathbf{u}_t, \quad (\text{D.3})$$

where $\mathbf{A}$ and $\mathbf{B}$ are computed as a second order integrator, and augmented tracking weight

$$\tilde{\mathbf{Q}}_t = \begin{bmatrix} \mathbf{Q}^{-1} + \boldsymbol{\mu}_t^* \boldsymbol{\mu}_t^{*\top} & \boldsymbol{\mu}_t^* \\ \boldsymbol{\mu}_t^{*\top} & 1 \end{bmatrix}. \quad (\text{D.4})$$

Notice that $\boldsymbol{\mu}_t^{*\top} = [\boldsymbol{\mu}(s_t^*)^\top, \dot{\boldsymbol{\mu}}(s_t^*)^\top]$, with $\dot{\boldsymbol{\mu}}(s_t^*) = \boldsymbol{\mu}'(s_t^*)\dot{s}_t^*$. This choice equals to require the EE to follow a minimum jerk trajectory along the curve $\boldsymbol{\mu}$ (Biess et al., 2007).

The problem in (D.2) admits an optimal control law $\hat{\mathbf{u}}_t = -\tilde{\mathbf{K}}_t \tilde{\mathbf{x}}_t$, with $\tilde{\mathbf{K}}_t$ the augmented feedback gain matrix. Considering $\tilde{\boldsymbol{\mu}}_t^{*\top} = [\boldsymbol{\mu}_t^{*\top}, \mathbf{1}^\top]$, one can compute the optimal feedback $\mathbf{K}_t^*$ by minimizing the quadratic cost $c = \sum_{t=1}^T \mathbf{e}^\top \mathbf{e}$ with residual $\mathbf{e} = \tilde{\mathbf{x}}_t - \tilde{\boldsymbol{\mu}}_t$, provided from (D.3) that the state trajectory is $\tilde{\mathbf{x}}_{t+1} = (\tilde{\mathbf{A}} - \tilde{\mathbf{B}}\tilde{\mathbf{K}}_t)\tilde{\mathbf{x}}_t$. Next, by adopting an approach similar to (Priess et al., 2014), the values of $\mathbf{c}$ that defines the precision matrix $\mathbf{Q}(\mathbf{c})$ are obtained solving

$$\hat{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{t=1}^{T-1} \|\tilde{\mathbf{K}}_t(\mathbf{c}) - \mathbf{K}_t^*\|_F^2, \quad (\text{D.5})$$

where $\|\cdot\|_F$ denotes the Frobenius norm. In particular, we have that

$$\tilde{\mathbf{K}}_{t-1}(\mathbf{c}) = (\tilde{\mathbf{B}}^\top \mathbf{P}_t(\mathbf{c}) \tilde{\mathbf{B}} + \mathbf{R})^{-1} \tilde{\mathbf{B}}^\top \mathbf{P}_t(\mathbf{c}) \tilde{\mathbf{A}},$$

where $\mathbf{P}_{t-1}(\mathbf{c})$ comes from the recursive evaluation of the discrete Riccati equation, namely

$$\mathbf{P}_{t-1}(\mathbf{c}) = \tilde{\mathbf{A}}^\top \mathbf{P}_t(\mathbf{c}) \tilde{\mathbf{A}} + \tilde{\mathbf{Q}}_{t-1}(\mathbf{c}) - \tilde{\mathbf{A}}^\top \mathbf{P}_t(\mathbf{c}) \tilde{\mathbf{B}} (\tilde{\mathbf{B}}^\top \mathbf{P}_t(\mathbf{c}) \tilde{\mathbf{B}} + \mathbf{R})^{-1} \tilde{\mathbf{B}}^\top \mathbf{P}_t(\mathbf{c}) \tilde{\mathbf{A}}.$$

In the equation above, we have that $\mathbf{P}_T = \tilde{\mathbf{Q}}_T(\mathbf{c})$, where $\tilde{\mathbf{Q}}_t(\mathbf{c})$ is computed by substituting $\mathbf{Q}(\mathbf{c})$ in (D.4).

Acknowledgement

Il dottorato mi ha messo a contatto con varie persone, alcune per molto tempo, altre in maniera solo passeggera, alcune in ambito lavorativo, altre al di fuori di esso. Il lavoro è fatto di interazioni, e sono sicuro che ogni parola scambiata in questi tre anni ha avuto un'influenza su questa tesi, anche se non si parlava di robotica.

Dall'università di Modena e Reggio Emilia vorrei ringraziare in molti. Innanzitutto, Luigi per avermi dato la possibilità di affacciarmi alla ricerca, e Laura per le chiacchiere scambiate assieme. Ringrazio anche Francesca, per essere stata sempre presente qualora le chiedessi una mano. Non per ultimi ringrazio ragazze e ragazzi di Autolab e Ideativa, per la loro amicizia dentro fuori dal laboratorio. Elisa, Dario e Nicolò, senza di voi il dottorato avrebbe perso la sua parte più importante.

Nel dottorato è fondamentale essere motivati, soprattutto nei momenti in cui ci si sente disorientati. Voglio ringraziare André per la sua amicizia e per essere sempre riuscito a darmi un supporto, convincendomi a cominciare il dottorato, e dimostrandosi una spalla su cui poter contare durante tutto il percorso. Ringrazio inoltre i ragazzi del Robotics Learning and Interaction group a Martigny, per la loro accoglienza, per avermi trasmesso passione e nuovi punti di vista. Ringrazio anche Sylvain, per avermi dato la possibilità di unirmi al suo gruppo e per le brillanti idee con cui ha saputo, forse involontariamente, aiutarmi e motivarmi.

Ringrazio Cacau, per il suo affetto incondizionato, e tutti i miei amici brasiliani, per essere sempre pronti ad accogliermi con allegria. Ringrazio gli amici di Fogliano, Luca, Marta, Valeria, Simone, Stefano e Claudio, e anche quelli della band, Valeria, Andrea, Damiano e Giovanni. Non ho mai dato per scontato la fortuna di aver persone come voi su cui poter contare. Ringrazio anche Pier, per avermi insegnato che un ingegnere deve prima pensare, scrivere, e solo dopo fare.

Infine ringrazio la mia famiglia, Valeria e Matteo per il forte legame che abbiamo saputo creare e per gli stimoli che sappiamo darci a vicenda. Ringrazio per ultimi mamma e papà, per essere stati capaci di creare a casa un ambiente in cui voler sempre tornare, e per darmi supporto qualsiasi siano le mie decisioni. Vi voglio bene, grazie.

List of the author's publications

This section reports the list of the author's publications.

- G. Braglia, D. Tebaldi, A. Lazzaretti, and L. Biagiotti, "Arc-length-based warping for robot skill synthesis from multiple demonstrations", 2024, *arXiv preprint* (doi: [10.48550/arXiv.2410.13322](https://doi.org/10.48550/arXiv.2410.13322))
- G. Braglia, D. Onfiani, D. Tebaldi, A. Lazzaretti, and L. Biagiotti, "Spatial Sampling for Alignment of Robot Demonstrated Trajectories in Upper Limb Rehabilitation Tasks", 2024, *CoRL 2024 Workshop CoRoboLearn: Advancing Learning for Human-Centered Collaborative Robots* (url: <https://openreview.net/forum?id=3XTrpK3Rhs>)
- G. Braglia, D. Tebaldi and L. Biagiotti, "Phase-Independent Dynamic Movement Primitives with Applications to Human-Robot Interaction and Time Optimal Planning", 2024, *arXiv preprint* (doi: [10.48550/arXiv.2401.08238](https://doi.org/10.48550/arXiv.2401.08238))
- G. Braglia, S. Calinon and L. Biagiotti, "A Minimum-Jerk Approach to Handle Singularities in Virtual Fixtures", 2024, *IEEE Robotics and Automation Letters* (doi: [10.1109/LRA.2024.3469814](https://doi.org/10.1109/LRA.2024.3469814))
- G. Braglia, M. Tagliavini, F. Pini, and L. Biagiotti, "Online Motion Planning for Safe Human–Robot Cooperation Using B-Splines and Hidden Markov Models", 2023, *Robotics* (doi: [10.3390/robotics12040118](https://doi.org/10.3390/robotics12040118))

References

- Abu-Dakka, F., Saveriano, M., & Peternel, L. (2024). Learning periodic skills for robotic manipulation: Insights on orientation and impedance. *Robotics and Autonomous Systems*, 180, 104763.
- Abu-Dakka, F. J., Nemec, B., Jorgensen, J. A., Savarimuthu, T. R., Kruger, N., & Ude, A. (2015). Adaptation of manipulation skills in physical contact with the environment to reference force profiles. *Autonomous Robots*, 39, 199–217.
- Abu-Dakka, F. J., & Saveriano, M. (2020). Variable impedance control and learning—a review. *Frontiers in Robotics and AI*, 7.
- Alhousani, N., Saveriano, M., Sevinc, I., Abdulkuddus, T., Kose, H., & Abu-Dakka, F. J. (2023). Geometric reinforcement learning for robotic manipulation. *IEEE Access*, 11, 111492–111505.
- Amirabdollahian, F., Loureiro, R., & Harwin, W. (2002). Minimum jerk trajectory control for rehabilitation and haptic applications. *Proceedings 2002 IEEE ICRA*, 4, 3380–3385.
- Amor, H. B., Neumann, G., Kamthe, S., Kroemer, O., & Peters, J. (2014). Interaction primitives for human-robot cooperation tasks. *2014 IEEE international conference on robotics and automation (ICRA)*, 2831–2837.
- Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.
- Asl, H. J., Yamashita, M., Narikiyo, T., & Kawanishi, M. (2020). Field-based assist-as-needed control schemes for rehabilitation robots. *IEEE/ASME Transactions on Mechatronics*, 25(4), 2100–2111.
- Atkeson, C. (1989). Using local models to control movement. *Advances in neural information processing systems*, 2.
- Barmann, L., Kartmann, R., Peller-Konrad, F., Niehues, J., Waibel, A., & Asfour, T. (2024). Incremental learning of humanoid robot behavior from natural interaction and large language models. *Frontiers in Robotics and AI*, 11, 1455375.

- Berndt, D. J., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. *KDD Workshop*.
- Bettini, A., Marayong, P., Lang, S., Okamura, A. M., & Hager, G. D. (2004). Vision-assisted control for manipulation using virtual fixtures. *IEEE Transactions on Robotics*, 20(6), 953–966.
- Biagiotti, L., & Melchiorri, C. (2008). *Trajectory planning for automatic machines and robots*. Springer Science & Business Media.
- Biagiotti, L., & Melchiorri, C. (2013). Online trajectory planning and filtering for robotic applications via b-spline smoothing filters. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5668–5673.
- Bianco, C. G. L., & Ghilardelli, F. (2017). A scaling algorithm for the generation of jerk-limited trajectories in the operational space. *Robotics and Computer-Integrated Manufacturing*, 44, 284–295.
- Biess, A., Liebermann, D. G., & Flash, T. (2007). A computational model for redundant human three-dimensional pointing movements: Integration of independent spatial and temporal motor plans simplifies movement dynamics. *Journal of Neuroscience*, 27, 13045–13064.
- Billard, A., Calinon, S., Dillmann, R., & Schaal, S. (2008). Survey: Robot programming by demonstration. *Springer handbook of robotics*, 1371–1394.
- Bischof, B., Gluck, T., & Kugi, A. (2016). Combined path following and compliance control for fully actuated rigid body systems in 3-d space. *IEEE Trans. on Control Systems Technology*, 25, 1750–1760.
- Bowyer, S. A., Davies, B. L., & y Baena, F. R. (2013). Active constraints/virtual fixtures: A survey. *IEEE Transactions on Robotics*, 30(1), 138–157.
- Braglia, G., Calinon, S., & Biagiotti, L. (2024). A minimum-jerk approach to handle singularities in virtual fixtures. *IEEE Robotics and Automation Letters*, 9(11), 10256–10263.
- Braglia, G., Onfiani, D., Tebaldi, D., Lazzaretti, A., Biagiotti, L., et al. (2024). Spatial sampling for alignment of robot demonstrated trajectories in upper limb rehabilitation tasks. *CoRoboLearn Workshop*.
- Braglia, G., Tagliavini, M., Pini, F., & Biagiotti, L. (2023). Online motion planning for safe human-robot cooperation using b-splines and hidden markov models. *MDPI Robotics*, 12(4).
- Braglia, G., Tebaldi, D., & Biagiotti, L. (2025). Phase-independent dynamic movement primitives with applications to human-robot co-manipulation and time optimal planning. <https://arxiv.org/abs/2401.08238>

- Braglia, G., Tebaldi, D., Lazzaretti, A. E., & Biagiotti, L. (2024). Arc-length-based warping for robot skill synthesis from multiple demonstrations. *arXiv preprint arXiv:2410.13322*.
- Brunton, S. L., & Kutz, J. N. (2022). *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press.
- Cacace, J., Caccavale, R., Finzi, A., & Lippiello, V. (2019). Variable admittance control based on virtual fixtures for human-robot co-manipulation. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 1569–1574.
- Calinon, S. (2009). *Robot programming by demonstration*. EPFL Press.
- Calinon, S. (2020a). Gaussians on riemannian manifolds: Applications for robot learning and adaptive control. *IEEE Robotics & Automation Magazine*, 27(2), 33–45.
- Calinon, S. (2020b). Mixture models for the analysis, edition, and synthesis of continuous time series. *Mixture Models and Applications*, 39–57.
- Calinon, S. (2023a). Programming industrial robots from few demonstrations. IET.
- Calinon, S. (2023b). Robotics codes from scratch (RCFS).
- Calinon, S., D’halluin, F., Sauser, E. L., Caldwell, D. G., & Billard, A. G. (2010). Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine*, 17(2), 44–54.
- Calinon, S., Guenter, F., & Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2), 286–298.
- Calinon, S., & Lee, D. (2017). Learning control. In *Humanoid robotics: A reference* (pp. 1–52). Springer Netherlands.
- Cao, A., Chintamani, K. K., Pandya, A. K., & Ellis, R. D. (2009). Nasa tlx: Software for assessing subjective mental workload. *Behavior research methods*, 41, 113–117.
- Chalvatzaki, G., Papageorgiou, X. S., Maragos, P., & Tzafestas, C. S. (2019). Learn to adapt to human walking: A model-based reinforcement learning approach for a robotic assistant rollator. *IEEE Robotics and Automation Letters*, 4(4), 3774–3781.
- Chaquet, J. M., Carmona, E. J., & Fernández-Caballero, A. (2013). A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 117(6), 633–659.

- Chen, H., Xu, W., Guo, W., & Sheng, X. (2024). Variable admittance control using velocity-curvature patterns to enhance physical human-robot interaction. *IEEE Robot. Automat. Lett.*
- Chen, L., Lei, Y., Jin, S., Zhang, Y., & Zhang, L. (2024). Rlingua: Improving reinforcement learning sample efficiency in robotic manipulations with large language models. *IEEE Robotics and Automation Letters*.
- Chiaravalli, D., Califano, F., Biagiotti, L., De Gregorio, D., & Melchiorri, C. (2018). Physical-consistent behavior embodied in b-spline curves for robot path planning. *IFAC-PapersOnLine*, 51(22), 306–311.
- Chiu, C.-Y., Chao, S.-P., Wu, M.-Y., Yang, S.-N., & Lin, H.-C. (2004). Content-based retrieval for human motion data. *Journal of visual communication and image representation*, 15(3), 446–466.
- Cifuentes, J., Pham, M. T., Moreau, R., Prieto, F., & Boulanger, P. (2013). An arc-length warping algorithm for gesture recognition using quaternion representation. *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 6248–6251.
- Colgate, E., & Hogan, N. (1989). The interaction of robots with passive environments: Application to force feedback control. *Advanced Robotics: 1989: Proceedings of the 4th International Conference on Advanced Robotics Columbus, Ohio, June 13–15, 1989*, 465–474.
- Colledanchise, M., & Ogren, P. (2018). *Behavior trees in robotics and ai: An introduction*. CRC Press.
- Cuturi, M., & Blondel, M. (2017). Soft-dtw: A differentiable loss function for time-series. *International conference on machine learning*, 894–903.
- Dahlin, A., & Karayiannidis, Y. (2021). Temporal coupling of dynamical movement primitives for constrained velocities and accelerations. *IEEE Robotics and Automation Letters*, 6(2), 2233–2239.
- Dechter, R., & Mateescu, R. (2007). And/or search spaces for graphical models. *Artificial intelligence*, 171(2-3), 73–106.
- Deshpande, A. D. (2022). Novel biomedical technologies: Rehabilitation robotics. *Current Opinion in Biomedical Engineering*, 22.
- Devaney, R. L. (1976). Reversible diffeomorphisms and flows. *Transactions of the American Mathematical Society*, 218, 89–113.
- Dimeas, F., Fotiadis, F., Papageorgiou, D., Sidiropoulos, A., & Doulgeri, Z. (2019). Towards progressive automation of repetitive tasks through physical human-robot interaction. *Human Friendly Robotics: 10th International Workshop*, 151–163.

- Doha, E., Bhrawy, A., & Saker, M. (2011). On the derivatives of bernstein polynomials: An application for the solution of high even-order differential equations. *Boundary Value Problems*, 2011, 1–16.
- Donoso, F., Bustos-Salas, J., Torres-Torriti, M., & Guesalaga, A. (2008). Mobile robot localization using the hausdorff distance. *Robotica*, 26, 129–141.
- Dragan, A. D., Muelling, K., Bagnell, J. A., & Srinivasa, S. S. (2015). Movement primitives via optimization. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2339–2346.
- Du, G., Wang, K., Lian, S., & Zhao, K. (2021). Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review. *Artificial Intelligence Review*, 54(3), 1677–1734.
- Duan, X., Tian, H., Li, C., Han, Z., Cui, T., Shi, Q., Wen, H., & Wang, J. (2021). Virtual-fixture based drilling control for robot-assisted craniotomy: Learning from demonstration. *IEEE Robotics and Automation Letters*, 6(2), 2327–2334.
- Eppner, C., Mousavian, A., & Fox, D. (2021). Acronym: A large-scale grasp dataset based on simulation. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 6222–6227.
- Escarabajal, R. J., Pulloquina, J. L., Zamora-Ortiz, P., Valera, A., Mata, V., & Valles, M. (2023). Imitation learning-based system for the execution of self-paced robotic-assisted passive rehabilitation exercises. *IEEE Robotics and Automation Letters*.
- Fang, H.-S., Wang, C., Gou, M., & Lu, C. (2020). Graspnet-1billion: A large-scale benchmark for general object grasping. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11444–11453.
- Farouki, R. T. (2012). The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6), 379–419.
- Ferraguti, F., Preda, N., Bonfe, M., & Secchi, C. (2015). Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4223–4228.
- Ferraguti, F., Talignani Landi, C., Sabattini, L., Bonfe, M., Fantuzzi, C., & Secchi, C. (2019). A variable admittance control strategy for stable physical human–robot interaction. *The International Journal of Robotics Research*, 38, 747–765.

- Ficuciello, F., Falco, P., & Calinon, S. (2018). A brief survey on the role of dimensionality reduction in manipulation learning and control. *IEEE Robotics and Automation Letters*, 3(3), 2608–2615.
- Ficuciello, F., Romano, A., Villani, L., & Siciliano, B. (2014). Cartesian impedance control of redundant manipulators for human-robot co-manipulation. *2014 IEEE/RSJ international conference on intelligent robots and systems*, 2120–2125.
- Fisac, J. F., Bajcsy, A., Herbert, S. L., Fridovich-Keil, D., Wang, S., Tomlin, C. J., & Dragan, A. D. (2018). Probabilistically safe robot planning with confidence-based human predictions. *arXiv preprint*.
- Flacco, F., Kröger, T., De Luca, A., & Khatib, O. (2012). A depth space approach to human-robot collision avoidance. *2012 IEEE International Conference on Robotics and Automation*, 338–345.
- Gams, A., Petric, T., Do, M., Nemec, B., Morimoto, J., Asfour, T., & Ude, A. (2016). Adaptation and coaching of periodic motion primitives through physical and visual interaction. *Robotics and Autonomous Systems*, 75, 340–351.
- Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marin-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6), 2280–2292.
- Gaspar, T., Nemec, B., Morimoto, J., & Ude, A. (2018). Skill learning and action recognition by arc-length dynamic movement primitives. *Robotics and autonomous systems*, 100, 225–235.
- Gattringer, H., Mueller, A., Oberherber, M., & Kaserer, D. (2022). Time-optimal robotic manipulation on a predefined path of loosely placed objects: Modeling and experiment. *Mechatronics*, 84, 102753.
- Gaz, C., Cognetti, M., Oliva, A., Giordano, P. R., & De Luca, A. (2019). Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization. *IEEE Robotics and Automation Letters*, 4(4), 4147–4154.
- Ginesi, M., Meli, D., Roberti, A., Sansonetto, N., & Fiorini, P. (2021). Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions. *Journal of Intelligent & Robotic Systems*, 101, 1–20.
- Ginesi, M., Sansonetto, N., & Fiorini, P. (2021). Overcoming some drawbacks of dynamic movement primitives. *Robotics and Autonomous Systems*, 144, 103844.

- Giordano, P. R., Delamare, Q., & Franchi, A. (2018). Trajectory generation for minimum closed-loop state sensitivity. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 286–293.
- Girgin, H., Low, T., Xue, T., & Calinon, S. (2023). Projection-based first-order constrained optimization solver for robotics. *arXiv:2306.17611*.
- Gopinath, D., Jain, S., & Argall, B. D. (2016). Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE robotics and automation letters*, 2(1), 247–254.
- Han, D., Nie, H., Chen, J., & Chen, M. (2018). Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection. *Robotics and computer-integrated manufacturing*, 49, 98–104.
- Hart, S. (1988). Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human mental workload/Elsevier*.
- Hersch, M., Guenter, F., Calinon, S., & Billard, A. (2008). Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6), 1463–1467.
- Hogan, N., & Sternad, D. (2013). Dynamic primitives in the control of locomotion. *Frontiers in computational neuroscience*, 7, 71.
- Hovland, G. E., Sikka, P., & McCarragher, B. J. (1996). Skill acquisition from human demonstration using a hidden markov model. *Proceedings of IEEE international conference on robotics and automation*, 3, 2706–2711.
- Huang, Y., Rozo, L., Silverio, J., & Caldwell, D. G. (2019). Kernelized movement primitives. *The International Journal of Robotics Research*, 38(7), 833–852.
- Hudson, D., Wiltshire, T. J., & Atzmueller, M. (2023). Multisyncpy: A python package for assessing multivariate coordination dynamics. *Behavior Research Methods*, 55(2), 932–962.
- Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural computation*, 25(2), 328–373.
- Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, 2, 1398–1403.
- Iturrate, I., Sloth, C., Kramberger, A., Petersen, H. G., Ostergaard, E. H., & Savarimuthu, T. R. (2019). Towards reversible dynamic movement primi-

- tives. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5063–5070.
- Jankowski, J., Bruder-muller, L., Hawes, N., & Calinon, S. (2023). Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 10125–10131.
- Jha, A., & Chiddarwar, S. S. (2017). Robot programming by demonstration using teleoperation through imitation. *Industrial Robot: An International Journal*, 44(2), 142–154.
- Jia, Y.-B. (2008). Quaternions and rotations. *Com S*, 477(577), 15.
- Julier, S. J., & Uhlmann, J. K. (1997). New extension of the kalman filter to nonlinear systems. *Signal processing, sensor fusion, and target recognition VI*, 3068, 182–193.
- Jurafsky, D., & Martin, J. H. (n.d.). Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.
- Kanazawa, A., Kinugawa, J., & Kosuge, K. (2019). Adaptive motion planning for a collaborative robot based on prediction uncertainty to enhance human safety and work efficiency. *IEEE Transactions on Robotics*, 35(4), 817–832.
- Kastritsi, T., & Doulgeri, Z. (2022). A passive admittance controller to enforce remote center of motion and tool spatial constraints with application in hands-on surgical procedures. *Robotics and Autonomous Systems*, 152, 104073.
- Kavraki, L. E., & LaValle, S. M. (2016). Motion planning. In *Springer handbook of robotics* (pp. 139–162). Springer.
- Keemink, A. Q., van der Kooij, H., & Stienen, A. H. (2018). Admittance control for physical human–robot interaction. *The International Journal of Robotics Research*, 37(11), 1421–1444.
- Khalil, H. K. (2002). Nonlinear systems third edition. *Patience Hall*, 115.
- Khansari-Zadeh, S. M., & Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5), 943–957.
- Khansari-Zadeh, S. M., & Billard, A. (2014). Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6), 752–765.

- Khansari-Zadeh, S. M., & Billard, A. (2012). A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32, 433–454.
- Khansari-Zadeh, S. M., & Khatib, O. (2017). Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. *Autonomous Robots*, 41, 45–69.
- Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1), 90–98.
- Khatib, O., Jorda, M., Park, J., Sentis, L., & Chung, S.-Y. (2022). Constraint-consistent task-oriented whole-body robot formulation: Task, posture, constraints, multiple contacts, and balance. *The International Journal of Robotics Research*, 41, 1079–1098.
- Kiss, B., & Szadeczky-Kardoss, E. (2008). Time-scaling in the control of mechatronic systems. *New Developments in Robotics Automation and Control*, 411–426.
- Klemm, V., Morra, A., Salzmann, C., Tschopp, F., Bodie, K., Gulich, L., Kung, N., Mannhart, D., Pfister, C., Vierneisel, M., et al. (2019). Ascento: A two-wheeled jumping robot. *2019 International Conference on Robotics and Automation (ICRA)*, 7515–7521.
- Kohler, M., et al. (1997). *Using the kalman filter to track human interactive motion: Modelling and initialization of the kalman filter for translational motion*. Citeseer.
- Koutras, L., & Doulgeri, Z. (2020a). A correct formulation for the orientation dynamic movement primitives for robot control in the cartesian space. *Conference on robot learning*, 293–302.
- Koutras, L., & Doulgeri, Z. (2020b). A novel dmp formulation for global and frame independent spatial scaling in the task space. *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 727–732.
- Kramberger, A., Shahriari, E., Gams, A., Nemec, B., Ude, A., & Haddadin, S. (2018). Passivity based iterative learning of admittance-coupled dynamic movement primitives for interaction with changing environments. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6023–6028.
- Kulvicius, T., Ning, K., Tamosiunaite, M., & Worgotter, F. (2011). Joining movement sequences: Modified dynamic movement primitives for joining movement sequences. *2011 IEEE International Conference on Robotics and Automation*, 2275–2280.

- Lauretti, C., Cordella, F., Guglielmelli, E., & Zollo, L. (2017). Learning by demonstration for planning activities of daily living in rehabilitation and assistive robotics. *IEEE Robotics and Automation Letters*, 2(3), 1375–1382.
- Liu, H., Qu, D., Xu, F., Du, Z., Jia, K., Song, J., & Liu, M. (2022). Real-time and efficient collision avoidance planning approach for safe human-robot interaction. *Journal of Intelligent & Robotic Systems*, 105(4), 93.
- Liu, I.-C. A., Uppal, S., Sukhatme, G. S., Lim, J. J., Englert, P., & Lee, Y. (2022). Distilling motion planner augmented policies into visual control policies for robot manipulation. *Conference on Robot Learning*, 641–650.
- Luenberger, D. G., Ye, Y., et al. (1984). *Linear and nonlinear programming* (Vol. 2). Springer.
- Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall.
- Meattini, R., Bernardini, A., Palli, G., & Melchiorri, C. (2022). Seng-based minimally supervised regression using soft-dtw neural networks for robot hand grasping control. *IEEE Robot. Automat. Lett.*, 7(4), 10144–10151.
- Meirovitch, Y., Bennequin, D., & Flash, T. (2016). Geometrical invariance and smoothness maximization for task-space movement generation. *IEEE Trans. Robot.*, 32, 837–853.
- Merckaert, K., Convens, B., Wu, C.-j., Roncone, A., Nicotra, M. M., & Vanderborght, B. (2022). Real-time motion control of robotic manipulators for safe human-robot coexistence. *Robotics and Computer-Integrated Manufacturing*, 73, 102223.
- Meuter, M., Iurgel, U., Park, S.-B., & Kummert, A. (2008). The unscented kalman filter for pedestrian tracking from a moving host. *2008 IEEE Intelligent Vehicles Symposium*, 37–42.
- Mongillo, G., & Deneve, S. (2008). Online learning with hidden markov models. *Neural computation*, 20(7), 1706–1716.
- Muhlbauer, M., Hulin, T., Weber, B., Calinon, S., Stulp, F., Albu-Schaffer, A., & Silverio, J. (2024). A probabilistic approach to multi-modal adaptive virtual fixtures. *IEEE Robotics and Automation Letters*.
- Mukadam, M., Dong, J., Yan, X., Dellaert, F., & Boots, B. (2018). Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11), 1319–1340.
- Muller, M. (2007). Dynamic time warping. *Information retrieval for music and motion*, 69–84.

- Nemec, B., Gams, A., Denisa, M., & Ude, A. (2014). Human-robot cooperation through force adaptation using dynamic motion primitives and iterative learning. *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 1439–1444.
- Nemec, B., Gams, A., & Ude, A. (2013). Velocity adaptation for self-improvement of skills learned from user demonstrations. *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 423–428.
- Nemec, B., Likar, N., Gams, A., & Ude, A. (2018). Human robot cooperation with compliance adaptation along the motion trajectory. *Autonomous robots*, 42, 1023–1035.
- Nemec, B., Simonic, M., Petric, T., & Ude, A. (2019). Incremental policy refinement by recursive regression and kinesthetic guidance. *2019 19th International Conference on Advanced Robotics (ICAR)*, 344–349.
- Nemec, B., Ude, A., et al. (2016). Speed adaptation for self-improvement of skills learned from user demonstrations. *Robotica*, 34(12), 2806–2822.
- Nemec, B., Zlajpah, L., Slajpa, S., Piskur, J., & Ude, A. (2018). An efficient pbd framework for fast deployment of bi-manual assembly tasks. *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 166–173.
- Ogata, K., & Yang, Y. (2002). *Modern control engineering* (Vol. 5). Prentice hall India.
- Onfiani, D., Caramaschi, M., Biagiotti, L., & Pini, F. (2024). Optimizing design and control methods for using collaborative robots in upper-limb rehabilitation.
- Palleschi, A., Garabini, M., Caporale, D., & Pallottino, L. (2019). Time-optimal path tracking for jerk controlled robots. *IEEE Robot. Automat. Lett.*, 4, 3932–3939.
- Paraschos, A., Daniel, C., Peters, J. R., & Neumann, G. (2013). Probabilistic movement primitives. *Advances in neural information processing systems*, 26.
- Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. *2009 IEEE International Conference on Robotics and Automation*, 763–768.
- Pastor, P., Righetti, L., Kalakrishnan, M., & Schaal, S. (2011). Online movement adaptation based on previous sensor experiences. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 365–371.

- Pentland, A., & Liu, A. (1999). Modeling and prediction of human behavior. *Neural computation*, 11(1), 229–242.
- Pervez, A., Ali, A., Ryu, J.-H., & Lee, D. (2017). Novel learning from demonstration approach for repetitive teleoperation tasks. *2017 IEEE World Haptics Conference (WHC)*, 60–65.
- Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients [Robotics and Neuroscience]. *Neural Networks*, 21(4), 682–697.
- Petitjean, F., Ketterlin, A., & Gancarski, P. (2011). A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, 44(3), 678–693.
- Piazzzi, A., & Visioli, A. (2000). Global minimum-jerk trajectory planning of robot manipulators. *IEEE Trans. on industrial electronics*, 47, 140–149.
- Pignat, E., Silverio, J., & Calinon, S. (2022). Learning from demonstration using products of experts: Applications to manipulation and task prioritization. *The International Journal of Robotics Research*, 41(2), 163–188.
- Priess, M. C., Conway, R., Choi, J., Popovich, J. M., & Radcliffe, C. (2014). Solutions to the inverse lqr problem with application to biological systems analysis. *IEEE Trans. on Control Systems Technology*, 23, 770–777.
- Quigley, M., Gerkey, B., & Smart, W. D. (2015a). *Programming robots with ros: A practical introduction to the robot operating system.* ” O’Reilly Media, Inc.”
- Quigley, M., Gerkey, B., & Smart, W. D. (2015b). *Programming robots with ros: A practical introduction to the robot operating system.* ” O’Reilly Media, Inc.”
- Quinlan, S., & Khatib, O. (1993). Elastic bands: Connecting path planning and control. *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 802–807 vol.2.
- Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286.
- Raiola, G., Lamy, X., & Stulp, F. (2015). Co-manipulation with multiple probabilistic virtual guides. *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 7–13.
- Raiola, G., Restrepo, S. S., Chevalier, P., Rodriguez-Ayerbe, P., Lamy, X., Tliba, S., & Stulp, F. (2018). Co-manipulation with a library of virtual guiding fixtures. *Autonomous Robots*, 42, 1037–1051.

- Raiola, G., Rodriguez-Ayerbe, P., Lamy, X., Tliba, S., & Stulp, F. (2015). Parallel guiding virtual fixtures: Control and stability. *2015 IEEE international symposium on intelligent control (ISIC)*, 53–58.
- Restrepo, S. S., Raiola, G., Chevalier, P., Lamy, X., & Sidobre, D. (2017). Iterative virtual guides programming for human-robot comanipulation. *2017 IEEE International conference on advanced intelligent mechatronics (AIM)*, 219–226.
- Reynoso-Mora, P., Chen, W., & Tomizuka, M. (2013). On the time-optimal trajectory planning and control of robotic manipulators along predefined paths. *2013 American Control Conference*, 371–377.
- Richardson, M. J., Garcia, R. L., Frank, T. D., Gergor, M., & Marsh, K. L. (2012). Measuring group synchrony: A cluster-phase method for analyzing multivariate movement time-series. *Frontiers in physiology*, 3, 405.
- Rodriguez, S., Tang, X., Lien, J.-M., & Amato, N. M. (2006). An obstacle-based rapidly-exploring random tree. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 895–900.
- Rosenberg, L. B., et al. (1993). The use of virtual fixtures to enhance operator performance in time delayed teleoperation. *Journal of Dynamic Systems Control*, 49(1), 29–36.
- Rotithor, G., Salehi, I., Tunstel, E., & Dani, A. P. (2022). Stitching dynamic movement primitives and image-based visual servo control. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(5), 2583–2593.
- Roveda, L., Magni, M., Cantoni, M., Piga, D., & Bucca, G. (2021). Human-robot collaboration in sensorless assembly task learning enhanced by uncertainties adaptation via bayesian optimization. *Robotics and Autonomous Systems*, 136, 103711.
- Ruckert, E., & d’Avella, A. (2013). Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. *Frontiers in computational neuroscience*, 7, 138.
- Sakr, M., Freeman, M., Van der Loos, H. M., & Croft, E. (2020). Training human teacher to improve robot learning from demonstration: A pilot study on kinesthetic teaching. *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 800–806.
- Samant, R., Behera, L., & Pandey, G. (2016). Adaptive learning of dynamic movement primitives through demonstration. *2016 International Joint Conference on Neural Networks (IJCNN)*, 1068–1075.

- Sampei, M., & Furuta, K. (1986). On time scaling for nonlinear systems: Application to linearization. *IEEE Transactions on Automatic Control*, 31(5), 459–462.
- Savarimuthu, T. R., Buch, A. G., Schlette, C., Wantia, N., Rossmann, J., Martinez, D., Alenya, G., Torras, C., Ude, A., Nemec, B., et al. (2017). Teaching a robot the semantics of assembly tasks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5), 670–692.
- Saveriano, M., Abu-Dakka, F. J., Kramberger, A., & Peternel, L. (2023). Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13), 1133–1184.
- Saveriano, M., Abu-Dakka, F. J., & Kyrki, V. (2023). Learning stable robotic skills on riemannian manifolds. *Robotics and Autonomous Systems*, 169, 104510.
- Saveriano, M., Franzel, F., & Lee, D. (2019). Merging position and orientation motion primitives. *2019 International Conference on Robotics and Automation (ICRA)*, 7041–7047.
- Sayols, N., Sozzi, A., Piccinelli, N., Hernansanz, A., Casals, A., Bonfe, M., & Muradore, R. (2020). Global/local motion planning based on dynamic trajectory reconfiguration and dynamical systems for autonomous surgical robots. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 8483–8489.
- Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6), 233–242.
- Sciavicco, L., & Siciliano, B. (2012). *Modelling and control of robot manipulators*. Springer Science & Business Media.
- Selvaggio, M., Chen, F., Gao, B., Notomista, G., Trapani, F., & Caldwell, D. (2016). Vision based virtual fixture generation for teleoperated robotic manipulation. *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, 190–195.
- Selvaggio, M., Fontanelli, G. A., Ficuciello, F., Villani, L., & Siciliano, B. (2018). Passive virtual fixtures adaptation in minimally invasive robotic surgery. *IEEE Robot. Automat. Lett.*, 3, 3129–3136.
- Senin, P. (2008). Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23), 40.

- Shapira Weber, R. A., Eyal, M., Skafte, N., Shriki, O., & Freifeld, O. (2019). Diffeomorphic temporal alignment nets. *Advances in neural information processing systems*, 32.
- Sharkawy, A.-N. (2021). Minimum jerk trajectory generation for straight and curved movements: Mathematical analysis. *arxiv*.
- Si, W., Wang, N., & Yang, C. (2021). A review on manipulation skill acquisition through teleoperation-based learning from demonstration. *Cognitive Computation and Systems*, 3(1), 1–16.
- Sidiropoulos, A., & Doulgeri, Z. (2021). A reversible dynamic movement primitive formulation. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 3147–3153.
- Sidiropoulos, A., & Doulgeri, Z. (2022). Dynamic via-points and improved spatial generalization for online trajectory planning with dynamic movement primitives. *arXiv preprint arXiv:2212.13473*.
- Sidiropoulos, A., Papageorgiou, D., & Doulgeri, Z. (2023). A novel framework for generalizing dynamic movement primitives under kinematic constraints. *Autonomous Robots*, 47(1), 37–50.
- Simonic, M., Petric, T., Ude, A., & Nemec, B. (2021). Analysis of methods for incremental policy refinement by kinesthetic guidance. *Journal of Intelligent & Robotic Systems*, 102(1), 5.
- Sola, J. (2017). Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*.
- Stevens, R. H., & Galloway, T. L. (2014). Toward a quantitative description of the neurodynamic organizations of teams. *Social Neuroscience*, 9(2), 160–173.
- Su, H., Mariani, A., Ovrur, S. E., Menciassi, A., Ferrigno, G., & De Momi, E. (2021). Toward teaching by demonstration for robot-assisted minimally invasive surgery. *IEEE Trans. on Automation Science and Engineering*, 18(2), 484–494.
- Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Russwurm, M., Kolar, K., & Woods, E. (2020). Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118), 1–6.
- Ten Pas, A., Gualtieri, M., Saenko, K., & Platt, R. (2017). Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14), 1455–1473.

- Todorov, E., & Jordan, M. I. (1998). Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *Journal of Neurophysiology*, 80, 696–714.
- Toponogov, V. A. (2006). *Differential geometry of curves and surfaces*. Springer.
- Toussaint, M., & Goerick, C. (2010). A bayesian view on motor control and planning. In *From motor learning to interaction learning in robots* (pp. 227–252). Springer.
- Turro, N., & Khatib, O. (2001). Haptically augmented teleoperation. *Experimental Robotics VII*, 1–10.
- Ude, A., Nemec, B., Petric, T., & Morimoto, J. (2014). Orientation in cartesian space dynamic movement primitives. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2997–3004.
- Vasquez, D., Fraichard, T., & Laugier, C. (2009). Incremental learning of statistical motion patterns with growing hidden markov models. *IEEE Transactions on intelligent transportation systems*, 10(3), 403–416.
- Vayer, T., Tavenard, R., Chapel, L., Courty, N., Flamary, R., & Soullard, Y. (2020). Time series alignment with global invariances. *arXiv preprint arXiv:2002.03848*.
- Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J., & Diehl, M. (2009). Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10), 2318–2327.
- Walter, M., Psarrou, A., & Gong, S. (1999). Learning prior and observation augmented density models for behaviour recognition. *BMVC*, 1–10.
- Wang, X., Zhu, W., Saxon, M., Steyvers, M., & Wang, W. Y. (2024). Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. *Advances in Neural Information Processing Systems*, 36.
- Wang, Y. (2012). Gauss–newton method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4, 415–420.
- Wasiela, S., Giordano, P. R., Cortes, J., & Simeon, T. (2023). A sensitivity-aware motion planner (samp) to generate intrinsically-robust trajectories. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 12707–12713.
- Weitschat, R., & Aschemann, H. (2018). Safe and efficient human–robot collaboration part ii: Optimal generalized human-in-the-loop real-time motion generation. *IEEE Robotics and Automation Letters*, 3(4), 3781–3788.

- Weitschat, R., Dietrich, A., & Vogel, J. (2016). Online motion generation for mirroring human arm motion. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 4245–4250.
- White, R. E. (1984). Signal and noise estimation from seismic reflection data using spectral coherence methods. *Proceedings of the IEEE*, 72(10), 1340–1356.
- Wiest, J., Hoffken, M., Kressel, U., & Dietmayer, K. (2012). Probabilistic trajectory prediction with gaussian mixture models. *2012 IEEE Intelligent vehicles symposium*, 141–146.
- Wiggins, S., Wiggins, S., & Golubitsky, M. (2003). *Introduction to applied non-linear dynamical systems and chaos*. Springer.
- Yang, C., Sue, G. N., Li, Z., Yang, L., Shen, H., Chi, Y., Rai, A., Zeng, J., & Sreenath, K. (2022). Collaborative navigation and manipulation of a cable-towed load by multiple quadrupedal robots. *IEEE Robotics and Automation Letters*, 7(4), 10041–10048.
- Zanchettin, A. M., Ceriani, N. M., Rocco, P., Ding, H., & Matthias, B. (2015). Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 13(2), 882–893.
- Zeestraten, M. J., Havoutis, I., & Calinon, S. (2018). Programming by demonstration for shared control with an application in teleoperation. *IEEE Robotics and Automation Letters*, 3(3), 1848–1855.
- Zhang, L., Guo, S., & Sun, Q. (2020a). An assist-as-needed controller for passive, assistant, active, and resistive robot-aided rehabilitation training of the upper extremity. *Applied Sciences*, 11, 340.
- Zhang, L., Guo, S., & Sun, Q. (2020b). Development and assist-as-needed control of an end-effector upper limb rehabilitation robot. *Applied Sciences*, 10(19), 6684.
- Zhang, Y., Xue, T., Razmjoo, A., & Calinon, S. (2024). Logic learning from demonstrations for multi-step manipulation tasks in dynamic environments. *IEEE Robotics and Automation Letters*.
- Zhao, H., Chen, Y., Li, X., & Ding, H. (2023). Robotic peg-in-hole assembly based on reversible dynamic movement primitives and trajectory optimization. *Mechatronics*, 95, 103054.
- Zheng, J., Li, Z., Huang, L., Gao, Y., Wang, B., Peng, M., & Wang, Y. (2022). A gmm-dtw-based locomotion mode recognition method in lower limb exoskeleton. *IEEE Sensors Journal*, 22(20), 19556–19566.

- Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., & Srinivasa, S. S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *The International journal of robotics research*, 32(9-10), 1164–1193.