UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2025-028
Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 10 March 2025 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN SCIENCES DE L'INGÉNIEUR

by

Alireza BAREKATAIN

Born on 22 October 1996 in Esfahan (Iran (Islamic Republic of))

A Combined Machine Learning Approach for the Engineering of Flexible Assembly Processes Using Collaborative Robots

Dissertation Defence Committee:

Dr. Holger VOOS, Dissertation supervisor
Professor, Université du Luxembourg

Dr. Olivier BRULS
Professor, Université de Liège

Dr. Radu STATE, Chairman
Professor, Université du Luxembourg

Dr. Carol MARTINEZ LUNA
Research Scientist, Université du Luxembourg

Dr. Nico HOCHGESCHWENDER, Vice Chairman
Professor, Universität Bremen

Abstract

The manufacturing industry is witnessing a rapid transformation from mass production to mass customization, driven by increasing consumer demand for highly personalized products. Collaborative robots (cobots) play a key role in enabling this shift, as their flexibility supports the dynamic and varied tasks necessary for producing high-mix, low-volume batches. However, traditional robot programming methods are not suited for unstructured environments with frequent task variations. These approaches quickly become cumbersome, prone to errors, and demand specialized robotic expertise. In response, Learning from Demonstration (LfD) has emerged as a promising paradigm for teaching tasks to robots by having them observe human demonstrations. This not only addresses the need for explicit programming but also allows non-experts to program robots efficiently. Nevertheless, practical deployment of LfD in real industrial scenarios remains challenging, particularly when ensuring that customized robotics solutions still meet the high-performance standards associated with traditional mass production processes. This thesis addresses these challenges by (1) proposing a practical roadmap that guides both researchers and industry practitioners in transitioning from rigid, mass production–oriented robotic tasks to flexible, LfD-based mass customization workflows; (2) introducing a one-shot demonstration framework, DFL-TORO, which captures time-optimal and smooth trajectories from a single human demonstration; and (3) presenting a modular, standardized software framework integrating LfD methodologies in manufacturing systems. Through an in-depth case study and experimental validations, the thesis lays the foundation for bridging the gap between academic research in LfD and its real-world adoption in mass customization settings.

Index

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Motivation

The manufacturing industry is experiencing a significant shift from mass production to mass customization, driven by the increasing demand for personalized products. Traditionally, mass production has been characterized by the production of large quantities of standardized goods, maximizing efficiency and cost-effectiveness through highly structured assembly lines. However, this model has limited flexibility, making it challenging to accommodate product variations or customization. In contrast, mass customization addresses the growing consumer demand for personalized products by combining the efficiency of mass production with the flexibility of custom-made items, allowing manufacturers to produce tailored goods in high-volume small batches while still maintaining cost-effectiveness and efficiency [1, 2].

A key enabler of this shift is the advancement in robotic technologies, specifically collaborative robots, or cobots. Traditional industrial robots, designed for repetitive tasks in structured environments, lack the flexibility required for dynamic and varied production processes. In contrast, mass customization requires advanced robots equipped with vision technologies, AI-driven decision-making, and increased dexterity to adapt to diverse and changing products [3, 4]. Cobots enhance this transition by working alongside human operators, fostering flexible, agile production processes. These cobots improve safety and facilitate intuitive human-robot interaction, making them ideal for environments with varied

and dynamic tasks that mass customization demands [5].

Despite the promise of cobots, traditional robot programming methods present significant limitations in mass customization environments. Conventional programming, which involves writing specific coordinate-based scripts for robot tasks, is rigid and highly task-specific. While effective in structured settings, it is cumbersome and time-consuming when frequent modifications are required to accommodate new tasks or changes in production [6]. Cobots, designed to operate in shared, unstructured environments with humans, add complexity to programming. This complexity grows exponentially with the need for diverse customizations, affecting program efficiency and production cycle times. Automating complex tasks using cobots in advanced industrial operations presents numerous challenges, especially when the objects to be manipulated are not in fixed coordinates. Traditional methods mainly rely on instructing the robot to follow a sequence of absolute coordinates. A simple example is symbolically illustrated in Figure 1.1, where to pick an object from a fixed location and place it elsewhere, one must record coordinates A, B, C, and D. The operator manually records the desired coordinates and then creates instructions in the program. However, this approach does not apply when a perception system is present and object locations are not fixed. One cannot record a set of fixed coordinates and expect the robot to sequentially move through them when the target object is in a variant position. Variant coordinates make traditional programming unintuitive and prone to errors. Figure 1.2 shows a symbolic illustration of such a situation. When a perception system is present and object locations are not fixed, the program needs to offset the variant coordinates to calculate the rest of the coordinates required to move the robot through the task. Creating and modifying these hand-coded programs for different tasks is time-consuming and difficult to maintain, and leads to increased downtime and lower efficiency [7].

Moreover, in complex and unstructured industrial environments, ensuring collision-free motion is critical. Figure 1.3 shows an example where an obstacle lies in the path of a pre-planned motion. In such cases, more coordinates are required to avoid collisions, exponentially increasing the program's complexity as the workspace becomes more complicated.

To address the limitations of manual programming, motion planning and optimization

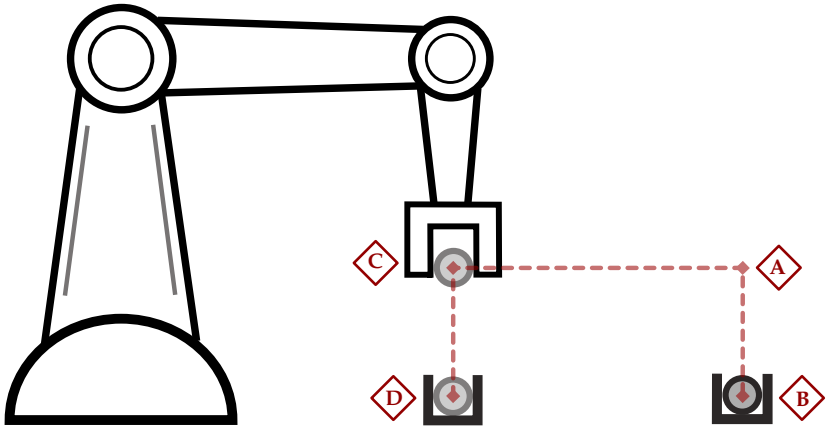


Figure 1.1: Traditional programming with fixed coordinates for pick-and-place tasks.

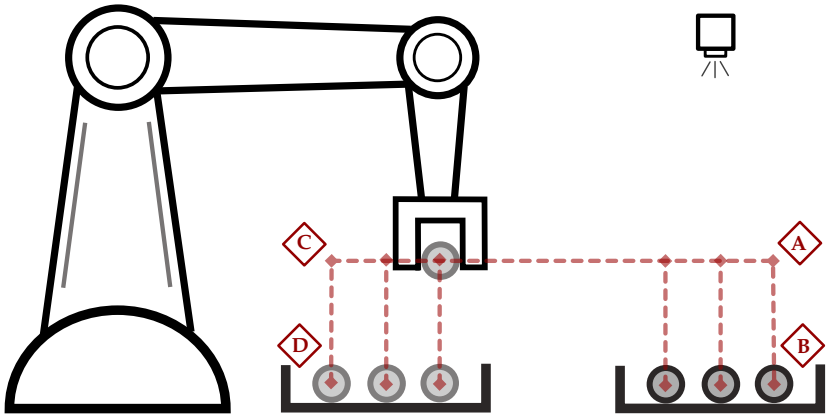


Figure 1.2: Offset calculations in traditional programming for handling variant object positions.

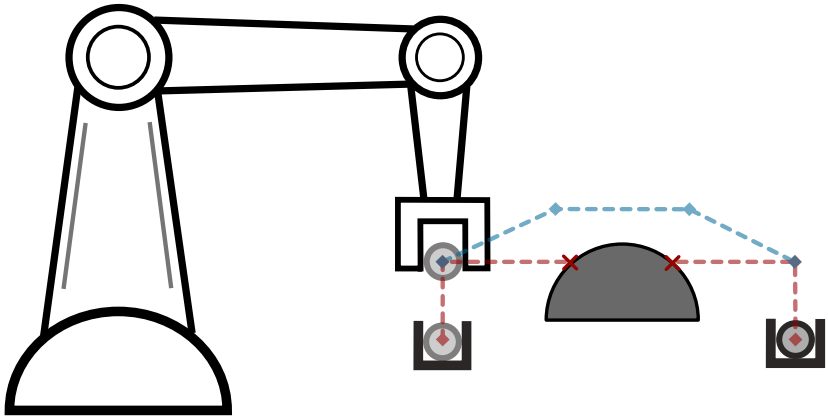


Figure 1.3: Collision avoidance in traditional programming through additional waypoints.

methods have been developed. These methods allow for higher-level programming by creating a robot model and using algorithms to optimize actions based on predefined costs and constraints, which effectively manages the complexity of tasks in dynamic and unstructured environments [8]. This approach improves operational efficiency but remains highly task-specific, requiring substantial reprogramming for different tasks or customization needs, which is inefficient for high-mix, low-volume production. Additionally, both manual and optimization-based programming methods demand significant robotic expertise, requiring an expert intermediary between the task specialist and the robot, thereby increasing implementation costs and impacting efficiency.

The most recent paradigm for flexible cobot programming is Learning from Demonstration (LfD), or Imitation Learning (IL), where robots learn tasks by observing and imitating human demonstrations [9, 10], *i.e.*, a robot learns to perform a task by watching a human's actions rather than being explicitly programmed. Unlike manual programming, LfD allows non-experts to teach complex behaviors without needing to provide explicit motion sequences. It also contrasts with motion planning and optimization, as it does not require modeling the robot's environment or defining explicit costs and constraints. Instead, LfD learns implicit task requirements from the demonstrations, achieving optimal behavior and efficient operation. This approach enhances implementation efficiency by enabling rapid and direct programming by non-expert users, eliminating the need for a robot expert to model environments or design specific optimizations for each task.

The advantages of LfD over traditional and optimization-based programming are particularly relevant in industrial settings, where the need for adaptability and agility in production processes is critical. By allowing robots to be programmed quickly and efficiently by non-expert users, LfD minimizes the need for robotic expertise, reduces programming time, and facilitates the seamless integration of robotic systems into modern manufacturing workflows. This makes LfD a crucial technology for industries aiming to leverage robotics in the era of mass customization [11, 12].

A symbolic example of LfD is shown in Figure 1.4. Here, the teacher demonstrates how to pick an object from a tray and place it on another tray (solid red line). This demonstration

contains several key pieces of conceptual information:

- **Core Task Concept:** It illustrates how to move any object from one tray to another, showing that the robot should approach the object perpendicularly to grasp it successfully.
- **Collision Avoidance:** The teacher naturally avoids obstacles during the demonstration, encoding environmental information that guides the robot to retreat higher and then descend to place the object, thus avoiding collisions.

By learning these concepts from the demonstration, the LfD approach can successfully plan and generalize the motion for other arbitrary object locations in the tray. The dashed lines in Figure 1.4 show examples of how the learned motion is applied to other objects while respecting task requirements and avoiding collisions.

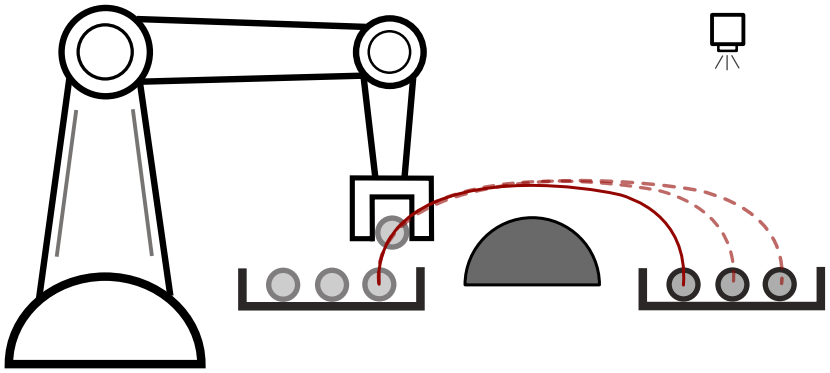


Figure 1.4: LfD example of learning and generalizing a pick-and-place task.

1.2 Problem Statement and Research Goals

While using LfD is shown to be conceptually advantageous in manufacturing settings with a mass customization theme, the implementation of LfD in a robotic manufacturing scenario poses several open research questions. The general problem this thesis addresses is the fundamental challenges faced when deploying LfD-based solutions to a real mass customization manufacturing scenario while ensuring efficiency with respect to the older

paradigm, mass production. This study encompasses several fundamental and practical contributions to the field, facilitating the transformation of robotic operations from mass production to mass customization. More precisely, the contributions of this thesis are as follows:

- A practical roadmap on transitioning robotic tasks from mass production to mass customization. Motivated by the paradigm shift from mass production to mass customization, it is crucial to have an easy-to-follow roadmap for industry practitioners with moderate expertise to transform existing robotic processes into customizable LfD-based solutions. To realize this transformation, this roadmap devises the key questions of "What to Demonstrate", "How to Demonstrate", "How to Learn", and "How to Refine". To follow through these questions, our comprehensive guide offers a questionnaire-style approach, highlighting key steps from problem definition to solution refinement. This roadmap equips both researchers and industry professionals with actionable insights to deploy LfD-based solutions effectively.

- DFL-TORO, A One-shot Demonstration Framework for Learning Time-Optimal Robotic Manufacturing Tasks. As the effectiveness of LfD is challenged by the quality and efficiency of human demonstrations, an approach is developed to intuitively capture task requirements from human teachers, by reducing the need for multiple demonstrations. Furthermore, this framework proposes an optimization-based smoothing algorithm that ensures time-optimal and jerk-regulated demonstration trajectories, while also adhering to the robot's kinematic constraints. The result is a significant reduction in noise, thereby boosting the robot's operation efficiency.

- A Software Framework for Standardized and Modular LfD in Industrial Manufacturing. Addressing the critical gaps in existing LfD software frameworks, this research introduces a comprehensive software solution that bridges the research and practical industrial deployment. By leveraging the Robot Operating System (ROS) [13] ecosystem and MoveIt [14], the framework provides a standardized, robot-agnostic architecture that supports modular integration of learning algorithms. Key innovations include seamless perception system integration, a high-level interface for non-expert users,

and a flexible approach to incorporating diverse LfD methodologies.

- An industrial case study to validate the proposed roadmap, DFL-TORO, and the developed LfD software framework. To highlight the feasibility and effectiveness of the developed methods and frameworks, a comprehensive industrial case study is conducted in collaboration with a manufacturing partner, where the findings of this thesis are put into action in a real manufacturing scenario.

1.3 Publications

The research conducted during this thesis has resulted in several peer-reviewed publications, contributing to the advancement of LfD for robotic manufacturing. These publications reflect the core theoretical developments, practical implementations, and case studies of this work. Below is a summary of the key publications:

1. Alireza Barekatain, et al. "A Practical Roadmap to Learning from Demonstration for Robotic Manipulators in Manufacturing" Published in *MDPI Robotics*, 2024. URL: mdpi.com/2218-6581/13/7/100.

2. Alireza Barekatain, et al. "DFL-TORO: A One-shot Demonstration Framework for Learning Time-Optimal Robotic Manufacturing Tasks" Published in *IEEE Access*, 2024. URL: ieeexplore.ieee.org/abstract/document/10735215.

3. Alireza Barekatain, et al. "Efficient Learning from Demonstration for Manufacturing Tasks" Presented in *IROS, Late Breaking Results*, 2023.

These publications collectively demonstrate the theoretical and practical contributions of this thesis to LfD in robotic manufacturing. Moreover, the open-source publication of the LfD software framework, including DFL-TORO, promotes accessibility and practical deployment of the thesis contributions. Software URL: https://snt-arg.github.io/dfl-toro/.

1.4 Outline

In Chapter 2 the research contributions are expanded along with their respective background and motivation, the state of the art, and the addressed gaps and challenges. In Chapter 3 the practical roadmap is presented, with a deep dive into the step-by-step questionnaire-style guideline on how to deploy LfD-based solutions. Chapter 4 introduces DFL-TORO, along with the detailed methodology and validation experiments. The LfD software framework is introduced and elaborated in Chapter 5. Finally, Chapter 6 is dedicated to the industrial case study, while the conclusion of the thesis is provided in Chapter 7 along with the possibilities and potentials of future directions.

Chapter 2

Background

This chapter outlines the core Research Contributions (RCs) presented in this thesis, emphasizing their role in addressing key challenges within modern manufacturing. Each contribution builds upon the critical need for adaptable and efficient robotic systems, driven by the industry's transition from mass production to mass customization. The proposed advancements target significant gaps in current practices, ranging from providing a practical roadmap for deploying LfD in industrial manipulators to introducing novel methodologies for improving operational efficiency, as well as developing a comprehensive software framework tailored for industrial applications. By offering structured, accessible, and actionable solutions, these contributions aim to bridge the gap between research and practice, enabling the seamless adoption of LfD in manufacturing. This chapter systematically presents the motivation, state-of-the-art context, and specific contributions of each RC, illustrating their potential to advance the field and address the pressing demands of contemporary manufacturing environments.

2.1 Overview of Research Contributions (RCs)

2.1.1 RC1: A Practical Roadmap

2.1.1.1 Background and Motivation

As elaborated in Chapter 1, recent advancements in the manufacturing industry have created an increasing need for adaptable robotic systems to perform flexibly with the variant demands of the market. The production schemes are shifting from mass production, where a fixed line of manufacturing is used to create products on a mass scale, to mass customization, where production is in smaller batches of different products according to the market need [1, 2]. To retain the efficiency and cost-effectiveness of mass production schemes, robotic systems have to quickly adapt to new tasks and manufacturing requirements [3, 4].

Industrial manipulators, as the backbone of automation in manufacturing, must evolve to meet these requirements. However, traditional programming methods—whether manual scripting or optimization-based approaches—are insufficient due to their rigidity, expertise requirements, and inefficiency in adapting to new tasks or changing environments [7, 8].

LfD emerges as a transformative approach, enabling robots to learn tasks through human demonstrations without requiring specialized robotic expertise. This makes LfD particularly suitable for industrial manipulators in manufacturing, where the ability to generalize tasks and quickly adapt to new production demands is crucial [11, 12]. Such transition and requirements have led to a significant application of LfD as a suitable solution in the manufacturing industry.

It means that the existing robotic tasks in the mass production scheme need to be transformed via LfD to meet the new requirements of mass customization, which is why it is necessary to provide guidance for industry practitioners to start deploying state-of-the-art LfD solutions in existing robotic tasks. Despite its promise, the implementation of LfD in industrial settings often lacks clear, practical guidance, making the transformation of existing robotic setups from mass production schemes to mass customization a pressing challenge. Addressing this gap is essential for leveraging LfD to its full potential and achieving the

flexibility required in modern manufacturing.

Notably, among all the robotic systems, industrial manipulators are the most popular and versatile robot types widely used in manufacturing and production lines. Therefore, while the application of robotic systems is not limited to manipulators, it is beneficial to have the focus of the roadmap narrowed down to industrial manipulators, due to their critical role in the automation of manufacturing and production lines.

2.1.1.2 State of the Art

Research on LfD has evolved significantly, with numerous studies addressing different aspects and applications. The work in [10] gives a general review of LfD and provides an updated taxonomy of the recent advances in the field. In [15] the authors survey LfD for robotic manipulation, discussing the main conceptual paradigms in the LfD process. In [16] the focus is on the applications of LfD in smart manufacturing. They introduce and compare the leading technologies developed to enhance the learning algorithms and discuss them in various industrial application cases. In [17] the focus of the survey is on robotic assembly. Specifically, they discuss various approaches to demonstrate assembly behaviors and how to extract features to enhance LfD performance in assembly. They also analyze and discuss various methods and metrics to evaluate LfD performance. Authors in [18] survey LfD advances with respect to human-robot collaborative tasks in manufacturing settings , dedicating their work to collaborative scenarios. They provide detailed insights into the role of various human interactions in different LfD methods. The technical nature of the survey makes it suitable for active researchers in LfD to seek new directions on making the LfD algorithms more collaborative. The authors in [19] conduct a thorough survey of all assembly tasks implemented by LfD, categorizing state-of-the-art LfD approaches in assembly and highlighting opportunities in academia. The work in [20] explores trajectory generation via LfD in technical depth, analyzing the performance of various learning methods. The authors in [21] provide an in-depth algorithmic perspective on learning approaches for robot manipulation, detailing learning methods, their representations, and limitations. The work in [22] focused on interactive learning, while [23] is dedicated to LfD for path planning, with a focus

on obstacle avoidance. The authors in [24] concentrate on LfD for contact tasks, offering detailed insights into this niche area. Besides the mentioned works, older reviews and surveys also exist such as [25, 26] which played a pivotal role in identifying main directions in LfD as well as providing insights from a general engineering perspective. However, with the rapid advancements of the field, newer works are required to encompass the most recent trends.

While the existing studies provide valuable insights into various aspects of LfD, they often lack a comprehensive roadmap or practical guidance for practitioners. They mostly focus on presenting the state of the art without providing a clear roadmap for practitioners to implement LfD in their robotic tasks. Moreover, the technical depth of most of the studies requires a strong background in LfD, making it inaccessible to non-academic practitioners or researchers who are new to the field. Therefore, there is a need for a new study that not only consolidates existing knowledge but also bridges the gap between research and practice.

2.1.1.3 Contribution

Motivated by the aforementioned considerations and in contrast to existing reviews, this study aims to present the state of the art via a practical and structured approach to implementing LfD in manufacturing tasks. Unlike the existing surveys and reviews, it takes the form of a comprehensive questionnaire-style guide, providing practitioners with a clear roadmap to integrate LfD-based robot manipulation. Tailored for moderate expertise requirements, this tutorial-style taxonomy offers step-by-step instructions, enabling both researchers and professionals to develop application-based LfD solutions. This novel perspective on the state of the art provides the readers with the main steps to define the problem and devise an LfD solution, as well as giving main research directions for refining the performance of the LfD.

The detailed contributions of the proposed roadmap are as follows:

- **Comprehensive Roadmap.** It offers a structured and practical roadmap for implementing LfD in diverse manufacturing tasks, guiding practitioners through key decision points and offering practical recommendations.

- **Tutorial-Style Taxonomy.** This roadmap includes a step-by-step tutorial designed for practitioners with moderate expertise, enabling the development of application-based LfD solutions.
- **Mapping of Existing Research.** It connects various elements from existing in-depth reviews and surveys, offering a high-level guide that helps practitioners identify specific challenges and access detailed insights for deeper understanding.

Remark: The application scenarios for robots in manufacturing are varied and unique, covering tasks like welding, painting, pick-and-place operations, machining, assembly, inspection, material handling, and packaging. Despite significant advancements in LfD for tasks such as peg insertion, gluing, interlocking, pick-and-place, sewing, wiring, stacking, screwing, hammering, and bolting, it remains impractical to comprehensively address all manufacturing tasks due to their diversity and the unique challenges they pose [19], especially in unstructured environments. Therefore, the proposed roadmap offers a comprehensive guideline to assist practitioners in navigating these challenges by identifying key decision points and providing practical implications, advantages, disadvantages, limitations, and recommendations. This roadmap is intended to be used iteratively to enhance the implementation of LfD.

2.1.2 RC2: DFL-TORO

2.1.2.1 Background and Motivation

With the adoption of cobots and LfD for mass customization, it is necessary to verify the performance of LfD methodologies when it comes to the efficiency of implementation and the efficiency of operation. Implementation efficiency refers to the effort, time, resources, and costs involved in transitioning a robotic task from concept to operational status, ensuring seamless integration of new tasks. Conversely, operation efficiency focuses on maximizing throughput and success rates during the robot's working phase, aiming to minimize operational costs like energy consumption and maintenance. Effective programming method-

ologies must balance these efficiencies to maintain high productivity and adaptability in the production process [16].

While LfD offers significant conceptual advantages over other programming methods, its deployment in practical manufacturing settings requires careful consideration and faces performance challenges in terms of implementation and operational efficiency. To maximize implementation efficiency, it is necessary to consider various factors such as the cost of demonstration setup, as well as the human time and cognitive effort required for demonstration. According to [10], kinesthetic teaching is particularly effective due to its minimal setup requirements and ease of demonstration to cobots. To further minimize human effort and time, focusing on one-shot demonstrations is crucial [20]. Among LfD approaches, Dynamic Movement Primitive (DMP) is a well-established method that can learn and generalize from a single demonstration. The choice of DMP in manufacturing is also further motivated by its high explainability and low implementation effort [9].

The efficiency of operation can be directly evaluated through task success rates and production throughput. While aiming for a 100% success rate and maximum throughput would theoretically maximize daily product output, achieving this ideal is nontrivial due to practical challenges. The noise in the recorded demonstration data, caused by joint sensor errors or hand tremors during demonstrations, negatively impacts operational efficiency in two significant aspects: accuracy and execution time.

Accuracy in LfD is determined by task-specific criteria that ensure the success of the task, similar to optimization-based approaches where constraints and costs guide an algorithm towards a successful solution. In human demonstrations, these constraints and costs are inherently encoded in the motion. For example, in a reaching task, the end-effector must approach the object from a specific direction to avoid collisions and grasp the object at the correct angle. These precise instructions are implicitly conveyed through human demonstration. However, noise can obscure these implicit cues, making them less effective. This degradation affects LfD algorithms' performance, making it challenging for the system to learn and generalize the intended motion, often leading to overfitting the noise and reducing task success rates [20].

Existing methods often rely on multiple demonstrations to regress noise-free demonstrations while maintaining accuracy [27, 28]. To numerically model the accuracy, they use the local variance of these demonstrations as a measure of significance for each segment: high local variability indicates low accuracy and vice versa. However, this approach compromises implementation efficiency by requiring the user to provide multiple demonstrations. To overcome this challenge, it is necessary to extract such accuracy measures while providing only one demonstration.

The execution time of robotic tasks is a crucial factor that directly influences production throughput. A lower execution time leads to a higher number of products produced per day and naturally a higher profit. Therefore, it is necessary to learn the fastest possible way to execute a task from human demonstration [19]. However, human demonstrations are naturally slow due to the cognitive complexity of teaching accurate, detailed motions [29]. Consequently, it is necessary to derive the optimal timing law from human demonstrations. Achieving optimal timing for robotic demonstrations is challenging due to two main issues. First, as detailed in [29], demonstrations cannot typically be sped up uniformly, hence a simple uniform scaling is not feasible. Second, while existing methods allow humans to locally speed up their demonstrations, they ignore the robot's kinematic limits. Due to the presence of noise, this approach leads to velocity and acceleration spikes that make the resulting trajectory kinematically infeasible. To avoid infeasibility, it is inevitable to settle for a non-optimal timing law. Additionally, this approach leads to a high-jerk motion. the motion jerk, indicating the rate of change of acceleration, affects motion smoothness. Low jerk profiles indicate smooth, human-like movement, which reduces mechanical stress, whereas high jerk increases wear, maintenance costs, and energy consumption [30]. Therefore, there is a need for a new approach to achieve the optimal execution time, while removing the demonstration noise and enhancing the smoothness of the motion by minimizing jerk.

2.1.2.2 State of the Art

To address the issue of slow demonstrations and execution time, several studies focused on the isolation of demonstrating path and timing, allowing humans to teach the path and

the timing law of demonstrations separately. Authors in [31] explore methods based on incremental learning to let the human teacher adjust the speed of their initial demonstrations, integrating human feedback into velocity scaling factors within the DMP formulation. The work in [32] further investigates the refinement of both path and speed through kinesthetic guidance, enabling teachers to fine-tune their demonstrations in real time, thereby enhancing the quality of the learned trajectories. Moreover, the study of [29] utilizes teleoperated feedback to locally speed up the task execution. Their work clearly signifies the importance of a well-demonstrated timing law in achieving high task success rates. To build upon these works, it is beneficial to incorporate the robot's kinematic limits into an optimization problem, solving for the best feasible timing law for the demonstrated path. Then, to allow the human teacher to refine the timing law, instead of merely speeding up, we can permit users to "slow down" demonstrations to achieve reliable execution, and hence ensure the optimal timing. Via this approach, optimizing for the timing law guarantees that the best achievable execution time is calculated, while the ability to slow down the demonstration provides direct control to the human teacher in order to make the task execution reliable and successful while maintaining fast performance.

In the context of LfD, there are limited studies addressing the regulation of motion jerk. In [30], authors tackled jerky demonstrations by introducing Gaussian noise to smoothen the trajectories. The study in [33] utilized the inherent smoothing properties of B-Splines by fitting demonstration trajectories with them, an approach also adopted by [34] and [35]. B-splines are advantageous for trajectory optimization due to their smoothness, local control, and efficient computation. Their piece-wise polynomial structure ensures smooth motion paths, making them ideal for complex motion optimization [36]. However, optimizing both timing and jerk simultaneously presents a challenge, as it can lead to an ill-defined problem. This difficulty arises because applying constraints over the demonstration path requires knowing the relative timing of each part of the path. Conversely, these timings cannot be treated as decision variables since the underlying trajectory representation in B-splines is a piece-wise polynomial [37]. As the original timing of the demonstration is suboptimal and cannot be used, it is possible to address this issue by employing a two-step solution: first,

optimizing the timing, and then regulating the jerk profile by removing noise.

When it comes to eliminating noise and unwanted behaviors in human demonstrations, the existing studies have taken various approaches to transform suboptimal demonstrations into optimal ones and enhance the learning and execution performance of LfD algorithms such as DMPs, Gaussian Processes (GPs), and Gaussian Mixture Models (GMMs). The performance of these methods relies heavily on the quality of the demonstrations. Suboptimal demonstrations can reduce the performance of LfD and cause overfitting or underfitting on the non-optimal demonstrations [20]. One approach is to focus on augmenting the demonstrations in order to fine-tune their data efficiency [38, 39], which aims at compensating the suboptimal demonstrations via various methods such as data augmentation or self-exploration. Another strategy focuses on extracting key information from demonstrations to capture essential implicit features [40]. to enhance the efficiency of the demonstration process, the work in [41] assessed the effectiveness of newly added demonstrations by quantifying their information gain. The authors in [42] optimized LfD outcomes via Reinforcement Learning (RL) by eliminating suboptimal demonstrations and retaining the useful ones. In [43], authors utilized GMM to increase the robustness of LfD to suboptimal training data, ensuring that the reproduced trajectory maintains the required precision.

A common theme in the mentioned works is the use of multiple demonstrations to optimize the demonstration process. This approach is driven by the absence of definitive ground truth in human demonstrations, making it challenging to distinguish between accurate demonstration data and noise or unwanted motion. The research question thus becomes how to develop a metric of accuracy that can serve as a ground truth while allowing some flexibility to modify or optimize the demonstration trajectories. Studies such as [27] and [28] highlight the use of task variability tolerances to adjust a robot's compliance. They suggest employing low impedance settings when high variability is detected, thereby accommodating more deviations in human demonstrations. The main idea is to align multiple demonstrations of a task and use the variance of each segment as a measure of significance or required accuracy. For example, a motion segment that is highly consistent across demonstrations indicates a need for precise replication, whereas segments with higher vari-

ability suggest permissible deviations while maintaining task accuracy.

Several studies have leveraged the statistical distribution of multiple demonstration trajectories, using variance to determine tolerance values that represent accuracy [44, 45, 46, 41]. Additionally, the study in [47] extracted tolerance bounds from multiple demonstrations to define obstacle-free regions. This approach provides flexibility to regress a noise-free trajectory within these bounds, ensuring the demonstration remains accurate, obstacle-free, and successful in achieving the task goal. Authors in [48] proposed a method where the human first moves the manipulator around the workspace to define a swept volume. This swept volume is then used as tolerance bounds to constrain the optimization method, while the optimization solves for the desired objective.

The goal of all of these studies is to develop tolerance bounds as numerical representations of accuracy, which can then be used to safely constrain the optimization methods and apply desired objectives to improve demonstrations. However, since all of the mentioned works rely on multiple demonstrations, they sacrifice the clear advantages of one-shot demonstration. Only a limited number of works have attempted to address this challenge using one-shot demonstrations. In [49], the authors used points recorded from a single demonstration to fit a piecewise polynomial trajectory within predefined and fixed tolerance bounds. Building on this, authors in [34] proposed a novel approach to adaptively extract tolerance bounds from one-shot kinesthetic demonstrations. They employed surface Electromyography (sEMG) signals, where the stiffness of the human teacher's hand, recorded through sEMG measurements during the demonstration, serves as a measure of accuracy or tolerance bounds. However, this method requires an expensive setup, increasing both the cost and effort of implementation. In manufacturing contexts, it is beneficial to capture these tolerance bounds from the human teacher while maintaining the simplicity of the demonstration setup and implementation efficiency.

2.1.2.3 Contribution

Given the outlined challenges, this study introduces DFL-TORO, a novel **D**emonstration **F**ramework for **L**earning **T**ime-**O**ptimal **R**obotic tasks via **O**ne-shot kinesthetic demonstra-

tion. DFL-TORO intuitively captures human demonstrations and obtains task tolerances, yielding smooth, jerk-regulated, time-optimal, and noise-free trajectories. DFL-TORO is introduced as a pivotal layer after capturing human demonstrations and before feeding the data to the LfD algorithm. For the first time, this issue is addressed by incorporating a new layer into the learning process. The detail of the contributions of DFL-TORO are as follows:

- An Optimization-based Smoothing algorithm, considering the robot's kinematic limits and task tolerances, which delivers time-and-jerk optimal trajectories and filters out the noise, enhancing operation efficiency. This work is the very first attempt to optimize the original demonstration trajectory with respect to time, noise, and jerk, before feeding to the learning algorithm.
- A method for intuitive refinement of velocities and acquisition of tolerances, reducing the need for repetitive demonstrations and boosting operation efficiency.
- Evaluation of DFL-TORO for a variety of tasks via a Franka Emika Research 3 (FR3) robot, highlighting its superiority over conventional kinesthetic demonstrations
- A case study via DMP, to showcase the benefits of incorporating DFL-TORO before training LfD algorithms.

2.1.3 RC3: A Software Framework

2.1.3.1 background and Motivation

The integration of LfD into industrial manufacturing requires a software framework that bridges the gap between research advancements and practical deployment. The manufacturing domain imposes unique requirements on software frameworks to ensure they are not only technically robust but also practical for industry practitioners with varying levels of robotic expertise. An ideal software framework must fulfill several key criteria:

1. **Standardization:** The software should adhere to a standard framework, such as ROS, which provides a widely accepted foundation for robotic software. ROS also offers

access to an extensive library of standardized robotic tools and packages tailored for industrial applications [13].

2. **Robot Agnosticism:** Given the diversity of robotic platforms in manufacturing, the framework must be platform-independent. It should generalize capabilities across robotic setups with minimal configuration. Leveraging tools like MoveIt, which provides a robot-agnostic interface, ensures seamless implementation of algorithms on various hardware platforms [14].

3. **Modular Learning Integration:** The rapid development of LfD methodologies and algorithms requires a framework that supports modularity, allowing for the seamless integration of new learning methods into the existing pipeline. This modularity is crucial for adapting to advancements in the field and extending the framework's functionality.

4. **Perception System Integration:** Flexible robotic manipulation relies on robust perception systems to localize objects for manipulation. The framework must enable the integration of perception modules, allowing LfD algorithms to utilize real-time object localization data for effective task execution.

5. **User Accessibility:** The software must cater to non-robotic experts, providing an intuitive, high-level interface. Industry practitioners should be able to design complex robotic tasks using learned subtasks without requiring extensive robotic expertise. High-level instructions must facilitate the assembly of arbitrary task workflows efficiently.

These requirements highlight the need for a robust and versatile software framework designed specifically for industrial applications of LfD.

2.1.3.2 State of the Art

Existing frameworks and tools for LfD address various aspects of the challenges in deploying LfD for industrial manufacturing but fall short of meeting all the critical requirements for practical implementation. For instance, Robomimic [50] is a modular framework that

supports learning multiple LfD algorithms from offline demonstrations. While it is valuable for research purposes, it lacks industry-standardization with ROS, does not support robot-agnostic implementation, and is not designed to execute and control physical robots directly, limiting its applicability in industrial contexts.

RVT2 [51] focuses on integrating visual perception into LfD but is narrowly scoped to perception tasks. It does not address other essential aspects, such as robot-agnostic implementation or the modular integration of diverse learning algorithms. Similarly, several works explore specific LfD algorithms. For example, [52], [53], and [54] delve deeply into particular methodologies, advancing algorithmic development. However, these approaches are limited in scope and do not generalize well across multiple LfD approaches or support robotic control in broader industrial applications.

Rofunc [55] is a Python-based framework that supports various LfD algorithms for robotic manipulation. Despite its versatility in algorithm support, it is not robot-agnostic, lacks standardization with ROS, and is not tailored for integration with industrial manufacturing workflows. On the other hand, Concept2Robot [56] excels in user accessibility, allowing non-experts to provide high-level instructions by translating language commands into manipulation concepts. However, it fails to meet other crucial requirements, such as industry-standard compliance, robot-agnostic implementation, and modular integration of advanced LfD algorithms.

While these existing solutions have made significant strides in their respective domains, they fail to offer a comprehensive, modular, and standardized framework tailored specifically for industrial manufacturing. Addressing these limitations is essential to bridge the gap between LfD research and its practical, scalable deployment in real-world industrial scenarios.

2.1.3.3 Contribution

To address the gaps identified in the state of the art, this research presents a comprehensive software framework for industrial LfD applications. The framework is designed to meet the diverse requirements of manufacturing by offering the following contributions:

- **Standardized and Modular Architecture:** Built on the ROS ecosystem, it ensures compatibility with industry standards and access to a wide range of robotic tools and packages. Moreover, the modular integration of LfD algorithms to accommodate existing methodologies and facilitate the seamless incorporation of new advancements.
- **Robot-Agnostic Implementation:** The software framework utilizes MoveIt as the control interface to ensure platform independence, enabling deployment across various robotic manipulators with minimal configuration effort.
- **Perception System Integration:** Incorporates perception modules for object localization and manipulation, providing a flexible interface for integrating vision-based systems into LfD workflows.
- **High-Level Accessibility for Non-Experts:** A high-level programming interface is designed for non-robotic experts, which enables them to compose LfD-based robotic tasks with a set of intuitive high-level instructions.

This framework bridges the gap between cutting-edge LfD research and practical industrial deployment. By focusing on standardization, modularity, and accessibility, it empowers both researchers and practitioners to harness the full potential of LfD in transforming manufacturing processes.

Chapter 3

A Practical Roadmap to Learning from Demonstration for Robotic Manipulators in Manufacturing

This chapter begins by presenting a practical roadmap for integrating LfD into industrial manufacturing tasks, addressing the increasing demand for flexible and adaptive robotic systems. As manufacturing shifts from mass production to mass customization, the ability of industrial manipulators to quickly adapt to new tasks has become essential. Traditional programming methods, with their rigidity and expertise requirements, fail to meet these demands. The roadmap bridges the gap between state-of-the-art LfD research and practical implementation, offering industry practitioners a structured, step-by-step guide to transforming existing robotic setups. By consolidating insights from the literature and providing actionable guidance, the roadmap empowers practitioners to navigate the complexities of deploying LfD solutions, ensuring efficient, scalable, and effective adoption in diverse manufacturing scenarios.

As depicted in Figure 3.1, the practitioner first addresses the question of “What to demonstrate” to define the “Scope of Demonstration”. Subsequently, the practitioner needs to answer the question of “How to Demonstrate” in order to devise a “Demonstration Mechanism”.

Accordingly, the question of “How to learn” equips the robotic manipulator with the proper “Learning mechanism”. Even though the LfD process implementation is accomplished at this stage, the evaluation of LfD performance leads to the question of “How to Refine”, which provides the research objectives and directions in which the performance of the LfD process can be further improved. Taking these points into account, the rest of the chapter is structured as follows:

1. **What to Demonstrate?** (Section 3.1): This section explores how to carefully define the task and identify certain features and characteristics that influence the design of the process.

2. **How to Demonstrate?** (Section 3.2): Building upon the scope of demonstration, this section explores effective demonstration methods, considering task characteristics and their impact on robot learning.

3. **How to Learn?** (Section 3.3): In this section, the focus shifts to implementing learning methods, enabling autonomous task execution based on human demonstrations.

4. **How to Refine?** (Section 3.4): Concluding the structured approach, this section addresses refining LfD processes to meet industrial manufacturing requirements, outlining challenges and strategies for enhancing LfD solutions in manufacturing settings.

3.1 What to Demonstrate

This section focuses on the first step in developing an LfD solution, *i.e.*, extracting the scope of the demonstration from the desired task. In this step, with a specific robotic task as the input, we explore how to determine the knowledge or skills a human teacher needs to demonstrate to the robot. The scope of demonstration establishes clear boundaries for what the robot should be able to accomplish after learning. Defining the scope is crucial because it sets the foundation for the entire LfD process. A well-defined scope ensures the provided demonstrations comprehensively and accurately capture the desired robot behavior.

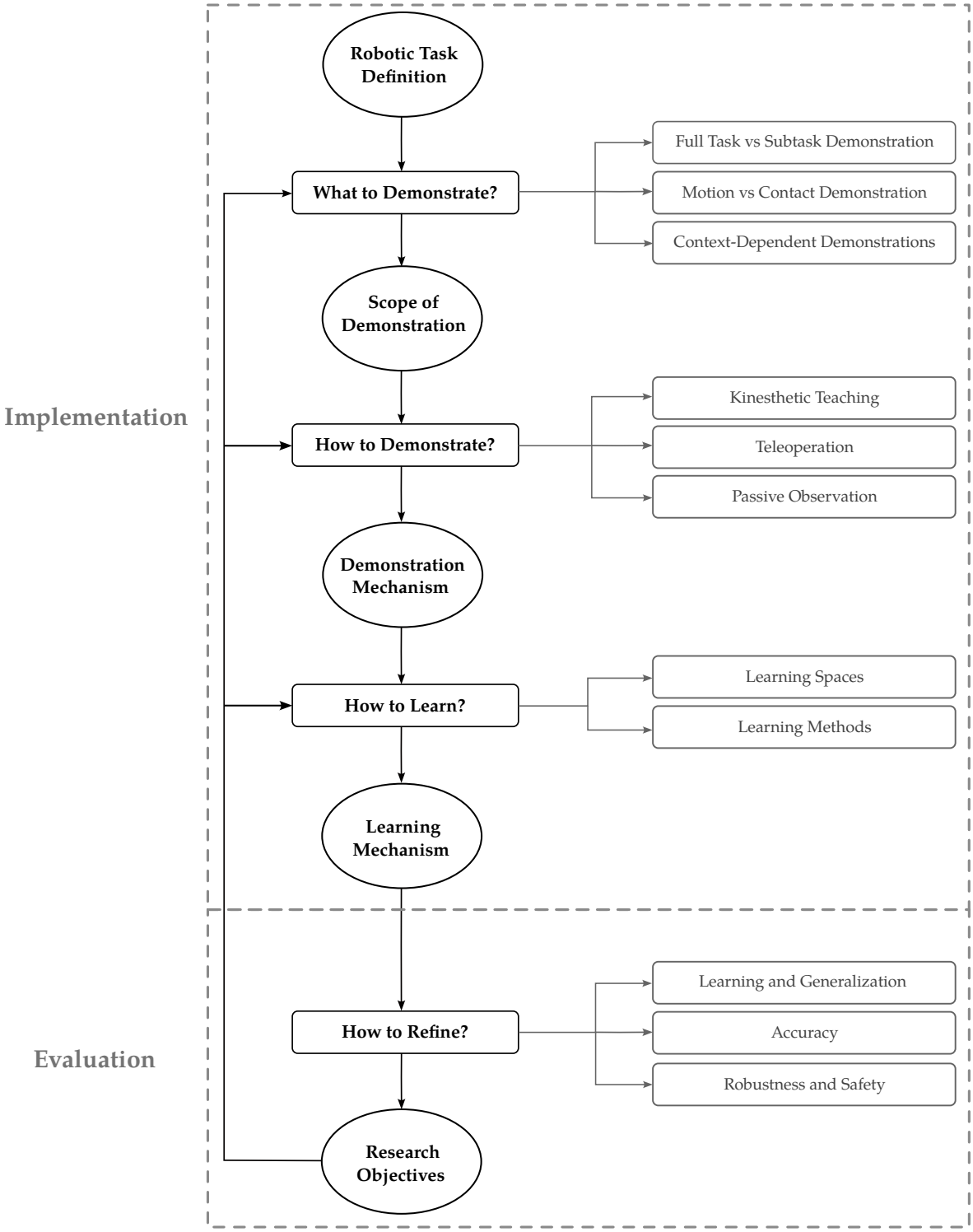


Figure 3.1: Overview of our proposed roadmap for LfD implementation.

Conversely, a poorly defined scope can lead to incomplete or inaccurate demonstrations, ultimately limiting the effectiveness of the LfD solution.

To determine “What to demonstrate”, three different aspects of the robotic task will be investigated as follows.

3.1.1 Full Task versus Subtask Demonstration

A full robotic task can be decomposed into smaller steps called subtasks, along with their associated task hierarchy or their logical sequence (Figure 3.2). In a full task demonstration, the human teacher demonstrates the entire process, including all subtasks in their logical order. In contrast, subtask demonstration focuses on teaching each subtask of the robotic task one at a time. Here we explore whether to demonstrate the full robotic task at once, or aim to demonstrate subtasks separately.

What Happens When Learning Full Task: In the case of full task demonstration, the LfD algorithm is required first to segment the task into smaller subtasks and learn them separately [57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]. Otherwise, training a single model on the entire task can lead to information loss and poor performance [68]. Beyond identifying subtasks, a full robotic task involves the logical order in which they should be executed – the task hierarchy. The LfD algorithm is required to extract these relationships between subtasks to build the overall task logic [57, 59, 60, 61, 65, 66]. This segmentation is achieved through spatial and temporal reasoning on demonstration data [58, 60, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73]. Spatial features help identify subtasks, while temporal features reveal the high-level structure and sequence.

For instance, consider demonstrating a pick-and-place task as a full task. The LfD algorithm can easily segment the demonstration into “reaching”, “moving”, and “placing” the object, along with their sequential task hierarchy. Because these subtasks involve clearly defined motions and follow a straightforward, linear order, the LfD algorithm can reliably extract the complete logic required to perform the entire pick-and-place task. On the other hand, consider a pick-and-insert task with tight tolerances. Precise insertion is challenging to demonstrate and requires retry attempts as recovery behavior. This creates a conditional

task hierarchy. The successful insertion depends on achieving tight tolerances, and if the initial attempt fails, the LfD system needs to learn the recovery behavior of repositioning the object and attempting insertion again. Consequently, LfD's reliance on automatic segmentation to extract the detailed task logic in such cases becomes less reliable. However, background information on the task characteristic can be provided as metadata by the human teacher and leveraged by the learning algorithm to improve the segmentation method in semantic terms [74, 63].

What Happens When Learning Subtask: When subtasks are demonstrated individually, the human teacher manually breaks down the full task and provides separate demonstrations for each one, along with the associated task hierarchy. This isolates the LfD algorithm's learning process, allowing it to focus on learning one specific subtask at a time [75, 76, 77, 78, 29, 79, 80]. It is evident that this approach requires extra effort from the teacher compared to full task demonstration.

For complex or intricate tasks such as pick-and-insert with tight tolerances, the teacher can individually provide "reaching", "moving", and "inserting" subtasks along with recovery behaviors. While this requires the teacher to separately define the task hierarchy and provide demonstrations for each subtask, it yields several benefits. First, the teacher can focus on providing clear and accurate demonstrations for each isolated step. Second, it allows for a reliable demonstration of the conditional hierarchy involved in complex tasks [81, 82].

When to Demonstrate Full Task versus Subtask: One of the significant differences between full-task and subtask demonstrations emerges in conditional behaviors, especially when recovery behaviors need to be demonstrated and implemented. For instance, in an assembly process with a stationary vision system, the vision system must detect a part for the robot to grasp. If the grasping fails, the robot should attempt the grasp again by first moving its arm outside the vision system's field of view, allowing the object to be re-detected before retrying. With full-task demonstration, it is impossible to showcase both the primary and recovery behaviors completely and in one pass. Therefore, subtask demonstration becomes necessary, either for the recovery behavior alone or for the entire task.

To further illustrate the difference, consider a welding task. Welding requires high pre-

cision and is dependent on the geometry of the object being welded. Making the learning method geometry-aware can enhance its generalization accuracy. In full-task demonstration, the practitioner needs to encode the entire geometry into the learning method and rely on the segmentation algorithm's efficiency to correctly segment the full demonstration and the geometric information. Conversely, subtask demonstration allows the practitioner to manually segment the full task based on their knowledge of the object's geometry. This approach provides better control over the welding performance, isolates the learning of subtasks, and facilitates the encoding of geometric information. In contrast, for tasks such as painting or stacking, full-task segmentation can be reliably achieved through spatial and temporal reasoning and can be left to the learning method. Manually dividing these tasks into subtasks would result in a large number of subtasks, making the demonstration process cumbersome and time-consuming.

Overall, demonstrating the entire robotic task at once can be efficient for teaching simple, sequential tasks, as the algorithm can segment and learn reliably. However, for complex tasks with conditional logic or unstructured environments, this approach struggles. Breaking the task into subtasks and demonstrating them individually is more effective in these cases. This removes the burden of segmentation from the learning algorithm and allows for better performance, especially when dealing with conditional situations. By teaching subtasks first, and then layering the task hierarchy on top, robots can handle more complex tasks and learn them more efficiently. In essence, full task demonstration is recommended for simple behaviors with linear task logic, while complex tasks are recommended to be decomposed into subtasks and demonstrated separately. As a general example, full task demonstration is recommended for tasks such as painting, material handling, stacking, and packaging, as the complexity of such tasks is limited and at the same time, division of the task into subtasks will result in numerous subtasks which increases the effort of task implementation. On the other hand, subtask demonstration is recommended for tasks such as welding, interlocking, screwing, and bolting, since such tasks are more precise and complex, and require more reliability to ensure high-quality demonstration. Nonetheless, these examples only serve as an intuitive guideline, and the actual choice depends on the detailed and various situations

of the desired task.

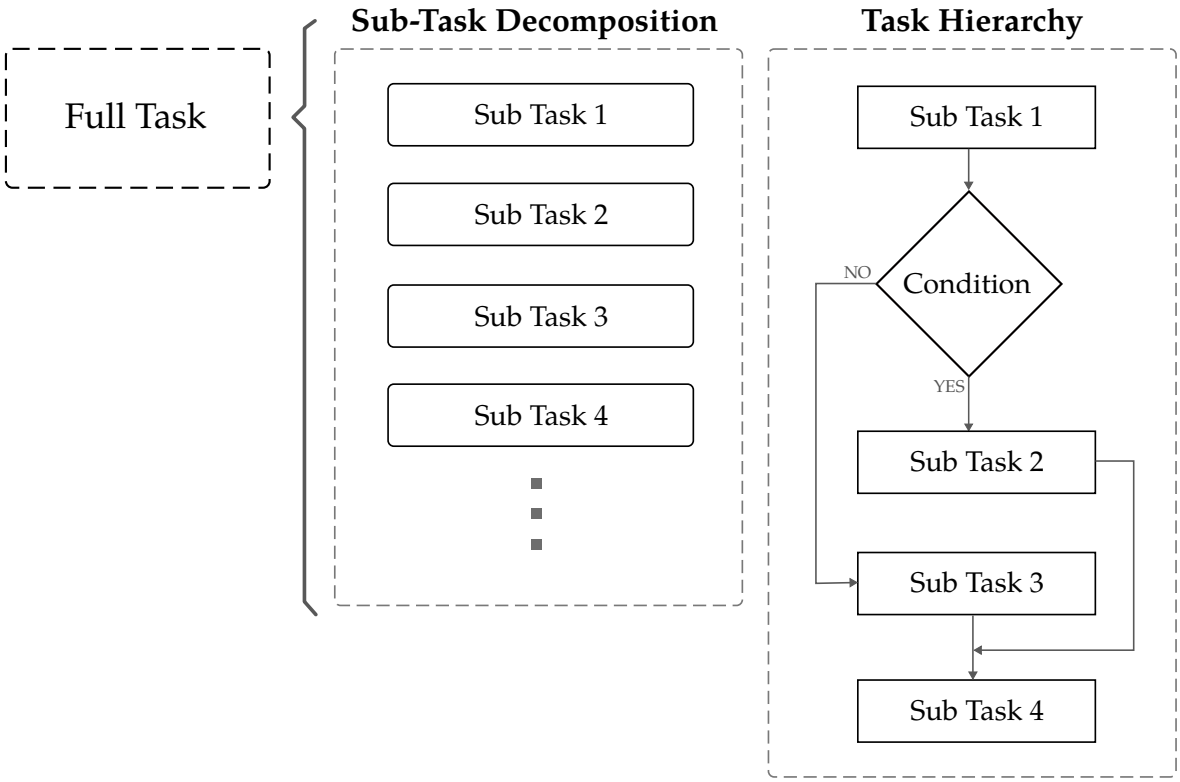


Figure 3.2: Illustration of how subtasks and task hierarchy comprise a full task.

3.1.2 Motion-Based versus Contact-Based Demonstration

Robot task demonstrations can be categorized into two main types: motion-based and contact-based. Motion-based tasks, like pick-and-place [29, 79, 80], focus on teaching the robot's movement patterns. The key information for success is captured in the robot's trajectory and kinematics, with limited and controlled contact with the environment [29, 79, 80, 83, 84, 85, 86, 87, 88, 89, 34]. Conversely, contact-based tasks, such as insertion [90, 91, 92, 32, 93, 94, 95, 96, 76, 97, 98, 46, 99], require the robot to understand how to interact with objects. Here, task success also relies on understanding forces and contact points [99]. Simply replicating the motion does not suffice and the robot needs to learn to apply appropriate force or adapt to tight tolerances to avoid failure. This highlights the importance

of contact-based demonstrations for tasks where interaction with the environment is crucial. The comparison of motion-based and contact-based tasks is illustrated in Figure 3.3. For deeper insights, the works in [23] and [24] dive deeper into motion-based and contact-based tasks, respectively.

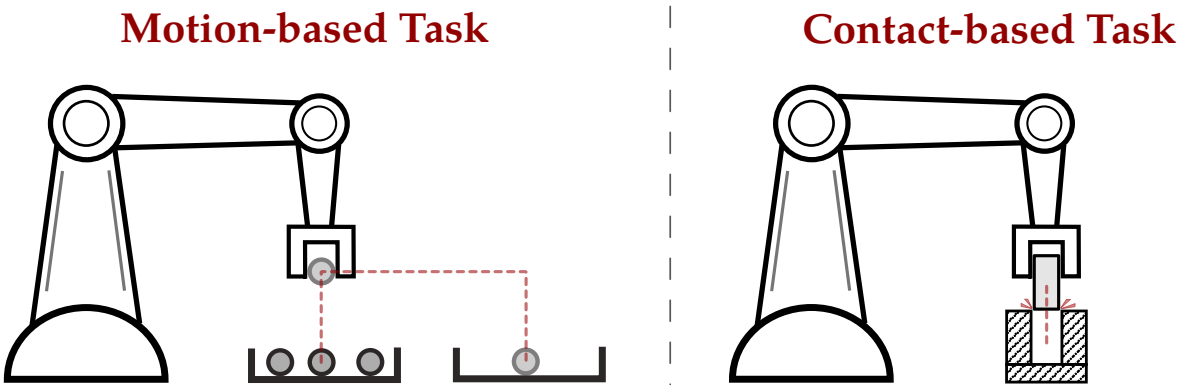


Figure 3.3: Comparison of motion-based versus contact-based task. On the left, the pick-and-place task has a structured and predictable interaction with the environment, while the insertion task on the right needs to deal with the contact resulting from tight tolerances to successfully perform the task.

Notion of Compliance: To understand whether a task is motion-based or contact-based, it is necessary to understand the notion of compliance. Compliance in robotics refers to the capacity of a robotic system to yield or adjust its movements in response to external forces, ensuring a more adaptable and versatile interaction with its environment [100]. This adaptability is typically achieved via Impedance Controllers, where the end effector of the robot is modeled as a spring-damper system to represent compliance (Figure 3.4) [101, 102]. In motion-based tasks, the robot prioritizes following a planned path with minimal adjustment (low compliance), *i.e.* external forces cannot alter the robot's behavior, while contact-based tasks allow for more adaptation (high compliance) to better interact with the environment. It is important not to confuse compliance with collision avoidance. Collision avoidance involves actively preventing contact with the environment by adapting the behavior on the kinematic level, while compliance relates to the robot's ability to adjust its behavior in response to external forces.

What is Learned and When to Teach: With motion-based demonstration, the LfD al-

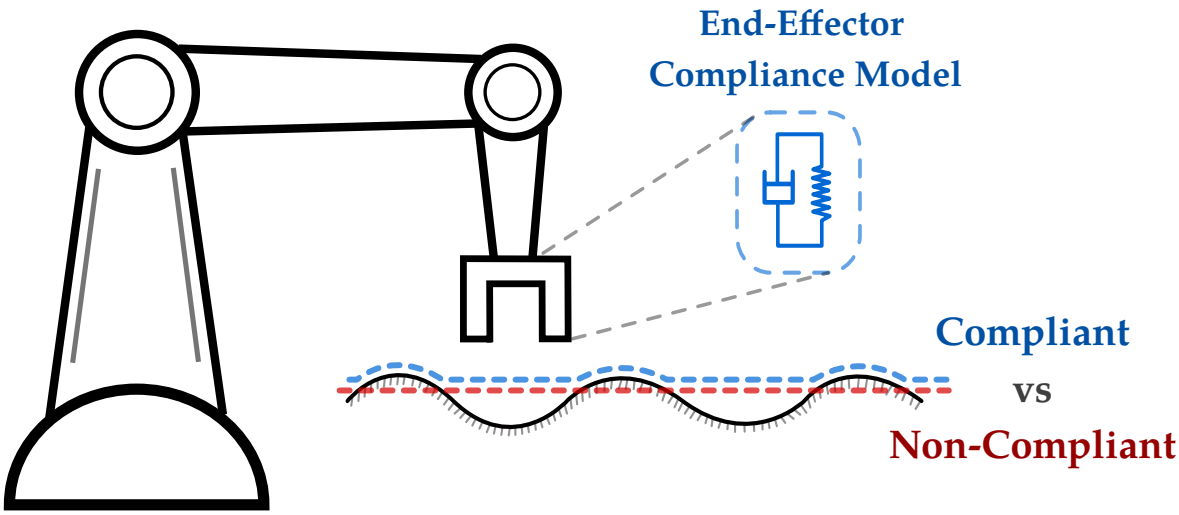


Figure 3.4: Illustration of compliant versus non-compliant behavior against the environment. The black line represents the environment surface, the red path represents a non-compliant behavior, and the blue path represents the compliant behavior against the environment surface. In Impedance Control, the end-effector is modeled as a spring-damper system.

gorithm learns under the assumption of zero compliance and replicates the behavior on a kinematic level, *i.e.*, strict motion. On the other hand, contact-based teaching enables the LfD algorithm to learn how to react when compliance is high, therefore it learns the skills on how to interact with the working environment.

One critical factor in selecting between motion-based and contact-based demonstration is the analysis of the desired task through the lens of its dependence on compliance. For example, in an insertion task with tight tolerances, if there are slight inaccuracies in measurements and the peg lands with a slight offset to the hole, compliance can be helpful to react skillfully to this misalignment and attempt to find the right insertion direction. Conversely, a pick-and-place task typically does not require compliance in the case when the grasping mechanism is simple and structured.

Practical Implications: From the point of view of practical implementation, the trade-off between motion and contact during the task execution is a key design factor to implement the task successfully [58, 76, 99, 46, 31, 44, 103]. For example, in tasks such as rolling dough [99], board wiping [76], and grinding [58], hybrid motion and force profile are learned as both

are crucial for successful task execution *i.e.* a force profile is needed to be tracked alongside the motion. As mentioned, this factor is best encoded via Impedance Control, where at each point of task execution, it can be determined whether position or force requirements are in priority [91, 104]. However, this method requires torque-controlled compliant robots and cannot be applied to many industrial robots which are position-controlled and stiff [32]. For such robots, the alternative to Impedance Control is Admittance Control, which operates with an external force sensor and can be implemented on stiff industrial robots [105].

As a general example, tasks such as inspection, welding, and packaging can rely mainly on motion demonstration, as environment interaction is either very limited or nonexistent. On the other hand, tasks such as peg insertion, interlocking, and bolting depend mainly on contact demonstration. Finally, the motion-contact trade-off plays a major role in tasks such as painting, screwing, and hammering, since both motion and contact demonstration should be respected to successfully complete the task. Again, the final design choice can vary depending on the requirements of the desired task and through iteratively finding the best scope of demonstration.

3.1.3 Context-Dependent Demonstrations

In addition to the choice of scope between full task versus subtask and motion versus contact demonstration, the operation of the robot is influenced by various specific contexts. Such contextual settings are highly dependent on the requirements of the task and often need to be custom-designed and tailored for the task. Nonetheless, here we discuss several common contexts alongside the considerations required for each.

3.1.3.1 Collaborative Tasks

Tasks that involve collaborating with humans or other robots typically provide interaction through an observation or interaction interface, which serves as a channel for information exchange between the robot and its collaborators [106, 107, 79, 108, 109, 110]. These interfaces can take various forms, such as physical interaction mechanisms or dedicated

communication protocols [109, 111, 112, 113], and are specifically designed and tailored to facilitate the collaborative task. Consequently, when providing demonstrations for such tasks, careful consideration must be given to ensure alignment with the interaction interface.

A typical example of a collaborative task is collaborative object transportation where one side of the object is held by the robot and the other part is held by the human [107]. In this case, the interaction interface is physical Human-Robot Interaction (pHRI), where not only the motion is important, but also the compliance becomes relevant. Another aspect to consider in collaborative tasks is safety and collision avoidance since the robot's operating environment is closely shared with the human collaborator [79]. It means that the human teacher needs to further teach safety strategies to the robot. Moreover, collaborative tasks have a more complicated task logic since the execution can depend on the collaborator's actions, which adds more conditions and more branching in the logic of the task. It also requires the robot to predict the intention of the human in order to follow the task hierarchy [111]. For a deeper insight into LfD for collaborative tasks, refer to [18].

3.1.3.2 Bi-Manual Tasks

Bi-manual tasks involve the coordinated use of both robot arms to manipulate objects or perform activities that require dual-handed dexterity [31, 83, 114, 115, 116, 70]. Prior research has explored strategies for optimizing synchronization, force distribution, and motion coordination in bi-manual learning, highlighting the importance of ensuring fluid and adaptive collaboration between the arms. Teaching a robot to perform bi-manual tasks through LfD should emphasize synchronization and coordination between the robot's multiple arms. For instance, in tasks like assembling components or handling complex objects, the robot needs to learn how to distribute the workload efficiently between its arms. Also, in dual-arm assembly, the coordination of both arms played a crucial role in achieving the desired precision [31, 117, 115]

Moreover, Bi-manual tasks often require specialized grasping strategies if both arms are simultaneously used for manipulating items [118, 116]. Given the proximity of both arms in bi-manual tasks, safety considerations become of great importance. The teaching

should emphasize safe practices, including collision avoidance strategies and safety-aware learning.

3.1.3.3 Via points

In certain contexts, teaching robot-specific via-points within a task can be a highly effective way to convey nuanced information and refine the robot's execution [97, 119, 120]. Via points serve as intermediate locations or configurations within a task trajectory, guiding the robot through critical phases or ensuring precise execution. One notable scenario where demonstrating via points can enhance the learning process is in assembly processes [121]. For complex machinery assembly, instructors can guide the robot through specific via points to ensure proper component alignment or correct part insertion. Additionally, via-points serve as an excellent measure of accuracy to optimize the robot motion and tool manipulation while ensuring successful task execution as long as the via point is passed throughout the path. Those enable the learning algorithm to understand optimal trajectories, adapt to changing conditions, and enhance its overall versatility.

3.1.3.4 Task Parameters

Some contexts require explicitly teaching a robot specific task parameters to enhance its understanding and performance in specialized scenarios. These task parameters go beyond the general actions and involve teaching the robot how to adapt to specific conditions or requirements. Here, the focus is on tailoring the robot's learning to handle variations in the environment, object properties, or operational constraints [109, 122, 123, 124, 120].

Teaching the robot about variations in object properties is essential for tasks where the characteristics of objects significantly impact the manipulation process. For instance, in material handling tasks, the robot needs to learn how to handle objects of different shapes, sizes, weights, and materials [125, 122]. Demonstrations can be designed to showcase the manipulation of diverse objects, allowing the robot to generalize its learning across a range of scenarios. Additionally, robots operating in manufacturing settings often encounter spe-

cific operational constraints that influence task execution. Teaching the robot about these constraints ensures that it can adapt its actions accordingly. Examples of operational constraints include limited workspace, restricted joint movements, or specific safety protocols [126].

3.2 How to Demonstrate

The next step after answering the question of "What to Demonstrate" and defining the scope of demonstration is to realize how to provide demonstrations to transfer the required knowledge and skills from the human teacher to the robot, *i.e.*, the channel through which the information intended by the teacher could be efficiently mapped to the LfD algorithm on the robot. According to [10], demonstration methods can be classified into three main categories: Kinesthetic Demonstration, Teleoperation, and Passive Observation. In this section, we discuss these categories and analyze their advantages and disadvantages with respect to the scope of demonstration, with a summary provided in Table 3.1.

3.2.1 Kinesthetic Teaching

Kinesthetic teaching is a method wherein a human guides a robot through a desired motion within the robot's configuration space. In this approach, a human physically guides the robot to perform a task, and the robot records the demonstration using its joint sensors (Figure 3.5(a)) [75, 29, 79, 91, 32, 96, 58, 127, 108, 98, 128, 129, 130]. This method has been widely studied for its effectiveness in intuitive robot programming, enabling robots to learn complex motions directly from human demonstrations. Research in this area has focused on improving the accuracy of motion capture, refining learning algorithms to generalize from demonstrations, and integrating multimodal feedback such as force sensing and electromyography to enhance teaching precision. The key aspect of kinesthetic teaching is the direct interaction between the human teacher and the robot in the robot's configuration space, which allows for precise guidance and facilitates natural, human-like motion learning.

What Setup is Required: Kinesthetic teaching offers a straightforward setup, requiring

only the robot itself. This simplicity contributes to ease of implementation and reduces the complexity of the teaching process, as well as minimizing the associated costs. This makes kinesthetic teaching an affordable and cost-effective option for training robots. Moreover, the interaction with the robot is intuitive for the teacher, making it easier to convey complex tasks and subtle details.

However, the suitability of kinesthetic teaching can be limited by the physical demands it requires, particularly with larger or heavier robots. Safety concerns also arise, especially in scenarios involving rapid movements or the handling of hazardous materials. This limitation can affect the scalability of kinesthetic demonstrations in diverse manufacturing contexts.

How Demonstration Data is Obtained: Through kinesthetic teaching, the robot records the demonstration using its joint sensors. The recorded data forms the basis for training the robot, allowing it to learn and replicate the demonstrated motion. The mapping of training data into the learning algorithm is straightforward, which enhances the reliability of the demonstration framework. However, the recorded demonstration data contain noise, as it depends on the physical interaction between the human and the robot. This noise can affect the smoothness of the training data and require additional processing to improve the learning algorithm's performance [131]. Additionally, since the training data depends on the robot hardware, the scalability of the training data to another setup will be limited.

Recommendations: Kinesthetic teaching is effective for instructing both full task hierarchies and low-level subtasks, especially excelling in demonstrating complex and detailed subtasks with its precise physical guidance of the robot. However, for full task demonstrations, additional post-processing of training data is advised to enhance segmentation and eliminate noise. While kinesthetic teaching offers precise control for motion demonstrations, the recorded data often suffers from noise and lacks smoothness due to the physical interaction between human and robot. However, it becomes limiting for contact-based demonstrations because unreliable torque readings from joint sensors prevent teaching the desired force profile to the robot, as the human guides its movements.

Table 3.1: Summary of the comparison of demonstration mechanisms.

	Kinesthetic Teaching	**Teleoperation**	**Passive Observation**
Concept	Physically guiding robot	Remotely guiding robot	Observing human actions
Advantages	Demonstrate Complex Motion Minimal Setup Intuitive Interaction Precise Manipulator Control	Safe Demonstration Isolation of Teaching	Safe Demonstration Ease of Demonstration
Limitations	Safety Concerns Physically Demanding	Complex Setup Requires Skills to Use	Complex Setup Inefficient for Complex tasks
Recommended Use	Full Task Demonstration Subtask Demonstration Motion Demonstration	Contact-Based Demonstration Iterative Refinement	Full Task Demonstration Large-Scale Data Collection

3.2.2 Teleoperation

Teleoperation refers to the process in which the human teacher remotely controls the movements and actions of the robot (Figure 3.5(b)). This control can be facilitated through different methods including joysticks, haptic interfaces, or other input devices, enabling the operator to teach the robot from a distance. Various studies have explored different aspects of teleoperation for robot learning, including methods for improving efficiency in learning from human demonstrations, adaptive control strategies, and the integration of vision-based feedback [76, 99, 132, 133, 134, 135, 136]. These works highlight the role of teleoperation in enhancing robot adaptability, leveraging human guidance more effectively, and incorporating multimodal feedback to refine task execution. Teleoperation differs from kinesthetic teaching as it allows for remote teaching to the robot.

What Setup is Required: Setting up teleoperation involves equipping the robotic manipulator with necessary sensors like cameras and force/torque sensors to relay feedback about the robot and its surroundings. On the human side, a well-designed control interface is required for intuitive and accurate control of the robot's movements. A robust communication system, whether wired or wireless, is necessary for real-time transmission of control signals and feedback. Safety measures, including emergency stop mechanisms, are essential to prevent unexpected behaviors, especially since the robot is operated remotely and immediate access to the robot is not possible in case of a malfunction.

The teleoperation setup is inherently well-suited for the tasks being operated in dangerous or hard-to-reach environments. Moreover, the design of the teleoperation interface can be versatile according to the target task, to provide the operator with a teaching interface closest to human-like dexterity. On the downside, teleoperation requires a more sophisticated setup compared to kinesthetic teaching, and it requires further designing the control and the communication interface according to the task. While it can promote intuitive teaching, it often requires extra training for the operator on how to use the setup for their teaching and demonstration. The remote nature of teleoperation can also raise concerns over the communication latency of the setup and how it affects the task demonstration depending on the task.

How Demonstration Data is Obtained: Through teleoperation, the acquisition of the training data can be flexibly designed based on what information is required for learning. The training data can be obtained by joint sensor readings, force/torque sensors on the joints or the end effector, haptic feedback, etc. It is the decision of the robotic expert how to map the raw teaching data to processed and annotated training data suitable for the LfD algorithm. One main advantage of teleoperation is that it is possible to isolate the teaching to a certain aspect of the task. For example, in [29], a joystick is used as the teleoperation device, where certain buttons only adjust the velocity of the robot, and other buttons directly affect the end-effector position and leave the execution velocity untouched. This benefit can enable better-tailored teaching or iterative feedback to increase the efficiency of the learning process.

Recommendations: Teleoperation is a suitable approach for demonstrating contact-rich tasks, since there is no physical interaction, and the joint torque sensors or the end effector force sensor on the robot can reliably record the contact-based demonstration as training data. It is also well-suited for tasks demanding real-time adjustments. One of the most common combinations of demonstration approaches is to use kinesthetic demonstration for motion demonstration, and use teleoperation for iterative refinements or providing contact-based demonstrations [53].

3.2.3 Passive Observation

Passive observation refers to the process of a robot learning by observing and analyzing the actions performed by a human or another source without direct interaction or explicit guidance, *i.e.* the teaching happens with the robot outside the loop while passively observing via various sensors (Figure 3.5(c)) [31, 137, 64, 138, 139, 140, 137, 115]. During passive observation, the robot captures and analyzes the relevant data, such as the movements, sequences, and patterns involved in a particular task. The features and characteristics of the task are extracted from the observation and fed into the LfD algorithm as training data for learning and generalization.

What Setup is Required: Setting up the teaching framework for passive observation involves various sensors such as 2D/3D cameras, motion capture systems, etc. to enable the LfD algorithm to observe the environment and the actions performed by the human teacher. The information from raw observation is then processed by sophisticated machine learning and computer vision algorithms to extract key features of the demonstration and track them throughout the teaching. In this setup, humans often teach the task in their own configuration space (*i.e.* with their own hands and arms), which makes the teaching easy and highly intuitive for the humans. However, the setup is complicated and expensive due to the requirement of various sensory systems.

When it comes to the scalability of demonstration across numerous tasks and transferability from one robotic platform to another, passive observation stands out as a suitable option for large-scale collections of demonstration datasets for various tasks, making it preferable when extensive datasets are needed for training purposes. This is while kinesthetic teaching and teleoperation mainly rely on a specific robotic platform and often cannot be scaled across tasks or robots.

How Demonstration Data is Obtained: Acquisition of training data through passive observation mainly relies on the extraction of the key features, as the learning performance critically depends on how well the features represent the desired behavior of the robot on the task. This forms a bottleneck in learning, since the more complex the task, the less efficient the feature extraction. Consequently, passive observation can suffer from learning and

performance issues when it comes to complex and detailed demonstrations. Overall, as the demonstration setup is complex for this approach and the correspondence problem limits the possibility of demonstrating complex tasks, this approach is not common for manufacturing use cases.

Recommendations Nonetheless, Passive observation has been used for demonstrating high-level full-task hierarchies. It is a suitable approach in scenarios where a diverse range of demonstrations across various tasks needs to be captured. For instance, in [31], human demonstrations are observed via a Kinect motion tracker, and then a motion segmentation is applied to build the overall task logic. In terms of scalability, passive observation stands out as a fit option for large-scale data collection, making it particularly suitable when extensive datasets are needed for training purposes.

3.2.4 Summary of Limitations, Efficiency, and Robustness

In addressing the limitations, efficiency, and robustness of the proposed demonstration methods for robot learning in manufacturing scenarios, several key insights emerge. Kinesthetic teaching, while straightforward and cost-effective, faces challenges due to its physical demands and safety concerns, particularly with larger robots or hazardous materials. The method's reliance on direct human-robot interaction introduces noise in demonstration data, requiring additional processing to ensure smooth learning and replication by the robot. Teleoperation offers flexibility in data acquisition and enhances safety in hazardous environments, yet it requires sophisticated setups and operator training, potentially introducing communication delays that affect task demonstration efficiency. Passive observation stands out for its scalability in collecting extensive datasets across various tasks but is restricted by complex setup requirements and limitations in extracting detailed features necessary for complex task demonstrations.

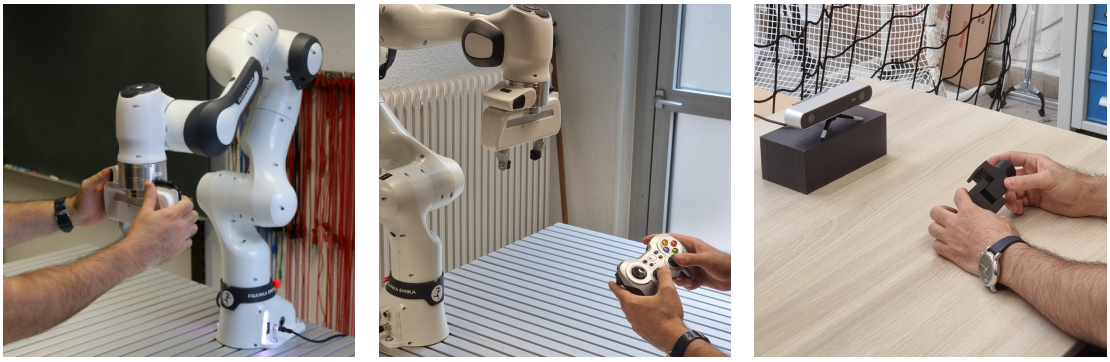
In terms of efficiency, kinesthetic teaching excels in its ease of implementation and intuitive interaction, enabling precise control and effective transfer of complex tasks. Teleoperation provides efficient teaching by isolating specific task aspects and supporting real-time adjustments, making it suitable for contact-rich tasks despite its setup complexities. Passive

observation proves efficient in large-scale data collection for training purposes but struggles with the efficient extraction of features crucial for complex task demonstrations. Robustness across these methods varies: kinesthetic teaching is robust for both hierarchical tasks and subtasks but constrained by physical limitations; teleoperation robustly handles contact tasks and real-time adjustments but faces setup complexities and latency issues; passive observation is robust in data scalability but limited by setup intricacies and feature extraction efficiency. These insights underscore the need for careful method selection based on specific manufacturing requirements to ensure optimal robot learning and deployment effectiveness.

To provide general examples, Kinesthetic teaching is recommended for tasks such as pick and place operations, inspection, and packaging. Kinesthetic teaching also proves effective for tasks requiring precise motion control and manipulation, such as welding. Teleoperation is recommended for tasks such as screwing, painting, and gluing. This is because teleoperation enables remote and safe demonstration for tasks such as gluing or painting, as well as allowing for haptic feedback to facilitate contact demonstration in tasks such as screwing and painting. Finally, passive observation, although less common in manufacturing due to its setup complexity, can be applied effectively in tasks like inspection and material handling. In material handling and packaging, passive observation enables robots to learn diverse grasping techniques and packaging sequences by analyzing human operators' actions from video feeds or motion capture data, enhancing versatility and scalability across different products and packaging formats. These examples serve as a guideline to better design the demonstration mechanism, while the final choice depends on the settings of individual manufacturing tasks in their own specific context.

3.2.5 Remarks

There are alternative approaches to provide demonstrations tailored to a specific context or situation. For example, in [141] the demonstration is in the form of comparison between trajectories, *i.e.* the robot produces a set of candidate trajectories, while the human provides preferences and comparison among them to imply which robot behavior is more suitable. In



(a) Kinesthetic Teaching (b) Teleoperation (c) Passive Observation

Figure 3.5: Illustrative examples of the main demonstration approaches.

[142], the human feedback has the form of a corrective binary signal in the action domain of the LfD algorithm during robot execution. The binary feedback signifies an increase or decrease in the current action magnitude.

3.3 How to Learn

This section focuses on the development of the LfD algorithm itself, after determining the scope of demonstration and the demonstration mechanism. The aim of this section is to consider how to design and develop a learning mechanism to meet the requirements of our desired task. We first discuss the possible learning spaces in which the robot can learn, and then we explore the most common learning methods used as the core of LfD algorithm.

3.3.1 Learning Spaces

Here we discuss the concept of learning spaces when representing demonstration data. The learning space not only encompasses where the training data from demonstrations are represented but also serves as the environment where the learning algorithm operates and generalizes the learned behavior. The choice of learning space is important as it provides background knowledge to the algorithm, thereby facilitating better learning and generalization within that designated space. While there are several possibilities for learning spaces,

here we focus on two main choices commonly used for robotic manipulators (Figure 3.6).

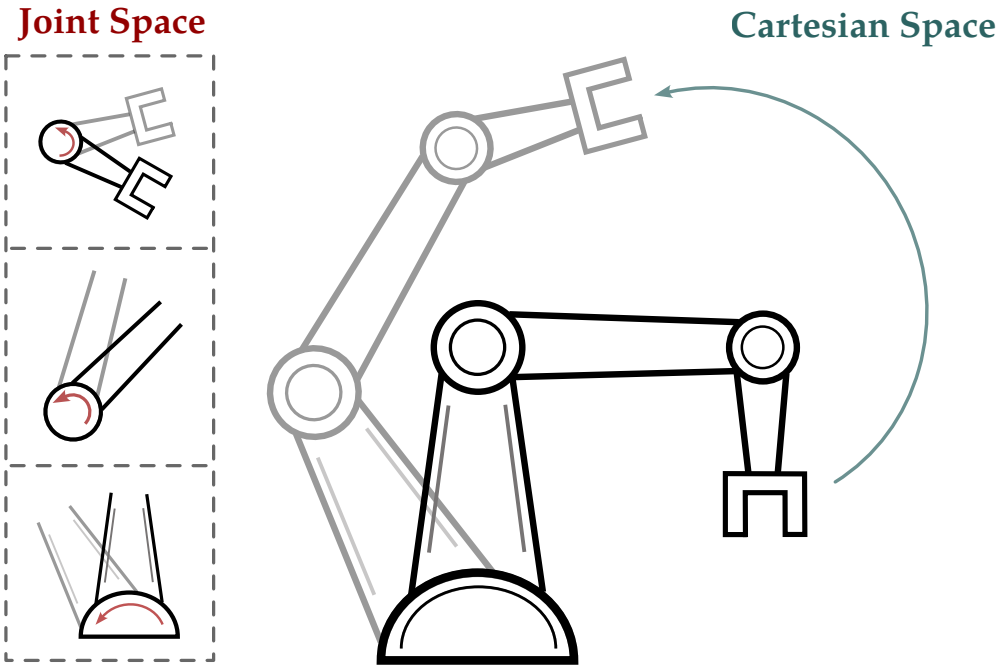


Figure 3.6: Illustrtive comparison of joint space and Cartesian space.

3.3.1.1 Joint Space

The joint space of a robot is a comprehensive representation of its individual joint configurations. This space serves as the primary language of motor commands which offers a low-level and precise representation of the possible configuration of each joint [75, 131, 143, 144].

Learning in Joint Space: Learning in joint space offers several advantages for LfD algorithms. Existing in Euclidean space makes the underlying math and data processing more efficient and straightforward. Additionally, since joint space directly corresponds to the robot's control layer, learned behaviors can be seamlessly integrated into the controller, which further simplifies the process.

However, a potential downside of learning in joint space is the risk of overfitting to specific demonstrations, which limits the robot's generalizability. Furthermore, focusing on joint

space learning neglects to capture higher-level contextual information which relate to the semantics of the task. Finally, joint space learning is sensitive to hardware variations, making it difficult when it comes to scalability and transferring skills between different robots.

Demonstrating in Joint Space: One notable advantage of demonstrating in joint space is the richness of the information obtained during the demonstration process, including the precise configuration of each individual joint. This level of detail is particularly advantageous for kinematically redundant manipulators, which can optimize null-space motion and obstacle avoidance.

However, a drawback lies in the intuitiveness of the learning process for human teachers. While joint space offers rich data for the robot to learn from, understanding how the learning algorithm learns and generalizes from this data is not intuitive for humans. The complexity involved in translating joint configurations into meaningful task representations can pose challenges for human teachers in understanding the learning process and predicting how the robot generalizes its learned skills.

In terms of acquiring demonstration data, kinesthetic teaching and teleoperation are straightforward methods for obtaining joint space data since joint states can be directly recorded. However, passive observation requires an additional step to convert raw information into joint space data, making it less practical to operate directly in joint space for this approach. In such cases, adding unnecessary transformations to obtain joint data is not justified, as it complicates the demonstration process.

3.3.1.2 Cartesian Space

This section introduces Cartesian space, a mathematical representation of three-dimensional physical space using orthogonal axes. In robotics, Cartesian space is commonly employed to describe the position and orientation of a robot's end-effector. LfD in Cartesian space involves training robots to imitate demonstrated tasks or movements by utilizing the coordinates of the end-effector [46, 29, 91, 92, 32, 93, 94, 77, 145, 44].

Learning in Cartesian Space: Cartesian space offers a natural representation for tasks involving end-effector movements and positioning, simplifying the learning process by di-

rectly addressing the space where tasks are being operated and executed. This choice is particularly advantageous for applications requiring precise end-effector control and potentially leads to better generalization in new situations. The consistency in representation allows for more effective generalization across different robotic systems and tasks.

One significant advantage of Cartesian space is the consistency of the dimension of the end-effector pose state across various robot platforms, regardless of their joint configurations. This standardized representation facilitates a more uniform approach to learning compared to joint space, which can vary in dimensionality from one robot to another. However, challenges arise when dealing with rotations within Cartesian coordinates, as rotation resides in a different manifold. It is required to utilize the calculus in a non-Euclidean space. This can increase the computational complexity of learning algorithms compared to those in the straightforward calculations of joint space.

Furthermore, the outcome of learning in Cartesian space cannot be directly integrated into the robot's controller. A transformation step is required to convert the learned information into joint space, adding an extra layer of complexity to ensure seamless integration into the robot's control system.

Demonstrating in Cartesian Space: Cartesian space is an intuitive space for the human teacher, which facilitates a better understanding of how the learning process happens. Additionally, this intuitiveness enables easier inclusion of task parameters and better iterative feedback to the LfD outcome, allowing the instructor to refine and optimize the robot's performance over successive teaching sessions.

However, a limitation arises in the encoding of end-effector behavior exclusively. In redundant manipulators, the representation in Cartesian space does not directly consider null space motion. The null space, which represents additional degrees of freedom beyond the end-effector behavior, is not explicitly encoded in Cartesian space. This limitation restricts the teacher's ability to convey and refine complex motions involving redundant manipulators, potentially overlooking certain aspects of the robot's capabilities.

Finally, acquiring training data in Cartesian space via kinesthetic teaching requires having the kinematic model of the robot including the end effector hand or tool, and applying

forward kinematic to the raw joint readings. The approach is similar in teleoperation, although it depends on the design of the teleoperation interface. Passive observation, however, demands the development of a mapping between the human hand or held tool and the robot's end effector. Leveraging pattern recognition and feature extraction techniques, the movements of the human hands are interpreted and translated into end-effector movements, serving as valuable Cartesian-space training data.

3.3.1.3 Remarks

While joint space and Cartesian space are the most common and fundamental spaces to represent a task, there are a variety of choices that can be particularly designed and developed for the designated task. For example, in [76], the pixel space of the camera was used as a measure of distance between the peg and hole. End-to-end training on visual images was used in [127]. Also, in [58], while the task is learned in Cartesian space, the Cartesian frame changes at each time step in a way that the z-axis always points towards the direction in which the force has to be applied.

When choosing approaches where a cost or reward function is involved, the design of such functions depends on the choice of latent/feature space, *i.e.* the space that the reward or cost function has to represent. In [141], the reward function is learned from demonstration, to represent as latent space of the training data. Later, the reward function is used by Reinforcement Learning to learn a suitable policy. Likewise, in [146], a cost function is learned via Inverse Optimal Control, which represents the task's requirement for successful execution.

3.3.2 Learning Methods

Here we provide a comparative analysis of the most common learning methods used for LfD. For each method, we discuss their learning concept and their training procedure, as well as their characteristics, strengths and weaknesses. Moreover, in Table 3.2, we present a summary of our comparative analysis. The table evaluates the learning methods across

several key metrics to provide insight into their respective practical strengths and weaknesses in the manufacturing context. The metrics assessed include the implementation effort, explainability, generalization capability, training data efficiency, and safety and robustness. Implementation effort helps evaluate the practicality and resource requirements of integrating a particular learning method into manufacturing processes. Explainability assesses how well the reasoning behind the learned behaviors can be easily understood and interpreted by operators. Generalization capability indicates the extent to which a learning method can adapt to new or unseen situations, enhancing its versatility and applicability. The efficiency of training data usage highlights how effectively a method can learn from limited datasets, optimizing resource utilization and reducing data acquisition costs. Finally, Safety and robustness metrics assess the reliability and resilience of learned behaviors in the face of uncertainties.

3.3.2.1 Movement Primitive (MP)

Learning Concept: A Movement Primitive (MP) encapsulates a low-level robot behavior defined by a trajectory generator and an exit condition. The trajectory generator specifies the desired robot motion, while the exit condition determines when the movement should stop [91, 92, 144, 147]. For instance, a typical MP involves moving a robot until contact with a surface, where the trajectory generator guides the robot until a predefined force threshold is reached, signaling the exit condition. MPs do not require demonstration at the subtask level. They are manually designed and optimized by robotic experts. Instead, LfD focuses on the task hierarchy, where the sequence in which to execute these pre-defined MPs is demonstrated to achieve a full task. Tasks composed of MPs can be structured using various methods such as state machines or task graphs. The essence of MPs is to offer a structured and optimized way to encode low-level robot behaviors that can be combined to achieve more complex tasks.

Training Procedure: The design, implementation, and optimization of MPs is performed by the robotic expert, while the human teacher demonstrates the task hierarchy composed of MPs. In this way, subtasks are structured and tailored to specific tasks, resulting in efficient,

reliable, and predictable behavior, while the demonstration effort is minimized. However, the manual design and tuning of MPs by robotic experts limit flexibility and control over lower-level behaviors for teachers. While MPs enhance learning performance by constraining the learning space to predefined combinations, their effectiveness depends on expert tuning, making them less adaptable to new behaviors, especially at the subtask level. MPs restricts generalization at the subtask level, demanding expert intervention to encode, tune, and integrate new behaviors into the MPs.

3.3.2.2 Dynamic Movement Primitive (DMP)

Learning Concept: The Dynamic Movement Primitive (DMP) combines a spring damper dynamical system, known as an attractor model, with a nonlinear function to achieve a desired goal configuration [148, 149]. The attractor model ensures convergence to the goal configuration, with its attractor point serving as the target goal. Without the nonlinear function, the model asymptotically converges to the goal. The role of the nonlinear function is to encode a certain behavior represented via the demonstration. By superposition of the nonlinear function and the attractor model, DMP replicates the demonstrated behavior. Essentially, the nonlinear function guides the attractor system towards the desired behavior, while eventually vanishing as the system reaches the goal [104, 150, 113, 32, 99, 58, 151, 152, 153, 154].

Training Procedure: To effectively train a DMP, the training data should primarily exhibit temporal dependence, with time progression as the independent variable (x-axis) and the dependent variable (y-axis) representing aspects such as position, force, or stiffness. Subtask learning involves collecting training data by creating trajectories from joint readings, force sensors, or stiffness profiles. For whole task sequences, teaching data must first be segmented into subtasks, with each subtask fitted with its own DMP. While DMPs are more suited for learning subtasks, approaches exist where multiple DMPs can be integrated into a single model to represent entire task sequences [155]. Motion-based and contact-based learning are feasible via DMPs, utilizing position trajectories [152, 32], force trajectories [156], or stiffness profiles [157, 158] as training data. Notably, DMPs offer the advantage

Table 3.2: Comparison of learning methods in manufacturing contexts.

Metric	MP	DMP	RL	GP	GMM	ProMP
Concept	Predefined deterministic behavior	Deterministic system with nonlinear forcing term	Interactive learning of reward and policy models	Probabilistic modeling of functions	Mixture of multiple Gaussians	Basis functions to model behavior
Implementation Effort	Moderate	Low	High	Moderate	Moderate	Moderate
Explainability	High	High	Low	High	High	Moderate
Generalization Capability	Low	Moderate	High	Moderate to High	Moderate to High	Moderate to High
Training Data Efficiency	High	High	Low	Moderate	Moderate	Low
Safety and Robustness	Moderate to High	Low	Moderate to High	Moderate	Moderate	Moderate

of requiring only a single demonstration to generate a training set and train the model, although multiple demonstrations can be utilized for a more comprehensive training set [156, 159]. Moreover, DMPs have been employed in context-dependent learning scenarios for collaborative tasks via points or task parameters [107, 160].

This training procedure involves representing time as a phase variable to make DMP systems autonomous from direct time dependence. Training data, along with their derivatives, are input into the DMP equation to generate target values for approximating the nonlinear term. Locally Weighted Regression (LWR) [161] is a typical choice for the nonlinear function approximator. The learned outcome is a nonlinear function mapping the phase variable to scalar values. The attractor model of DMP is designed to ensure critical damping, facilitating convergence to the goal without oscillation. The core idea of DMP lies in representing behavior as a deterministic system augmented by forcing terms, enabling learning at a higher level. While DMPs offer simplicity and reliability in implementation and generalization, their generalization mechanism is limited to a region around the original demonstration's configurations. Hyperparameters, although manually tuned, can be applied across various demonstrations without re-tuning. DMPs can be trained in joint space or Cartesian space, with multiple DMPs synchronized via the phase equation for joint space training, and a modified version for rotational values in Cartesian space represented using quaternions.

3.3.2.3 Reinforcement Learning (RL)

Learning Concept: Reinforcement Learning (RL) in the context of LfD involves training robots to execute tasks by interacting with their environment, receiving feedback in the form of rewards or penalties, and adjusting their actions to maximize cumulative rewards [162, 93, 94, 97, 141, 27, 145, 163, 164]. At its core, RL involves an agent (the robot) learning optimal actions to achieve a predefined objective within a given environment. This learning process hinges on the development of a policy, a strategy that guides the agent's actions based on the current state of the environment. The policy is refined through the optimization of a value function, which estimates the expected cumulative reward for specific actions in particular states. Rewards serve as feedback, reinforcing desirable actions and discouraging undesired behavior, thereby shaping the agent's learning trajectory. RL algorithms balance exploration and exploitation, enabling the robot to discover effective strategies while leveraging known successful actions. However, RL alone does not suffice for successful LfD without an accurate reward function encapsulating the task requirements.

Inverse Reinforcement Learning (IRL) acts as a bridge between RL and LfD. Via human demonstrations, IRL seeks the underlying implicit reward structure that guides those actions. The fundamental idea is to reverse engineer the decision-making process of the human teacher. Capturing the latent reward function enables the robot to replicate and generalize learned behavior to achieve similar goals in diverse contexts [165, 166, 167].

Training Procedure: To Train and LfD algorithm via RL-based methods, two key components are essential: a reward function and a policy function. The reward function encapsulates the task's definition, requirements, and success metrics, essentially encoding all the information provided by the teacher. Meanwhile, the policy function serves as the brain of the robot, dictating its behavior to execute the task. Training proceeds in two stages: first, designing or learning an appropriate reward function that accurately represents the desired task features and requirements, and second, training an RL algorithm to learn a policy function based on this reward function through interaction with the environment.

During the first stage, the focus lies on devising a reward function that encapsulates the teacher's instructions regarding the task. While this function can be manually designed, a

more comprehensive LfD solution involves learning the reward function from human demonstrations [93, 141, 96, 168]. The effectiveness of the LfD solution is directly linked to how well the reward function encapsulates the task's key aspects. In the second stage, assuming a reward function is already established, an RL algorithm learns a policy function by iteratively refining its behavior based on feedback from the environment and rewards obtained from the reward function.

Training via RL-based approaches offers flexibility in encoding information and learning skills, with training data ranging from raw images to robot trajectories. However, it requires careful engineering of the reward function and task analysis by robotic experts. Additionally, RL-based learning requires a dataset, not just a single demonstration, and involves modeling the environment, adding complexity to the implementation of LfD algorithms in this manner. Tuning hyperparameters associated with the policy and reward functions, as well as potentially modifying the environment model for each task, are the steps that require robotic experts to ensure successful and efficient learning.

3.3.2.4 Gaussian Process (GP)

Learning Concept: Gaussian Processs (GPs) are a probabilistic modeling approach in machine learning, capturing entire functions through mean and covariance functions known as kernel functions. The kernel function in GPs determines the similarity between function values at different points. This allows GPs to capture intricate patterns and relationships in data, while also estimating the uncertainty in those predictions. GPs are non-parametric, which means they are capable of learning from limited data points while being able to adjust their complexity with more data. Predictions from GPs include both anticipated function values and associated uncertainty, particularly useful in scenarios with sparse or noisy data [29, 77, 120, 119].

The conceptual advantage of learning methods like GP is their ability to quantify uncertainty, which is important for understanding the model's confidence in its predictions. This feature enhances human comprehension of the learning process and can serve as a safety mechanism for robots, where uncertain policies could lead to errors or damages. By moni-

toring the model's uncertainty and providing feedback, human teachers can refine the GP's behavior and adjust uncertainty bounds accordingly, ensuring safer and more robust execution.

Training Procedure: The training process for GP models involves learning from input-output pairs, where the independent variable can be any sequential variable, such as time or the position of the end effector. However, it is important to maintain the sequential nature of the data, *e.g.*, restricting scenarios where the end effector revisits a location. GPs excel at learning subtasks with limited demonstration data, even one-shot demonstration input. To improve the generalization capability, it is advisable to train GP in Cartesian space. This is because small changes in joint values can result in significant changes in the end effector in joint space. Moreover, training in joint space makes the uncertainty measures less interpretable.

GP is not particularly demanding in terms of implementation and hyperparameter tuning. The robot expert needs to design a kernel function in the formulation of GP, as well as very few hyperparameters. Moreover, GP is flexible across various tasks, *i.e.* it does not often require significant hyperparameter tuning or design alterations when transitioning from one task to another.

3.3.2.5 Gaussian Mixture Model (GMM)

Learning Concept: Gaussian Mixture Models (GMMs) offer a probabilistic method similar to GPs for modeling functions, representing them as a mixture of multiple Gaussian distributions. Each Gaussian component within a GMM represents a cluster in the dataset. In the context of LfD, GMMs can be used to model the underlying structure of human demonstrations. Due to their multi-modal nature, GMMs can capture complex behaviors while being flexible in terms of the number of learning variables [76, 141, 169]. They can also encode variability in demonstrations, giving a measure of accuracy at each point. similar to GPs. Additionally, GMMs can be updated locally with new data without affecting other segments of the learned behavior. This allows GMMs to naturally cluster data and learn complex behaviors from human demonstrations.

Training Procedure: GMMs allow for flexible training variable dimensions through multivariate Gaussian distributions, enabling each distribution to represent and train on different aspects of a task or subtask. Unlike GPs which can learn from single demonstration, GMMs require multiple demonstrations to capture the statistics and effectively learn the task. Additionally, GMMs are typically more intuitive when learned in Cartesian space rather than joint space, similar to GPs.

3.3.2.6 Probabilistic Movement Primitive (ProMP)

Learning Concept: Probabilistic Movement Primitive (ProMP) is a learning approach developed for LfD which employs Gaussian basis functions to model demonstrated behavior [170]. The parameters of ProMP are typically weights associated with the basis functions. ProMP introduces a probabilistic aspect into learning, similar to GP and GMM, allowing the model to accommodate uncertainty in learned movements and generalize them to different conditions or contexts [75, 79, 122, 171].

Training Procedure: The training procedure for ProMPs involves representing training data as time-series trajectories, with each trajectory corresponding to a specific demonstration of the task. These trajectories are often normalized or preprocessed to ensure consistency across different demonstrations. Through an optimization process, the model seeks to find the parameters that best fit the observed trajectories from demonstrations, constructing a probabilistic model capturing the distribution of trajectories and their likelihood at each point in time. However, ProMPs generally require a larger training dataset compared to other probability-based LfD methods.

3.3.2.7 Remarks

In addition to the mentioned methods, various other learning approaches serve as the core of LfD algorithm, with customization based on task-specific requirements. For example, Hidden Markov Models (HMMs) are commonly utilized for learning behaviors, as used in [46]. Additionally, methods based on optimal control are also employed, which are concep-

tually similar to RL. For example, in [146], Inverse Optimal Control (IOC) is used to learn the cost function of the task, similar to IRL. This function is later used to find an optimal policy to generalize the desired task. For deeper insights, the work in [21] provides technical analysis into learning methods used in LfD.

3.3.3 Summary of Limitations, Efficiency, and Robustness

In joint space learning, challenges such as the risk of overfitting to specific demonstrations, hardware sensitivity across different robot configurations, and the omission of higher-level contextual information underscore potential limitations. These factors can constrain the generalizability and adaptability of learned behaviors, impacting performance in diverse manufacturing tasks requiring contextual understanding. Conversely, Cartesian space learning introduces computational complexities due to non-Euclidean rotations and requires additional transformation steps for integration into robot controllers. Issues like null space motion constraints further affect the capabilities of redundant manipulators. Meanwhile, learning methods such as MPs, DMPs, and RL each present distinct challenges related to effectiveness, adaptability, and computational resource requirements. MPs and DMPs offer efficiency advantages with minimal training data but may lack flexibility for novel tasks, while RL demands extensive data and expert intervention for environment modeling and parameter tuning.

Efficiency considerations highlight methods like DMPs and GPs for their ability to optimize resource utilization and reduce data acquisition costs, crucial for practical deployment in manufacturing settings. However, the varying implementation efforts across these methods underscore the need for careful consideration of computational demands and expertise required for effective application. In terms of robustness and safety, probabilistic models like GPs and GMMs provide inherent uncertainty quantification, enhancing decision-making and safety in dynamic and uncertain manufacturing environments. These factors collectively emphasize the importance of addressing these limitations and efficiency concerns through ongoing research to enhance the applicability and reliability of these methods in future robotic manufacturing applications.

To provide general examples, tasks such as welding and painting benefit from learning

in Cartesian space, since their performance is dependent on the geometry of a surface, which is naturally encoded in Euclidean space. learning in Cartesian space can enhance the learning and generalization performance of such tasks. On the other hand, tasks such as pick and place operations, assembly behaviors, material handling, and packaging can efficiently learn and generalize in joint space as well, maintaining the performance while keeping the learning complexity minimal. The choice of learning method is highly dependent on the available resources and the specifics of the desired task. For example, if one-shot demonstration is a priority to minimize implementation effort for tasks such as pick and place or packaging, GPs and DMPs are recommended. If multiple demonstrations are available, probabilistic approaches can be utilized. For instance, the geometry of the surface can be incorporated as a task parameter to enhance the reliability of welding. Such requirements are numerous and the practitioners should consider their available context to implement the suitable learning mechanism.

3.4 How to Refine

The final step after completing the design and development of an LfD process, is to analyze and evaluate their performance, which guides the question of “How to Refine”. This section dives into the main key trends and directions within the state-of-the-art that aim to refine LfD algorithms across various aspects. For each trend, we will explore how it can improve LfD performance and identify potential areas for further research. The goal is to provide insights into possible research objectives for evaluation and improvement. This analysis will provide a refined perspective on the previously discussed aspects, ultimately contributing to an iterative loop of improvement for the entire LfD process.

3.4.1 Learning and Generalization Performance

Although current LfD approaches can learn from human teachings and generalize the behavior to new situations, it is still far from the idea of learning behavior that can be seen from humans. If human learning capabilities are considered the ideal case, LfD approaches are

not even close to cognitive learning capabilities. Therefore, it is a crucial line of refinement on the LfD approaches to get them closer to the cognitive learning power of human beings. The performance of learning and generalization refers to how well the algorithm can capture the essence of the desired behavior from human demonstration, and how intelligently the algorithm generates essentially the same behavior but adapts to the new environment or situation or context condition. Learning and generalization are two entangled factors that directly affect each other. A better learning performance subsequently leads to a better generalization, and improving generalization essentially means that learning performance has been improved. While using typical state-of-the-art LfD approaches already performs well in learning and generalization, they operate under assumptions, and actually cannot generalize to every possible case or situation. That is why it is necessary to improve the LfD approach based on what we require for our task and what is missing in the current LfD approaches. Improving learning performance means improving how the human demonstrations are processed by the learning algorithm, as well as modifying the core algorithm of learning to better capture different aspects of the behavior from the demonstrations. One trend is the approach of incremental learning [77, 88, 75, 32, 127]. Incremental learning refers to the ability of a robot to continuously acquire and refine its knowledge and skills over time as it interacts with its environment or receives additional demonstrations from a human operator. In [77], A GP is learned via one demonstration, but more demonstrations are provided through the operation of the robot to further refine GP training and improve its learned behavior. As another example, authors in [88] focused on continual learning for teaching letters in the alphabet incrementally without the algorithm forgetting the previously learned letters. The algorithm was able to write “Hello World” at the end, with the accumulated knowledge.

Building upon incremental learning, the concept of interactive learning emerges [22, 75, 122, 99]. Interactive learning refers to a learning paradigm in which the robot actively engages with the human teacher or the learning environment to acquire knowledge or skills. Unlike passive learning methods where information is simply presented to the learner, interactive learning involves two-way communication and dynamic interaction between the

learner and the learning material. In [142], authors provide an interactive learning framework through which non-expert human teachers can advise the learner in their state-action domain. In [75], the human teacher kinesthetically corrects the robot's trajectory during execution to teach the robot to perform the task accurately. A joint probability distribution of the trajectories and the task context is built from interactive human corrections. This distribution is updated over time, and it is used to generalize to the best possible trajectory given a new context.

Another technique is Active Querying [122, 77, 141]. In this technique, the learner dynamically decides which data points or demonstrations are most informative for the learning or decision-making process and requests the corresponding information from the teacher. This approach is particularly helpful for improving performance in an efficient way and acquiring the most relevant information. In [122], they focused on the demonstration distribution when training ProMPs, and the fact that it is not trivial how to add a good demonstration in terms of improving generalization capabilities. So they learn a GMM over the demonstrations distribution. and use epistemic uncertainty to quantify where a new demonstration query is required. Their proposed active learning method iteratively improves its generalization capabilities by querying useful and good demonstrations to maximize the information gain. In a similar way, the uncertainty measure of GPs is used in [77] to trigger a new demonstration request.

Aside from the mentioned learning paradigms, it is often required to modify learning algorithms based on the application use case or the context in which the LfD algorithm is employed. For example, there are several works to improve the performance of LfD with respect to contact-rich tasks [91, 92, 99, 128]. In [92], they proposed an approach to reduce the learning time of insertion tasks with tolerances of up to sub-millimeters, with application in manufacturing use cases. In [91], a novel task similarity metric is introduced, and it is used to generalize the already-learned insertion skills to novel insertion tasks without depending on domain expertise. In another context, [90] considers the assembly use cases and explores the idea that skillful assembly is best represented as dynamic sequences of manipulation primitives, and that such sequences can be automatically discovered by Rein-

forcement Learning. The authors in [125] extended DMPs to manipulating deformable objects such as ropes or thin films, to account for the uncertainty and variability from the model parameters of the deformable object. Another important aspect of learning is to learn factor which are in non-euclidean spaces. In [154], the formulation of DMP is modified to become geometry aware, so that the new formulation can be adapted to the geometric constraints of Riemannian manifold.

Finally, for certain contexts, some works attempt to design and develop control schemes to be learned in order to better learn and execute the behavior. The work in [58] has focused on the fact that some tasks such as grinding, sanding, polishing, or wiping require a hybrid control strategy in order to accurately follow the motion and the force profile required for successful task execution. To enhance the learning process of these tasks, they proposed some pre-programmed control strategies with parameters to tune via non-expert demonstrations. Such parameters are extracted from one-shot demonstrations and learn the motion-force task more efficiently.

Aside from the context, several works have attempted to provide improvements on the learning performance of a certain algorithm. Such algorithm-based improvements are dedicated to resolve the performance issues inherent in a learning method. In [153], they proposed a new formulation for DMPs in order to make it reversible in both directions and make it more generalizable. Authors in [32] introduce Constant Speed Cartesian DMPs, which completely decouples the spatial and temporal components of the task, contributing to a more efficient learning. In their LfD approach based on IOC, the work in [146] proposed an ensemble method that allows for more rapid learning of a powerful model by aggregating several simpler IOC models. The work in [77] combined GPs with DMPs, as GPs do not guarantee convergance to an arbitrary goal. Therefore, a DMP is learned on the GP output to ensure that any arbitrary goal is reached. With respect to Deep learning based approaches, [94] has focused on reducing the sample complexity by introducing self-supervised methods. For RL-based approaches, authors in [96] have developed a method to use demonstrations for improving learning sparse-reward problems. Both demonstrations and interactions are used to enhance the performance of learning.

3.4.2 Accuracy

While accuracy and precision can be mainly improved by improving the learning performance, it is not necessarily sufficient to achieve the desired accuracy by focusing generally only on the learning performance. While improving learning can enhance the overall generalization performance of the algorithm in the case of new scenarios, it is not a guarantee that the generalization outcome is accurate in terms of the desired task's metrics. Not only does the teaching process influence the accuracy of the outcome, but the execution strategy of the LfD is also the final stage that plays a major role in the outcome.

The notion of accuracy has a subjective nature, *i.e.* it can be defined differently from task to task. Therefore, there is no unified definition to describe the metric of accuracy across arbitrary tasks. However, several factors can be associated generally with the metrics of accuracy, to measure how accurate is the LfD algorithm, to some extent, with respect to the outcome. One of these factors is the success rate. Over various execution of the task via LfD policy, the success rate of the task to achieve a desired goal, can be a simple yet effective measure of how accurately a task is executed. If the accuracy requirements are not satisfied until a threshold, the task will not succeed at the end. Hence, success rate can be a measurable factor to describe the accuracy of a task, and a tangible metric to work towards improving by enhancing accuracy.

Besides focusing on the learning algorithm's performance, there are more dedicated approaches and trends to improve the accuracy of an LfD algorithm. One direction is to focus on the question of how to modify the teaching and demonstration method to enable human teachers to teach the task more accurately [98, 107, 31, 75, 29]. For example, In [107], they separated the demonstration of the shape of the trajectory from the timing of the trajectory. While it is cognitively demanding to demonstrate a motion with high accuracy and a high velocity at the same time, the ability to independently demonstrate the path enables the teachers to solely focus on teaching and refining the robot path and define the behavior in a more precise way. Moreover, in [98], they have combined incremental learning with variable stiffness of the robot during kinesthetic feedback. They used the variability of the already-captured demonstrations to adjust the stiffness of the robot. When there is

low variation in a region, *i.e.* higher accuracy, the robot is more stiff, allowing the teacher to provide smaller adjustments, while regions with higher variability have lower stiffness, allowing the teacher to move the robot more freely.

Another approach is to focus on how the LfD output plan is executed finally with the robot. Here, the focus is on execution strategies with the goal of improving the success rate of a task [44, 97, 92]. In [44] they considered a small-parts assembly scenario, where the tolerances are comparatively low, which leads to more sensitivity to errors in misalignments due to tight tolerances. They have proposed an impedance control strategy to drive the robot along the assembly trajectory generated by LfD, but also record and track a required wrench profile so that tolerance misalignments can be resolved with the compliance of the contact, and in this way increasing the success rate of the task and avoiding failures when there is contact due to tolerance errors. In [97], the authors claim that LfD output performs well in generalization but fails to maintain the required accuracy for a new context. Hence they use the LfD output as an initial guess for an RL algorithm designed according to the task's model, and refine the LfD output find the optimal policy for the specific task.

3.4.3 Robustness and Safety

The general LfD process is mainly concerned with successfully learning and executing a desired task with a specific accuracy, while it is crucial to consider how the process lifecycle is aligned with safety requirements and how well the system can handle unexpected scenarios. This is of extra importance in manufacturing cases where the robot shares an industrial environment alongside humans, with more potential dangers. Although LfD can generalize to new scenarios, there is no guarantee that the devised task policy is aligned with safety requirements or passes through the safety region. It is also not guaranteed that in case of unforeseen situations during the task execution, the robot makes a safe decision to accommodate the new situation. Therefore, it is important to equip the LfD processes with proper mechanisms to ensure robustness and safety in the robot's operational environment.

The concept of robustness gains meaning when there is a possibility of a fault, disturbance, or error throughout the process, that might mislead the learning process or cause the

system to end up in an unknown state. In this case, a robust LfD system is capable of reliably recovering from the imposed state and successfully finishing the task whatsoever. Alongside robustness, the concept of safety ensures that the robot's operations and decisions do not cause any harm to the humans working along, especially during the teachings and interactions. It also ensures that the operations remain in a safe region to prevent damage to the working environment, work objects, and the robot itself. It is evident that robustness and safety are essential components of LfD systems that must be carefully addressed to enable their effective deployment in real-world applications [172, 173].

One main aspect of robustness and safety in LfD processes is related to the Human-Robot Interaction (HRI). Since LfD lifecycle is mainly concerned with interaction with humans, it is evident that focusing on HRI robustness and safety becomes a major trend on enhancing the reliability of LfD systems [174].

One main trend of improving safety and robustness is focused on HRI [27, 145, 29, 111, 107]. This includes the teaching and demonstration as well as any other form of interaction throughout the process. With respect to robustness, in [27], to efficiently learn from suboptimal demonstrations, the paper proposes an RL-based optimization where the demonstrations serve as the constraints of the optimization framework. the objective was to outperform the suboptimal demonstrations and find the optimal trajectory in terms of length and smoothness. The work in [145] proposes an interactive learning approach where instead of depending on perfect human demonstrations to proceed with learning, the human can interactively and incrementally provide evaluative as well as corrective feedback to enhance the robustness of learning against imperfect demonstrations. In terms of safety, in [29] kinesthetic teaching is replaced with teleoperation to enhance safety while providing local corrections to the robot. This is because kinesthetic teaching can become more dangerous with increasing the robot's velocity of execution. Moreover, in many works such as [111, 107] the control scheme is based on impedance control to guarantee compliant interactions with humans, avoid sudden unexpected motions, and improve HRI safety.

Another trend for improving robustness is focused on the robustness against various disturbances and errors through out the LfD cycle [108, 76, 94]. The work in [108] has

focused on the fact that while LfD can allow non-experts to teach new tasks in industrial manufacturing settings, experts are still required to program fault recovery behaviors. They have proposed a framework where robots autonomously detect an anomaly in execution and learn how to tackle that anomaly by collaborating with human. They represent a task via task graphs, where a task is executed from start to end. If the robot detects an anomaly, it waits and asks humans whether to demonstrate a recovery behavior or refine the current execution behavior. In case of learning a new recovery behavior from the teacher, a conditional state is added to the graph at the point of anomaly, and next time the robot checks for the condition to see if it should continue the normal execution or switch to the recovery behavior. Similarly, the authors in [76] proposed Bayesian GMM to quantify the uncertainty of the imitated policy at each state. They fuse the learned imitation policy with various conservative policies in order to make the final policy robust to perturbations, errors and unseen states. For example, in a board wiping task, the imitation policy learned the force profile required to wipe the board, while the conservative policy ensured circular motion on the board. together, they make the board wiping policy robust so that the motion is desirable while applied force is enough to actually wipe the board.

Lastly, there are several works focusing on robustness and safety considering the limitations in the robot's operating environment [151, 152]. In [151], the problem of collision avoidance was considered. They proposed a modified formulation of DMP, where they incorporated a zeroing barrier function in the formulation and solved a nonconvex optimization in order to find a collision free path through their constrained DMP. From another perspective, the work in [152] addressed the issue that there is no guarantee that LfD outcome respect kinematic constraints when generalizing. so they formed a QP optimization problem that enforces the kinematic constraints and finds the DMP weights where the optimal trajectory is closest to the original DMP trajectory while respecting the constraints.

3.4.4 Remarks

Besides the mentioned trends and directions, the future application of LfD is set to undergo substantial evolution driven by technological advancements and shifting industrial

paradigms. Key areas of development include the integration of Large Language Models (LLMs) such as GPT-4, which promise enhanced communication between human operators and robots through natural language processing. LLMs also enable intelligent data analysis, helping to identify patterns and suggest process improvements in manufacturing. Moreover, by facilitating interactive learning and troubleshooting, LfD with LLM integration allows robots to engage in real-time dialogue, enhancing their performance continuously [175].

Furthermore, the adoption of LfD within the framework of Industry 5.0 emphasizes collaboration between humans and machines in manufacturing [176, 177]. This approach enhances personalized assistance for operators, improves workplace safety through robot training in hazardous tasks, and supports adaptive manufacturing processes that respond dynamically to changes. Emerging paradigms like edge computing and Internet of Things (IoT) integration bring real-time data processing capabilities to LfD, enabling quicker decision-making and data-driven insights[178, 179, 180]. Additionally, digital twins provide a virtual environment for testing and optimizing LfD algorithms, while Augmented Reality (AR) and virtual reality (VR) enhance the training and demonstration phases by providing immersive learning experiences [181, 182, 183, 184].

3.5 Concluding Remarks

This chapter presented a structured approach to integrating LfD into the roboticization process using manipulators for manufacturing tasks. It addressed key questions of ”What to Demonstrate,” ”How to Demonstrate,” ”How to Learn,” and ”How to Refine,” providing practitioners with a clear roadmap to implement LfD-based robot manipulation. First, we identified the scope of demonstration based on the desired task’s characteristics and determined the knowledge and skill required to be demonstrated to the robot. Then, based on the scope of demonstration, we explored demonstration methods and how human teachers can provide demonstrations for the robot, providing insights to extract the demonstration mechanism. Next, we focused on the learning approaches to enable efficient task learning and execution from the provided demonstrations. We first explored the possible learning spaces, followed

by the common learning methods along with their pros and cons. Finally, we provided trends and insights to evaluate and improve the LfD process from a practical point of view, giving research directions and objectives for building upon the state of the art.

The final stage of the roadmap involves analyzing and evaluating its effectiveness across various dimensions. Key trends in LfD refinement focus on enhancing learning and generalization performance, accuracy, and robustness/safety. Improving learning and generalization performance aims to bring LfD algorithms closer to human cognitive learning capabilities. This involves strategies like incremental learning, where robots continuously refine skills over time through additional demonstrations or interactions. Interactive learning further engages humans in the teaching process, enabling dynamic feedback and correction during task execution. Active querying methods enhance learning efficiency by strategically requesting informative demonstrations, thereby optimizing the learning process. Accuracy in LfD refers to how precisely and reliably robots execute tasks based on learned behaviors. Strategies to enhance accuracy include modifying teaching methods to allow more precise demonstrations and refining execution strategies to align closely with task requirements. Techniques such as impedance control and reinforcement learning integration help achieve and maintain desired task accuracies. Robustness and safety are critical for deploying LfD systems in real-world environments, especially in industrial settings where robots interact closely with humans.

It is noteworthy to mention that the application scenarios of robots in the manufacturing field are diverse and distinctive. While the state of the art in LfD has advanced significantly in a number of tasks, it is impractical to cover all manufacturing tasks comprehensively due to their diverse nature and the unique challenges each task presents, particularly in unstructured environments. For instance, the complexity integration of LfD into small-part assembly is higher than that of large-part assembly due to the precision required and the greater impact of environmental uncertainties. In unstructured scenarios, which are common in mass customization, the variations are numerous and depend on specific task requirements. These include the scale of objects, the level of uncertainty in the perception system, calibration reliability between the perception system and the robot coordinates, the

available setup at the production site such as the Programmable Logic Controller (PLC) and sensory system alongside the robot, the robot hardware and its capabilities, the available grippers, the need for gripper changes during manufacturing, and many other factors.

Given the vast array of possible variations, addressing each scenario individually is impractical. Therefore, our proposed roadmap serves as a comprehensive guideline to help practitioners navigate these challenges by identifying the main key decision points at each step and providing practical implications, advantages, disadvantages, limitations, and recommendations. The roadmap is meant to be used through iterative attempts to improve the implementation of LfD. For example, in small-part assembly, the initial choice for the scope of demonstration might be motion-based demonstration, typically suitable for large-part assembly. Throughout the implementation, if the uncertainty of the vision system violates the tolerance thresholds, the practitioner can include contact-based demonstration suggested by the roadmap, to address this uncertainty. Similarly, for kinesthetic demonstration, if gravity compensation mode of the available robot is admittance-based and complicates accurate motion demonstration, the practitioner can refer to the roadmap and consider teleoperation to enhance the process. Regarding the learning mechanism, the practitioner might start with joint space and DMP. If DMP's generalization capability proves insufficient, the roadmap advises switching to Cartesian space learning or using another learning method such as GP for their probabilistic benefits. Finally, if the execution speed is low, affecting production efficiency, the roadmap guides the practitioner to explore incremental learning to refine robot behavior and increase speed.

This example is an illustration of how the roadmap is meant to be applied in the operation process and in order to facilitate the transformation from mass production to mass customization, and since variations in the task requirements are numerous, the roadmap does not aim to list every possible scenario but provides a toolkit to equip practitioners with the knowledge to address any arising challenges. It helps them identify decision points, leverage the state of the art, or define new research problems systematically. By doing so, the roadmap ensures a robust framework for handling the diverse and unstructured scenarios in manufacturing, enabling practitioners to iteratively and effectively implement LfD in

various tasks.

By providing a detailed and structured analysis into determining the scope of demonstration, devising demonstration mechanisms, implementing learning algorithms, and refining LfD processes, our review enables both researchers and industry professionals to develop application-based LfD solutions tailored for manufacturing tasks. This paper offered a practical and structured guide, making LfD accessible to practitioners with moderate expertise requirements. Through comprehensive questionnaire-style guidance, we provided step-by-step instructions and main research directions for refining LfD performance in manufacturing settings, thus bridging the gap between research and practice in the field of robotic automation.

Chapter 4

DFL-TORO: A Demonstration Framework for Learning Time-Optimal Robotic tasks via One-shot Kinesthetic Demonstration

This chapter introduces DFL-TORO, a novel framework designed to enhance the implementation of LfD for time-optimal robotic tasks in industrial manufacturing. DFL-TORO addresses critical challenges in leveraging LfD, including the need for efficient, noise-free, and smooth trajectories derived from one-shot kinesthetic demonstrations. By incorporating optimization-based smoothing algorithms and task tolerance acquisition, the framework ensures that robotic tasks achieve both implementation and operational efficiency. DFL-TORO uniquely bridges the gap between human demonstration and robotic learning by providing a preprocessing layer that refines trajectory data before training LfD algorithms.

The rest of the chapter is structured as follows. In Section 4.1 the problem definition and the objectives are formulated in detail. Also, several technical preliminaries are given. In Section 4.2, DFL-TORO is introduced along with its workflow and consisting modules. In section 4.3, experimental results are provided and discussed. Throughout the chapter, the

following notations are utilized. The bold variables, e.g. $\boldsymbol{q}$, represent a vector. $\mathbb{S}^3$ represents the the space of quaternions. Symbol "$\leq$" represents element-wise inequality of vectors when used with vector variables.

4.1 Problem Statement and Preliminaries

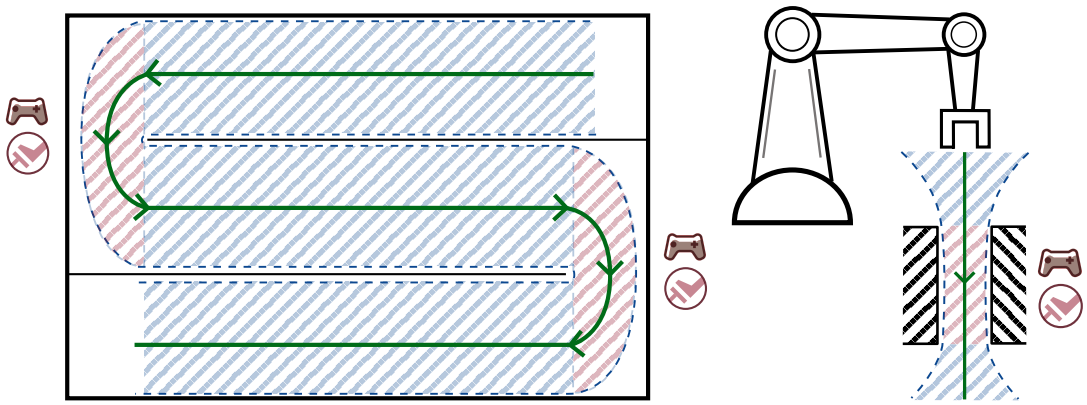
4.1.1 Problem Statement

Let $\boldsymbol{q}_o(t)$ be the initial noisy demonstration trajectory provided via kinesthetic guidance to an n DoF manipulator. The path underlying the demonstration trajectory consists of m waypoints, denoted by w_i for $\forall i = 1, \cdots, m$. Each waypoint w_i includes joint configuration $\mathbf{q}_{w_i} \in \mathbb{R}^n$, the end-effector's position $\mathbf{p}_{w_i} = [x_{w_i}, y_{w_i}, z_{w_i}]^T \in \mathbb{R}^3$ and orientation $\boldsymbol{\theta}_{w_i} \in \mathbb{S}^3$, represented as quaternion. The proposed framework aims to achieve the following objectives:

- **OB1**: Given the w_i and since the original timing law of $\boldsymbol{q}_o(t)$ is slow, we aim to solve for the time-optimal demonstration trajectory $\boldsymbol{q}_f(t)$. Moreover, the noise in the demonstration is inherent in the waypoints w_i and needs to be eliminated in order to achieve the best achievable timing law. We design an Optimization-based Smoothing approach that given the joint position limits $\mathbf{q}_{min}$, $\mathbf{q}_{max} \in \mathbb{R}^n$, joint velocity limits $\mathbf{v}_{min}$, $\mathbf{v}_{max} \in \mathbb{R}^n$, joint acceleration limits $\mathbf{a}_{min}$, $\mathbf{a}_{max} \in \mathbb{R}^n$, and joint jerk limits $\mathbf{j}_{min}$, $\mathbf{j}_{max} \in \mathbb{R}^n$, finds the optimal timing law for the demonstrated path constrained by w_i. Since jerk is the reflection of unfavored points in the demonstration [20], i.e. demonstration noise, our optimization objective also involves minimizing the motion jerk, which pushes the optimization to eliminate the noise from w_i and deliver a smooth noise-free path. Finally, the path of $\boldsymbol{q}_f(t)$ must be constrained to w_i via tolerance bounds to ensure that $\boldsymbol{q}_f(t)$ does not deviate from the original demonstration and the accuracy is retained.

- **OB2**: The optimization process in **OB1** overrides the timing law of $\boldsymbol{q}_o(t)$ with the fastest timing law. Therefore, the objective is to provide a one-time Refinement Phase, where the human teacher can slow down the timing of $\boldsymbol{q}_f(t)$. In the meantime, the task

tolerances $\boldsymbol{\epsilon}_p^i \in \mathbb{R}^3$ and $\epsilon_\theta^i \in \mathbb{R}$ are extracted for each waypoint w_i. Although the demonstrations are represented in the joint space, we extract tolerance bounds in the Cartesian space, as the end-effector pose is the ultimate point of accuracy for the task. Hence, $\boldsymbol{\epsilon}_p^i$ is the allowed tolerance for the deviation from $\mathbf{p}_{w_i}$, and ϵ_θ^i is the allowed tolerance for the deviation from $\boldsymbol{\theta}_{w_i}$ in terms of the value of angular difference. The outcome of this phase is fine-tuned trajectory $\boldsymbol{q}_f^r(t)$, taking tolerance values and timing modifications into account. $\boldsymbol{q}_f^r(t)$ is the final demonstration trajectory that not only respects the accuracy tolerance bounds of the task, but performs with an optimal execution time.



(a) Maze navigation (b) Narrow passage

Figure 4.1: Illustrative examples to clarify the proposed refinement concept. The shaded regions represent the tolerance bounds. Red shaded areas indicate zones where the robot is intuitively expected to slow down based on teleoperated brake commands from the human operator. These regions require narrower tolerance bounds compared to other parts of the trajectory, represented by blue-shaded areas.

To clarify the concept of **OB2**, we present two illustrative examples in Fig. 4.1. In the first example, depicted in Fig. 4.1(a), imagine the robot's end-effector navigating through a maze along the green line at maximum speed. The human operator can teleoperate the system to slow down the robot whenever necessary. Naturally, the robot's behavior is more critical at the maze's turning points, where it is more reliable if the robot reduces its speed. These turning points coincide with regions where the tolerance bounds must be narrower to avoid

colliding with the maze walls.

Similarly, in Fig. 4.1(b), when the robot encounters a narrow passage, it is intuitive to slow down to ensure reliable execution. This passage also requires narrower tolerance bounds. Based on this conceptual analogy, our refinement objective in **OB2** is to enable the human operator to decrease the robot's speed to ensure reliable execution. Simultaneously, meaningful tolerance bounds are determined based on where a reduction in execution speed was commanded.

4.1.2 Preliminary Concepts

4.1.2.1 Dynamic Movement Primitive

The formulation of DMP consists of two main components: a transformation system and a canonical system [143, 149]. The transformation system governs the shape of the trajectory. It is described by a second-order differential equation resembling a damped spring-mass system with an additional forcing term to encode the desired movement shape:

$$\tau \dot{z} = K(g - y) - Dz - K(g - y_0)x + Kf(x) \tag{4.1}$$

$$\tau \dot{y} = z, \tag{4.2}$$

where y and z are the system's position and velocity, respectively. τ is a temporal scaling vector. K and D are positive constants that determine the system's stiffness and damping, respectively. y_0 is the start and g is the goal states. $f(x)$ is a nonlinear forcing term that shapes the trajectory. x is the phase variable from the canonical system. The forcing term $f(x)$ is typically represented as a weighted sum of basis functions as:

$$f(x) = \frac{\sum_{\mathrm{i}=1}^{N} \psi_{\mathrm{i}}(x) \mathrm{w}_{\mathrm{i}} x}{\sum_{\mathrm{i}=1}^{N} \psi_{\mathrm{i}}(x)}, \tag{4.3}$$

where $\psi_{\mathrm{i}}(x)$ are Gaussian basis functions defined as

$$\psi_{\mathrm{i}}(x) = \exp(-h_{\mathrm{i}}(x - c_{\mathrm{i}})^2). \tag{4.4}$$

$\mathrm{w_i}$ are the weights and c_i and h_i are the centers and widths of the basis functions, respectively.

The canonical system drives the phase variable x, which monotonically decreases from 1 to 0 during the course of the movement. It is governed by the following first-order differential equation:

$$\tau \dot{x} = -\alpha_x x, \tag{4.5}$$

where α_x is a constant that controls the speed of phase progression.

4.1.2.2 B-Spline

B-Splines, or Basis Splines, are a family of piecewise-defined polynomials used in robotics to represent trajectories. B-Splines provide a powerful way to model smooth curves that can be easily manipulated by controlling a set of control points [36]. Their piecewise polynomial nature ensures smooth motion paths, making them highly suitable for complex motion planning.

The basis of B-Splines is the set of basis functions, which are defined recursively. The B-Spline basis functions of degree k, denoted $N_{\hat{i},k}(s)$, are defined over a non-decreasing sequence of real numbers called the knot vector, $U = \{u_0, u_1, ..., u_{\hat{n}+k+1}\}$. The recursive definition of the basis functions is as follows:

1. Zero-degree (piecewise constant) basis functions defined as:

$$N_{\hat{i},0}(s) = \begin{cases} 1 & \text{if } u_{\hat{i}} \leq s < u_{\hat{i}+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

2. Higher-degree basis functions defined as:

$$N_{\hat{i},k}(s) = \frac{s - u_{\hat{i}}}{u_{\hat{i}+k} - u_{\hat{i}}} N_{\hat{i},k-1}(s) + \frac{u_{\hat{i}+k+1} - s}{u_{\hat{i}+k+1} - u_{\hat{i}+1}} N_{\hat{i}+1,k-1}(s). \tag{4.7}$$

where $N_{\hat{i},k-1}(s)$ and $N_{\hat{i}+1,k-1}(s)$ are the basis functions of degree $k-1$.

A B-Spline curve of degree k is defined as a linear combination of these basis functions. Given a set of control points $\{\boldsymbol{P}_0, \boldsymbol{P}_1, ..., \boldsymbol{P}_{\hat{n}}\}$, the B-Spline curve $\boldsymbol{\xi}(s)$ is defined by:

$$\boldsymbol{\xi}(s) = \sum_{\hat{i}=0}^{\hat{n}} N_{\hat{i},k}(s) \boldsymbol{P}_{\hat{i}}, \tag{4.8}$$

where $\boldsymbol{P}_{\hat{i}}$ are the control points and $N_{\hat{i},k}(s)$ are the B-Spline basis functions.

4.1.3 Preliminary Functions

The following functions are used in the optimization procedure:

4.1.3.1 QuatDiff Function

The $\mathsf{QuatDiff}(.,.) \in [0, \pi]$ function is a conventional function derived based on the dot product of quaternions to compute the absolute angular difference between two quaternions [185]. It is mathematically defined as:

$$\mathsf{QuatDiff}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = 2 \cdot \arccos(|\boldsymbol{\theta}_1 \cdot \boldsymbol{\theta}_2|). \tag{4.9}$$

where $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are the quaternions representing the orientations, and the dot product $|\boldsymbol{\theta}_1 \cdot \boldsymbol{\theta}_2|$ provides the cosine of the angle between them. This formula effectively computes the smallest angle required to rotate from one orientation to the other. During the trajectory optimization, the function ensures that the end-effector's orientation remains within a specified tolerance of the desired orientation at each waypoint.

4.1.3.2 Forward Kinematics Function

The forward kinematic function, denoted as $\mathsf{F_K}$, is used to determine the position and orientation of a robot's end-effector based on its joint configurations. This function maps the joint configuration of $\boldsymbol{q}_f(t)$ to the end-effector's position $\boldsymbol{p}_f(t)$ and orientation $\boldsymbol{\theta}_f(t)$ in the Cartesian space. Mathematically, this is expressed as:

$$(\boldsymbol{p}_f(t), \boldsymbol{\theta}_f(t)) = \mathsf{F_K}(\boldsymbol{q}_f(t)). \tag{4.10}$$

Here, $\boldsymbol{p}_f(t) = [x_f(t), y_f(t), z_f(t)] \in \mathbb{R}^3$ is a vector representing the position coordinates, and $\boldsymbol{\theta}_f(t) \in \mathbb{S}^3$ represents the orientation in quaternion form.

Remark: As this work focuses on representing the robot trajectory via B-Splines, the span of the knot values is defined within the range $[0, 1]$ which means that $s \in [0, 1]$. Therefore, $\boldsymbol{\xi}(s)$ models the trajectory over the normalized time scale s. An additional parameter T is associated with $\boldsymbol{\xi}(s)$ which maps s into actual time t. Finally, $\boldsymbol{P}_{\hat{i}}$ and T are the decision variables to be solved via optimization. Throughout this paper, we refer to s as normalized time and refer to t as actual/real time.

4.2 DFL-TORO Methodology

To achieve **OB1** and **OB2**, we have compiled our proposed methods into a comprehensive demonstration framework, DFL-TORO. This section details the methods and approaches incorporated in the DFL-TORO workflow. We begin with an overview of the workflow, illustrating how DFL-TORO transforms noisy demonstrations into optimal ones while addressing **OB1** and **OB2**. Following this, we explore the technical details of the DFL-TORO modules specifically designed to meet these objectives.

4.2.1 Overall Workflow

The overall workflow of DFL-TORO is illustrated in Fig. 4.2, with the steps labeled from (A) to (D). In this section, we provide a detailed walkthrough of each step, indicating where

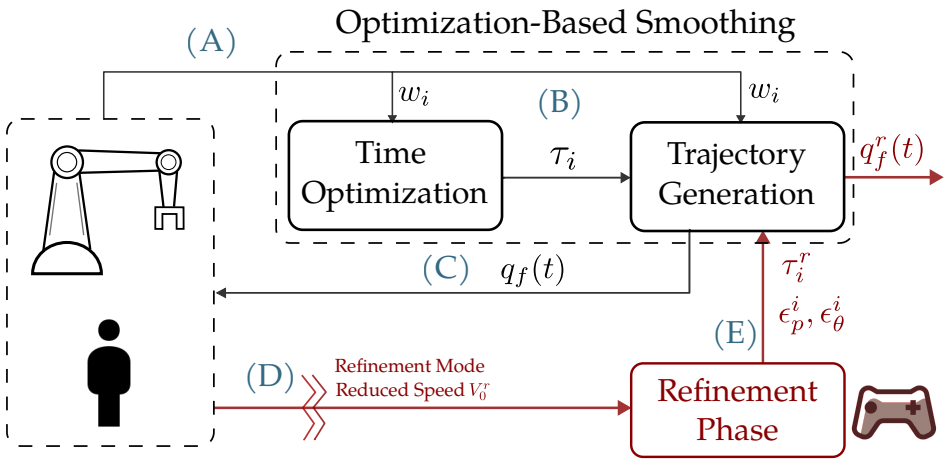


Figure 4.2: DFL-TORO workflow. Red arrows indicate interactive procedures with the human-robot in the loop.

OB1 and **OB2** are addressed. Moreover, the workflow of DFL-TORO involving these steps is provided in Algorithm 1.

A) We start by recording $\boldsymbol{q}_o(t)$ from the inherently noisy human demonstration. The underlying path is then extracted and represented as w_i. The derivation of w_i involves filtering the end-effector's position and orientation, ensuring the robot moves beyond a certain threshold before considering a new waypoint. This approach allows us to extract meaningful waypoints from the path.

B) The "Optimization-based Smoothing" module takes w_i as input and solves a comprehensive optimization problem to satisfy the requirements of **OB1**. The outcome of this module is $\boldsymbol{q}_f(t)$.

C) The optimized trajectory $\boldsymbol{q}_f(t)$ is replayed on the robot under the supervision of the human teacher. In this step, the teacher analyses the execution performance of $\boldsymbol{q}_f(t)$ and checks whether the task is reliably performed under the new noise-free path and the optimized timing law. In case of inaccuracies or unreliable execution speed during the performance, the teacher proceeds with the refinement of $\boldsymbol{q}_f(t)$.

D) The "Refinement Phase" allows the teacher to interactively slow down and correct the

timing law. The robot replays the trajectory, allowing the human teacher to visually assess the execution speed. The teacher can pinpoint areas where the speed is either unreliable or unsafe and determine which segments require slowing down. Since the current trajectory $\boldsymbol{q}_f(t)$ is rapidly executed, it leaves no time for humans to observe the motion and provide feedback simultaneously. Therefore, the Refinement Phase operates at a reduced speed $\mathrm{v}_0^r \in \mathbb{R}_+$, giving humans enough reaction time to make changes.

E) The Refinement Phase provides revised timing law for the trajectory, as well as the task tolerances $\boldsymbol{\epsilon}_p^i$ and ϵ_θ^i extracted from human feedback. These updated values are subsequently fed to the "Optimization-based Smoothing" module to calculate a fine-tuned trajectory $\boldsymbol{q}_f^r(t)$ in accordance with the new timings and tolerances. This step is the last step of the workflow and the one that satisfies the requirements of **OB2**.

As mentioned, step (B) is where **OB1** is addressed via the Optimization-based Smoothing module, while **OB2** is addressed in steps (D) and (E) through the Refinement Phase. In the following sections, we elaborate on these modules and how our proposed methods tackle **OB1** and **OB2**.

Remark: Since $\boldsymbol{\epsilon}_p^i$ and ϵ_θ^i are extracted in the Refinement Phase (Step (D)), the Trajectory Generation module utilizes a set of default tolerance values $\boldsymbol{\epsilon}_p^d$ and ϵ_θ^d in step (B) to find $\boldsymbol{q}_f(t)$. After obtaining tolerance bounds, step (E) fine-tunes the trajectory to get $\boldsymbol{q}_f^r(t)$ based on the acquired tolerances, replacing the default values.

4.2.2 Optimization-based Smoothing

In this section, we explain our proposed method underlying the Optimization-based Smoothing, designed to transform $\boldsymbol{q}_o(t)$ into $\boldsymbol{q}_f(t)$. The objective is to find the best timing law given w_i and regulate the jerk profile $\dddot{\boldsymbol{q}}_f(t)$, as an indication of noise. The constraints are the kinematic limits of the robot, as well as tolerance values binding $\boldsymbol{q}_f(t)$ to the waypoints w_i. Mathematically, let t_i be the decision variable representing the time at which $\boldsymbol{q}_f(t)$ passes

through w_i. Considering (4.10), $\boldsymbol{q}_f(t)$ must satisfy the following constraint:

$$-\boldsymbol{\epsilon}_p^i \leq \boldsymbol{p}_f(t_i) - \mathfrak{p}_{w_i} \leq \boldsymbol{\epsilon}_p^i, \tag{4.11}$$

$$\mathsf{QuatDiff}(\boldsymbol{\theta}_f(t_i), \boldsymbol{\theta}_{w_i}) \in [0, \epsilon_\theta^i], \tag{4.12}$$

Let $\boldsymbol{\xi}_f(s)$ with control points $\boldsymbol{P}_{\hat{i},f}$ and duration variable T_f be the B-Spline model representing $\boldsymbol{q}_f(t)$. To enforce the constraints of (4.11) and (4.12), we need to calculate the symbolic equation of $\boldsymbol{\xi}_f(\tau_i)$ where $\tau_i = \frac{t_i}{T_f}$, with respect to the decision variables $\boldsymbol{P}_{\hat{i},f}$ and τ_i. According to (4.8) this leads to:

$$\boldsymbol{\xi}_f(\tau_i) = \sum_{\hat{i}=1}^{\hat{n}} N_{\hat{i},k}(\tau_i) \boldsymbol{P}_{\hat{i},f}. \tag{4.13}$$

Following (4.6) and (4.7), this requires calculation of $N_{\hat{i},0}(\tau_i)$ as:

$$N_{\hat{i},0}(\tau_i) = \begin{cases} 1 & \text{if } u_{\hat{i}} \leq \tau_i < u_{\hat{i}+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

where $u_{\hat{i}}$ are the values from the knot vector. The issue here is that since τ_i is not known, (4.14) leads to a highly non-linear behavior and does not result in a symbolic equation to be used for optimization. This is due to the fact that the timing of each waypoint across the B-Spline determines which basis function is used for the B-Spline equation, and since timing is a decision variable, it is not known between which knot values w_i falls. Therefore, optimizing for w_i and t_i simultaneously leads to an ill-defined and highly nonlinear optimization formulation.

To overcome this, we approach the problem by solving a two-stage optimization problem. The first stage, i.e. the Time Optimization module, considers τ_i as decision variables and solves for the timing law. Then, in the Trajectory Generation module, w_i are treated as decision variables, and a comprehensive optimization problem is solved to eliminate the noise and regulate the jerk profile. In the following, we delve into the details of the Time

Optimization and the Trajectory Generation modules.

4.2.2.1 Time Optimization Module

The objective of this optimizer is to find the minimum-time trajectory $\boldsymbol{q}_t(t)$ that strictly passes through all the waypoints $\mathbf{q}_{w_i}$. At this stage, we ignore constraints related to acceleration and jerk since the noise is still present in the path. Adding such constraints would prevent the optimizer from finding the ideal timings. B-Spline sub-trajectories $\boldsymbol{\xi}_j$ for $\forall j = 1, \cdots, m-1$ are used to represent the trajectory between every two adjacent waypoints w_j and w_{j+1}, with control points $\boldsymbol{P}_{i,j}$ and durations T_j, as shown in Fig. 4.3(a). Then, the Time Optimization problem is formulated as:

$$(\boldsymbol{P}^*_{i,j}, T^*_j) = \underset{\boldsymbol{P}_{i,j}, T_j}{\arg\min} \sum_{j=1}^{m-1} T_j, \tag{4.15a}$$

$$\boldsymbol{q}_j(t_j) = \boldsymbol{\xi}_j(\frac{t_j}{T_j}), \tag{4.15b}$$

$$\mathbf{q}_{min} \leq \boldsymbol{q}_j(t_j) \leq \mathbf{q}_{max}, \tag{4.15c}$$

$$\mathbf{v}_{min} \leq \dot{\boldsymbol{q}}_j(t_j) \leq \mathbf{v}_{max}, \tag{4.15d}$$

$$\boldsymbol{\xi}_j(0) = \mathbf{q}_{w_j}, \tag{4.15e}$$

$$\boldsymbol{\xi}_j(1) = \mathbf{q}_{w_{j+1}}, \tag{4.15f}$$

$$\dot{\boldsymbol{\xi}}_1(0) = \dot{\boldsymbol{\xi}}_{m-1}(1) = 0, \tag{4.15g}$$

for $\forall t_j \in [0, T_j]$. (4.15b) relates the normalized and actual trajectories. We set bounds $\mathbf{q}_{min}$, $\mathbf{q}_{max}$, $\mathbf{v}_{min}$ and $\mathbf{v}_{max}$ for joint position and velocity as (4.15c) and (4.15d), respectively. Continuity constraints (4.15e) and (4.15f) are applied at the intersection of these sub-trajectories. (4.15g) enforces the trajectory to start and rest with zero velocity. $\boldsymbol{q}_t(t)$ is a compilation of all the sub-trajectories $\boldsymbol{q}_j$ together.

It is important to note that the result of this optimization is typically an infeasible trajectory, as the existing noise causes high acceleration and jerk values. (4.15a) determines the ideal timings T^*_j to move through all waypoints with highest velocity, which provides normalized

time τ_i for $\forall i = 1, \cdots, m$ defined as

$$\tau_i = \begin{cases} \frac{\sum_{j=1}^{i-1} T_j^*}{T_{m-1}^*} & i \geq 2 \\ 0 & i = 1 \end{cases}. \tag{4.16}$$

This sets the groundwork for the Trajectory Generation module, which addresses the full optimization problem.

4.2.2.2 Trajectory Generation Module

Given τ_i, we fit one B-Spline for $\boldsymbol{q}_f(t)$ across all the waypoints, denoted as $\boldsymbol{\xi}_f$ with control points $\boldsymbol{P}_{\hat{i},f}$ and duration variable T_f, as shown in Fig. 4.3(b). We match the number of control points with the waypoints, i.e., $\hat{n} = m - 1$, to give flexibility to the optimizer to locally adjust the trajectory around each waypoint. The cost function is formulated as:

$$J_f = \alpha T_f + \beta \int_0^1 ||\dddot{\boldsymbol{\xi}}_f||^2 ds + \gamma \sum_{i=1}^{m} ||\boldsymbol{P}_{\hat{i},f} - \mathbf{q}_{w_i}||^2, \tag{4.17}$$

where α, β and γ are positive weights. The initial term T_f involves the minimization of the overall execution time of $\boldsymbol{q}_f(t)$. Note that the Time Optimization module calculates the relative timing of waypoints in normalized time (τ_i), while the actual timing law is optimized in this stage with respect to all the kinematic limits and meanwhile the noise removal. The term $\int_0^1 ||\dddot{\boldsymbol{\xi}}_f||^2 ds$ attempts to minimize the normalized jerk profile, exploiting the tolerances of waypoints. This term is the key objective that pushes the optimization to eliminate the noise of the waypoints. The reason the normalized trajectory $\boldsymbol{\xi}_f$ is used in this term is that if $\boldsymbol{q_f}$ is used, the optimizer can simply increase T_f to reduce jerk, instead of modifying the path. Therefore, using $\boldsymbol{\xi}_f$ isolates the jerk values from T_f and forces the optimization to optimize the path. Finally, the term $\sum_{i=1}^{m} ||\boldsymbol{P}_{\hat{i},f} - \mathbf{q}_{w_i}||^2$ ensures that control points align closely with the original joint configurations at the waypoints. Given the robot manipulator's kinematic redundancy, an end-effector pose can correspond to multiple configurations. This term limits the robot's configuration null space, prompting the optimizer to remain near the

initial configuration. The Trajectory Generation problem is formulated as follows:

$$(\boldsymbol{P}^*_{\hat{i},f}, T^*_f) = \underset{\boldsymbol{P}_{i,f}, T_f}{\arg\min}\, J_f, \tag{4.18a}$$

$$\boldsymbol{q}_f(t) = \boldsymbol{\xi}_f(\frac{t}{T_f}), \tag{4.18b}$$

$$\mathfrak{q}_{min} \leq \boldsymbol{q}_f(t) \leq \mathfrak{q}_{max}, \tag{4.18c}$$

$$\mathfrak{v}_{min} \leq \dot{\boldsymbol{q}}_f(t) \leq \mathfrak{v}_{max}, \tag{4.18d}$$

$$\mathfrak{a}_{min} \leq \ddot{\boldsymbol{q}}_f(t) \leq \mathfrak{a}_{max}, \tag{4.18e}$$

$$\mathfrak{j}_{min} \leq \dddot{\boldsymbol{q}}_f(t) \leq \mathfrak{j}_{max}, \tag{4.18f}$$

$$\boldsymbol{\xi}_f(0) = \mathfrak{q}_{w_1}, \tag{4.18g}$$

$$\boldsymbol{\xi}_f(1) = \mathfrak{q}_{w_m}, \tag{4.18h}$$

$$\dot{\boldsymbol{\xi}}_f(0) = \dot{\boldsymbol{\xi}}_f(1) = 0, \tag{4.18i}$$

$$(\boldsymbol{p}_f(t), \boldsymbol{\theta}_f(t)) = \mathsf{F_K}(\boldsymbol{q}_f(t)), \tag{4.18j}$$

$$-\boldsymbol{\epsilon}^i_p \leq \boldsymbol{p}_f(\tau_i T_f) - \mathfrak{p}_{w_i} \leq \boldsymbol{\epsilon}^i_p, \tag{4.18k}$$

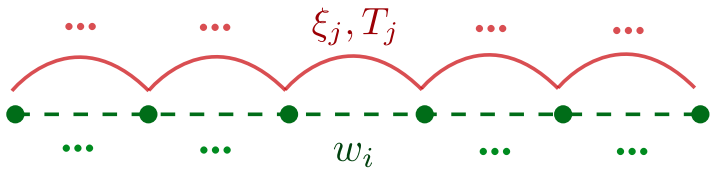
$$\mathsf{QuatDiff}(\boldsymbol{\theta}_f(\tau_i T_f), \boldsymbol{\theta}_{w_i}) \in [0, \epsilon^i_\theta], \tag{4.18l}$$

for $\forall t \in [0, T_f]$. We have introduced acceleration and jerk bounds $\mathfrak{a}_{min}$, $\mathfrak{a}_{max}$, $\mathfrak{j}_{min}$ and $\mathfrak{j}_{max}$ in (4.18e) and (4.18f), respectively, to ensure the trajectory's feasibility. Constraints (4.18g)-(4.18i) enforce the trajectory to start at $\mathfrak{q}_{w_1}$ and rest at $\mathfrak{q}_{w_m}$ with zero velocity. (4.18j) represents the forward kinematic function $\mathsf{F_K}$, which is introduced in (4.10). The goal is to limit the position and angle deviation of the end-effector at each waypoint w_i to $\boldsymbol{\epsilon}^i_p$ and ϵ^i_θ, via (4.18k) and (4.18l), respectively, at normalized time τ_i. Solving (4.18a) yields $\boldsymbol{P}^*_{\hat{i},f}$, with which $\boldsymbol{q}_f(t)$ is computed using (4.8) and (4.18b).

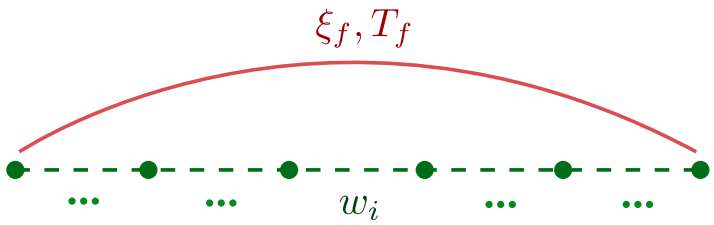
Note that in the case of using default tolerance bounds (step (B)), the equations (4.18k) and (4.18l) are replaced with:

$$-\boldsymbol{\epsilon}^d_p \leq \boldsymbol{p}_f(\tau_i T_f) - \mathfrak{p}_{w_i} \leq \boldsymbol{\epsilon}^d_p, \tag{4.19a}$$

$$\mathsf{QuatDiff}(\boldsymbol{\theta}_f(\tau_i T_f), \boldsymbol{\theta}_{w_i}) \in [0, \epsilon^d_\theta]. \tag{4.19b}$$



(a) Time Optimization



(b) Trajectory Generation

Figure 4.3: Illustration of B-Spline fitting approach for each optimization step. The green lines represent the path and waypoints. Each red arc represents one B-Spline.

4.2.3 Refinement Phase

Until this point, the one-shot demonstration process is finished, and $\boldsymbol{q}_f(t)$ is optimized to deliver the best timing law with default tolerances. Although the best timing law is naturally the most desirable in manufacturing and $\boldsymbol{q}_f(t)$ can be used as is, there might be cases where the human teacher prefers to slow down the execution in some parts or modify the default tolerances to imply a better accuracy and increase execution reliability and success rate. In this case, humans can optionally go through the Refinement Phase, where they can adjust the velocity of execution by slowing down $\boldsymbol{q}_f(t)$ and simultaneously teach the tolerance bounds.

The refinement process starts with the robot replaying $\boldsymbol{q}_f(t)$ under the supervision of the human teacher. the role of the human teacher is to monitor the motion and give real-time teleoperated feedback on every segment of $\boldsymbol{q}_f(t)$ that the robot executes. The human feedback is only in the form of a brake command denoted as $C(t) \in [-1, 0]$, stating where the trajectory should slow down. At every time t during the execution of $\boldsymbol{q}_f(t)$, $C(t)$ determines how much the velocity of execution of $\boldsymbol{q}_f(t)$ should be reduced, with $C(t) = -1$ indicating maximum reduction, and $C(t) = 0$ indicating no reduction in the execution velocity. In other

Algorithm 1 DFL-TORO Architecture

(A): Kinesthetic Guidance
Input: Noisy human demonstration $\boldsymbol{q}_o(t)$
Output: Path waypoints $w_i = \left(\mathbf{q}_{w_i}, \mathbf{p}_{w_i}, \boldsymbol{\theta}_{w_i}\right)$
(B-1): Time Optimization
Input: w_i
Find ideal timings using (4.15a)-(4.15g) and (4.16)
Output: Initial timings τ_i
(B-2): Trajectory Generation
Input: w_i, τ_i, default tolerances $\boldsymbol{\epsilon}_p^d, \epsilon_\theta^d$
Solve for smooth trajectory using (4.18a)-(4.18i) and (4.19a)-(4.19b)
Output: Trajectory $\boldsymbol{q}_f(t)$
(C): Replay Trajectory under Supervision
Input: $\boldsymbol{q}_f(t)$
Human teacher assesses the performance of $\boldsymbol{q}_f(t)$
if refinement is required **then**
    Proceed to step (D)
end if
(D): Refinement Phase
Input: $\boldsymbol{q}_f(t)$, human command $C(t)$, η
Initialize $v(0) = \mathrm{v}_0^r, s_r(0) = \mathrm{v}_0^r \cdot t$
while Replaying Trajectory **do**
    Compute $s_r(t)$ using (4.21)
    Send $\xi(s_r(t))$ to robot for interactive feedback
end while
Compute new timings τ_i^r using (4.22)
for each w_i **do**
    $\boldsymbol{\epsilon}_p^i \leftarrow \Gamma_p(\tau_i)$ using (4.23)
    $\epsilon_\theta^i \leftarrow \Gamma_\theta(\tau_i)$ using (4.23)
end for
Output: New timings τ_i^r, new tolerances $\boldsymbol{\epsilon}_p^i, \epsilon_\theta^i$
(E): Trajectory Fine-Tuning
Input: w_i, τ_i^r, $\boldsymbol{\epsilon}_p^i$, ϵ_θ^i
Re-optimize $\boldsymbol{q}_f(t)$ via updated timings and tolerances using (4.18a)-(4.18l)
Output: Optimal Demonstration $\boldsymbol{q}_f^r(t)$

words, the human feedback acts as a brake pedal, decelerating $\boldsymbol{q}_f(t)$ in real time. In this interaction loop, $\boldsymbol{q}_f(t)$ executes with duration T_f, which is fast and makes it cognitively challenging for humans to rapidly observe, respond, and interact with the robot. Hence, $\boldsymbol{q}_f(t)$ is replayed with a reduced speed, giving enough time for the human teacher to observe the motion and provide interactive feedback in the loop.

The following will detail how we can adjust the velocity of $\boldsymbol{q}_f(t)$ and extract meaningful tolerance bounds $\boldsymbol{\epsilon}_p^i$ and ϵ_θ^i, leading to the refined trajectory $\boldsymbol{q}_f^r(t)$. A visual summary of the refinement process is shown in Fig. 4.4, through which all the steps and the variables are shown through the workflow of the Refinement Phase.

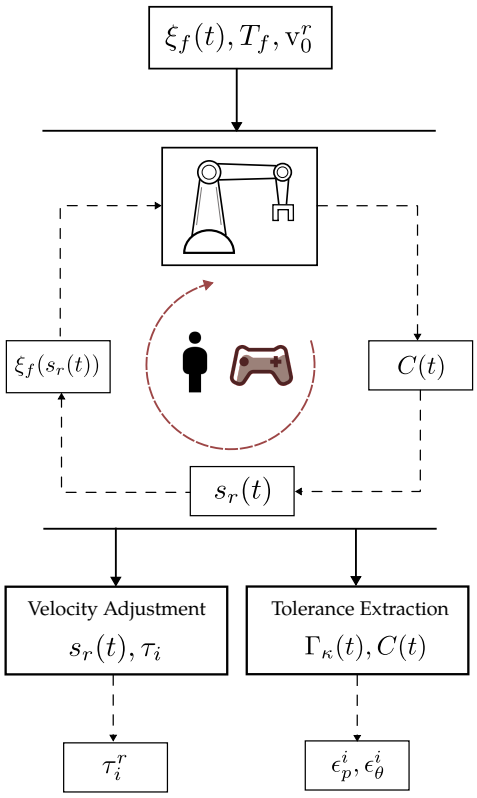


Figure 4.4: Visual diagram of the Refinement Phase. $C(t)$ and $s_r(t)$ are obtained during the interactive loop, leading to velocity adjustment and tolerance extraction.

4.2.3.1 Velocity Adjustment

To apply the deceleration effect of $C(t)$, we consider $\boldsymbol{q}_f(t) = \boldsymbol{\xi}_f(s(t))$, where $\boldsymbol{\xi}_f(s)$ is the trajectory in the normalized time in the range $[0, 1]$, and $s(t)$ is the function that maps the real time t to the normalized time s. Normally, $\boldsymbol{q}_f(t)$ is calculated by setting $s(t) = \frac{t}{T_f}$. However, here we reduce the speed by a factor of $\eta > 1$ by setting

$$s(t) = \mathrm{v}_0^r \cdot t, \tag{4.20}$$

where $\mathrm{v}_0^r = \frac{1}{\eta T_f}$. To apply the deceleration effect of $C(t)$, we manipulate the mapping of $s(t)$ in (4.20). Instead of directly altering the trajectory itself, we take $C(t)$ as a time deceleration factor, and slow down the progression of time in (4.20). This leads to a modified mapping $s_r(t)$. Using $C(t)$, this mapping is computed by integration the following equations throughout the interactive process:

$$\begin{cases} \dot{v}_r(t) = C(t) & v_r(t) \in [\mathrm{v}_{min}^r, \mathrm{v}_0^r], \\ \dot{s}_r(t) = v_r(t) & s_r(t) \in [0, 1] \end{cases}, \tag{4.21}$$

To prevent the robot from completely stopping, we bound $v_r(t)$ to a minimum execution speed $\mathrm{v}_{min}^r \in \mathbb{R}_+$. In this way, slowing down portions of $\boldsymbol{q}_f(t)$ is achieved via calculation and execution of $\boldsymbol{\xi}_f(s_r(t))$ throughout the interaction loop.

At the end of the interaction loop, by having $s_r(t)$, we can transform τ_i to the updated timing τ_i^r as

$$\tau_i^r = \frac{s_r^{-1}(\tau_i)}{s_r^{-1}(\tau_m)}, \tag{4.22}$$

where s_r^{-1} is the inverse of s_r obtained via interpolation. It is worth mentioning that refinement at the reduced speed v_0^r applies the same proportional adjustment on the real speed of execution, as τ_i^r are in normalized time.

4.2.3.2 Tolerance Extraction

To extract the tolerances, we pose a direct correlation between the value of $C(t)$ and the desired level of accuracy at a given point on the trajectory. This means that the more intensively the brake pedal is pressed, the greater the criticality of that trajectory portion, demanding increased precision. This is captured by introducing the functions $\Gamma_p(t)$ and $\Gamma_\theta(t)$, defined as:

$$\Gamma_\kappa(t) = (\epsilon_\kappa^{max} - \epsilon_\kappa^{min})(1 - (-C(t))^{\bar{n}+\zeta}) + \epsilon_\kappa^{min}, \tag{4.23}$$

where $\kappa \in \{p, \theta\}$. In (4.23), $C(t) = 0$ is associated with maximum tolerances $\boldsymbol{\epsilon}_p^{max}$ and ϵ_θ^{max}, whereas a $C(t) = -1$ corresponds to the minimum tolerances $\boldsymbol{\epsilon}_p^{min}$ and ϵ_θ^{min}. The parameters ζ and $\bar{n}$ determine the shape and curve of Γ_κ. By computing $\Gamma_\kappa(\tau_i)$ for each waypoint, we extract the requisite tolerances and pass it along with τ_i^r for re-optimization.

4.3 Validation Experiments

In order to validate the effectiveness of the proposed demonstration framework, DFL-TORO, experiments were conducted on various robotic tasks by recording one-shot kinesthetic demonstrations. The purpose of these experiments is to validate the effectiveness of the DFL-TORO framework across different setups, including various robotic platforms and task environments. The experiments are designed to demonstrate the framework's ability to optimize trajectories' timing law, reduce noise, and regulate jerk. Our primary objectives are as follows:

- Validate DFL-TORO's performance concerning trajectory smoothness, execution time, and noise reduction.
- Showcase the enhancement of LfD outcomes through optimal demonstrations using DFL-TORO, exemplified by a case study with DMPs.

4.3.1 Validation Setup and Methodologies

4.3.1.1 Robotic Setup

In this study, the FR3 robotic platform is used for evaluation. It is equipped with a 7 DoF arm, capable of precise motion control through position, velocity, and torque commands. The experiments conducted with the FR3 are focused on analyzing the performance of DFL-TORO across its various modules.

4.3.1.2 Data Collection

The demonstrations were recorded via kinesthetic guidance, capturing the robot's joint configurations through its joint encoders. These data were captured using a ROS interface at a sampling frequency of 10 Hz. Waypoints w_i are automatically selected based on an end-effector movement threshold, requiring a shift of at least 1 cm or 0.1 radians. For teleoperation, a wireless controller was employed to send continuous brake commands. The kinematic limits of FR3 are collected from the manufacturer's datasheets.

4.3.1.3 Software Configuration

Regarding the software configuration of the robotic setups, the FR3 robot is operated using the "libfranka" ROS interface [186]. For capturing demonstrations, the robot is set to gravity-compensation mode, where torque commands are applied to the joints to compensate for the robot's weight, allowing the user to move the joints freely. For trajectory execution, the robot receives position commands at a control frequency of 1 kHz.

To implement the optimization modules, we utilize the PyDrake toolbox [187]. The Time Optimization module is solved using the SNOPT solver, while the Trajectory Generation module is handled by the IPOPT solver. Additionally, the implementation of DMP was based on the "dmpbbo" toolbox as referenced in [188].

4.3.1.4 Implementation details

Regarding the implementation parameters, B-Splines of order $k = 4$ are selected. This choice of parameter is for the sake of achieving smoothness up to the velocity level. The knot values U are distributed in a clamped uniform manner [36] to create basis functions in (4.8). The weights in (4.17) are set as $\alpha = 1$, $\beta = 0.04$, and $\gamma = 1$. These parameters are chosen through trial and error and are task-independent, i.e., it is not required to alter these coefficients from one robotic setup or robotic task to another. Finally, default tolerance values are chosen as $\boldsymbol{\epsilon}_p^d = [2, 2, 2]$ cm and $\epsilon_\theta^d = 0.1$ radians. The choice of default tolerance values is mainly based on the context requirements of the robotic task and the quality of the captured demonstration. For example, if the task involves an accurate motion and the demonstration is recorded deliberately, lowering the default values will force the optimization to stay closer to the demonstration and hence maintain a higher accuracy.

In Refinement Phase, $\eta = 5$ and $\mathrm{v}_{min}^r = 0.2\mathrm{v}_0^r$ are selected. These parameters determine how slow the demonstration is to be replayed during the refinement, as well as the limiting cap on how much the user can locally slow down the trajectory. Also, $\boldsymbol{\epsilon}_p^{max} = [5, 5, 5]$ cm, $\boldsymbol{\epsilon}_p^{min} = [1, 1, 1]$ cm, $\epsilon_\theta^{max} = 0.3$ radians, and $\epsilon_\theta^{min} = 0.1$ radians are chosen. Again, the choice of these parameters is dependent on the requirements of the task, similar to the choice of default tolerance values. Furthermore, the parameters of $\Gamma_\kappa(t)$ are chosen as $\zeta = 0.9$ and $\bar{n} = 1$, determining the relationship between the brake intensity and the tolerance adjustment. While the same parameters could be used for arbitrary tasks, it is possible to alter the parameters in order to modify the sensitivity of tolerance adjustment to the brake commands. For the DMP case study, 15 basis functions were used in (4.3) to train the trajectories in the joint space. For the sake of comparison, DMP generalizations are calculated for the same start and goal configurations as the demonstration trajectory.

4.3.1.5 Evaluation Metrics

The metrics used for evaluating and comparing the trajectories are the duration of the trajectory and the Maximum Absolute Normalized Jerk (MANJ). Since DFL-TORO optimizes

the duration of demonstration trajectories, directly comparing the jerk profiles of these trajectories is not feasible, as a longer execution time naturally results in a lower jerk value. Therefore, we evaluate a normalized jerk value over normalized time $s \in [0, 1]$ and use the maximum absolute value of each trajectory for comparison. For example, to calculate MANJ for $\boldsymbol{q}_f(t)$, we consider $\boldsymbol{\xi}_f(s)$. Let $\xi_{f,1}, \ldots, \xi_{f,n}$ be the trajectories for each individual joint. MANJ is then calculated as follows:

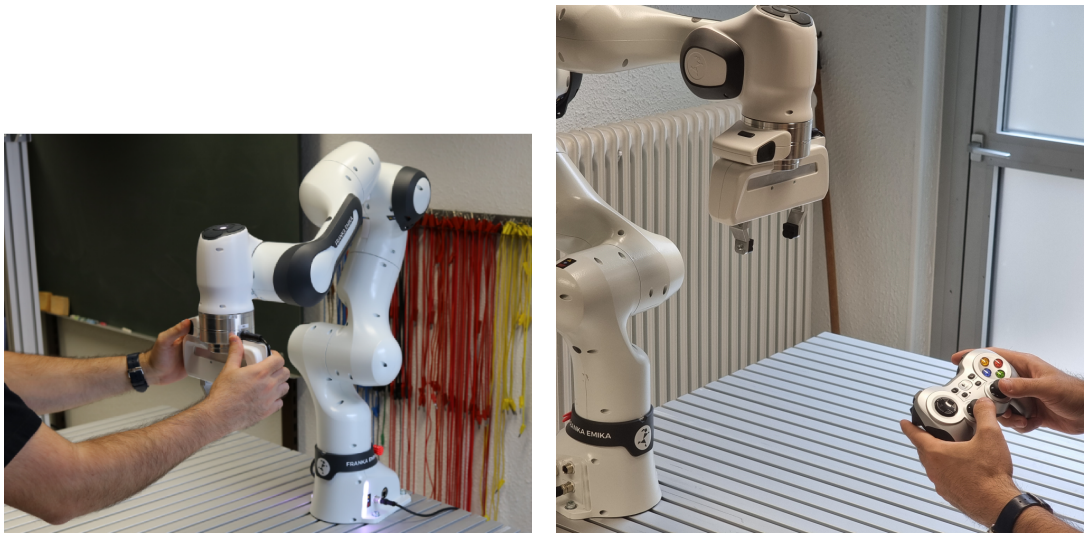
$$\text{MANJ} = \max_{l=1,\ldots,n} (\max_{s} |\dddot{\xi}_{f,l}|). \tag{4.24}$$

This metric effectively indicates the smoothness and noise level in the trajectories. The experimental results were visualized to demonstrate performance across different tasks, and a quantitative analysis was conducted to validate the improvements achieved by our method.

4.3.1.6 Validation Technique

The validation method for this study involves two main techniques. The first technique involves a performance evaluation of the Optimization-based Smoothing and Refinement Phase modules. This method assesses the improvements in time and jerk metrics by comparing the original demonstrations with those optimized by DFL-TORO. For this method, we utilize the FR3 experiments to showcase that the trajectories are free of noise, smooth, within the kinematic constraints, and accurate with respect to the tolerance bounds. The second validation is a case study using DFL-TORO with DMPs. This method highlights that incorporating DFL-TORO before using DMP significantly enhances task execution efficiency and precision. By first applying DFL-TORO to filter noise and optimize timing, the subsequent application of DMP yields more efficient and precise robotic movements compared to using DMP alone. For this technique, we utilize experiments from FR3 to substantiate the improvements in both the quality and efficiency of robotic task execution when DFL-TORO is integrated into the demonstration process. It is worth noting that even though DMP is used in our case study, DFL-TORO applies to any other LfD algorithm and can serve as an

intermediary layer to optimize demonstrations before feeding into the core algorithm.

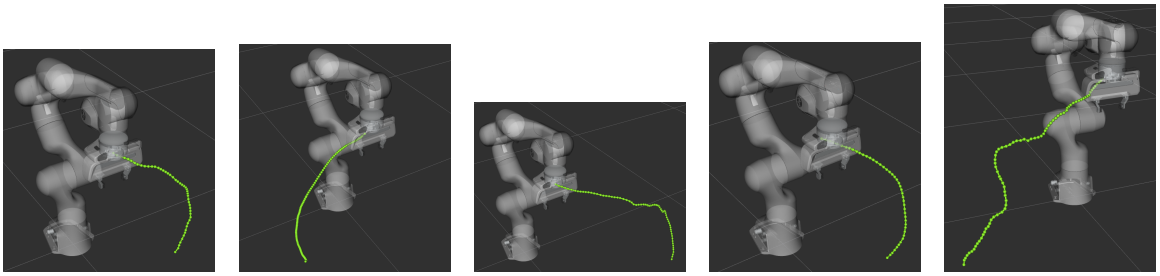


(a) Kinesthetic teaching setup (b) Teleoperation setup

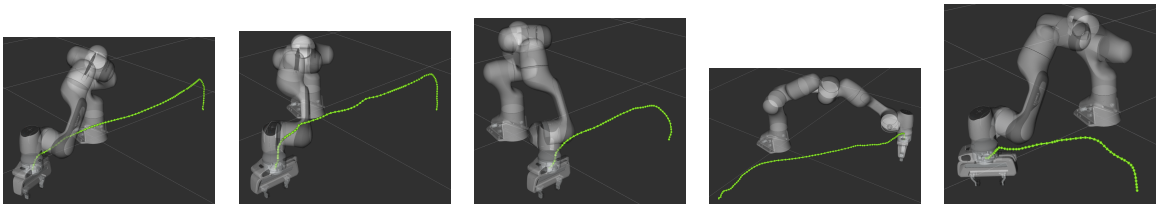
Figure 4.5: The FR3 experimental setup

4.3.2 Experiments with FR3

In the experiments with FR3, A variety of motions are recorded. The experimental setup for FR3 is shown in Fig. 4.5. The demonstrations were recorded to cover typical robotic motions such as in pick-and-place operations, the Reaching motion to pick an object, and the moving motion to place it in another location. These tasks represent the majority of real-world applications. Overall, five different reaching tasks (RT1-RT5) and five different moving tasks (MT1-MT5) are recorded, which is shown in Fig. 4.6. While three tasks (RT1, RT2, MT1) are utilized for visualization purposes, the remaining tasks are included in a comprehensive analysis to demonstrate the robustness and effectiveness of the proposed methodology.



(a) RT1 (b) RT2 (c) RT3 (d) RT4 (e) RT5



(f) MT1 (g) MT2 (h) MT3 (i) MT4 (j) MT5

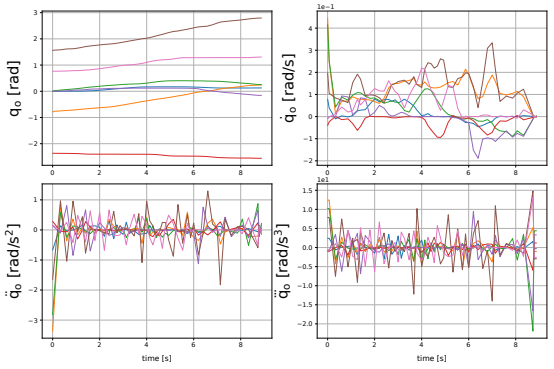
Figure 4.6: Visualization of the demonstration trajectories recorded via FR3.

4.3.2.1 Performance of Optimization-based Smoothing

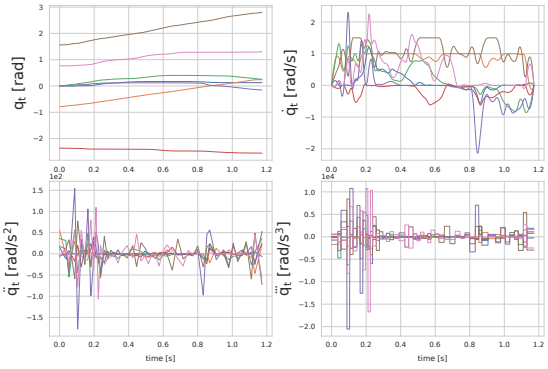
To analyze the performance of the Optimization-based Smoothing module, we visualized the evolution of the RT1 trajectory. The original demonstration's joint trajectory, $\boldsymbol{q}_o(t)$, is shown in Fig. 4.7(a). The velocity, acceleration, and jerk profiles were differentiated in a non-causal manner [188]. Notably, the demonstration trajectory is slow with a duration of 8.88 seconds, while the presence of noise reflects itself in all the profiles, especially the jerk profile.

After applying the Time Optimization module, we obtain the trajectory $\boldsymbol{q}_t(t)$, with the relative timing law τ_i extracted using (4.16). As shown in Fig. 4.7(b), while this optimization reduces the trajectory duration to 1.17 seconds, the noise remains present, and both acceleration and jerk spikes reach extremely high values, up to approximately 150 rad/s^2 for acceleration and 20000 rad/s^3 for jerk. These values are unrealistic and significantly exceed the kinematic limits, as only the timing law τ_i is modified without altering the underlying path of the demonstration.

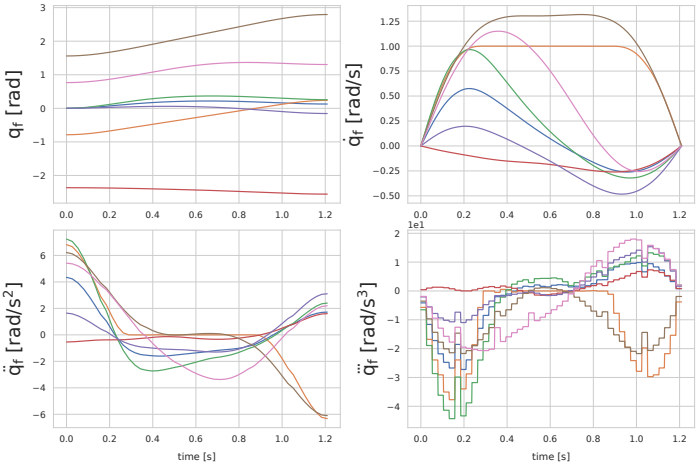
The final trajectory, $\boldsymbol{q}_f(t)$, is generated by applying τ_i along with $\boldsymbol{\epsilon}_p^d$ and ϵ_θ^d in the Trajectory Generation module. This resulted in a smooth, noise-free, and time-optimal trajectory,



(a) Original Demonstration $\boldsymbol{q}_o(t)$.



(b) Time Optimization Output $\boldsymbol{q}_t(t)$.



(c) Trajectory Generation Output $\boldsymbol{q}_f(t)$.

Figure 4.7: Evolution of RT1 from original to smooth time-optimal joint trajectory. The motion profiles including position, velocity, acceleration, and jerk are shown for all 7 joint axes.

depicted in Fig. 4.7(c). The profiles clearly show the removal of path noise, adherence to kinematic limits, and a peak jerk value reduced to around 400 rad/s^3. The trajectory duration was adjusted to 1.21 seconds, slightly longer than the result from the Time Optimization module. This adjustment is due to the extraction and use of the normalized timing law $\tau_i \in [0, 1]$ in the Trajectory Generation module and the inclusion of the term T_f in the second optimization, which finds the optimal timing law while regulating jerk.

In addition to the visual analysis, we compared the original trajectory $\boldsymbol{q}_o(t)$ and the optimized trajectory $\boldsymbol{q}_f(t)$ for RT1-RT5 and MT1-MT5 in Table 4.1 with respect to their execution time and MANJ.

4.3.2.2 Refinement Phase Analysis

To show the performance of the Refinement Phase in locally slowing down the timing of $\boldsymbol{q}_f(t)$ and capturing meaningful tolerance values, we use $\boldsymbol{q}_f(t)$ from RT1 and pass it through the Refinement Phase to slow down the trajectory toward its end. In this experiment, we acquire $C(t)$ through teleoperated feedback. It is expected that the Refinement Phase modifies the timing law of $\boldsymbol{q}_f(t)$ to slow it down proportional to the teleoperated brake command $C(t)$. Moreover, the extracted tolerance bounds are expected to be narrower towards the end of the motion, relative to the intensity $C(t)$. Besides lowering tolerances in these portions, we assign maximum tolerance when $C(t) = 0$, providing more freedom for the Trajectory Generation module to adapt the motion.

Given $C(t)$, the progression of $s(t)$ and $s_r(t)$ are calculated via (4.20) and (4.21) throughout the refinement and shown in Fig. 4.8. Initially, without any command, $s(t)$ and $s_r(t)$ overlap, leaving the timing law unchanged. Upon receiving the command at t^*, $s_r(t)$ progression changes to slow down the motion and thus extend execution time. This revised mapping leads to the derivation of τ_i^r using (4.22). Also, we extract $\boldsymbol{\epsilon}_p^i$ and ϵ_θ^i via (4.23).

With the extracted information, The fine-tuned trajectory $\boldsymbol{p}_f^r(t) = [x_f^r(t), y_f^r(t), z_f^r(t)]^T$ is calculated and shown in Fig. 4.9. In Fig. 4.9(a), The tolerance range is adjusted to be more precise towards the end of reaching, where slow speed was commanded. Moreover, to showcase the velocity adjustment effect on the trajectory's timing law, we present the distri-

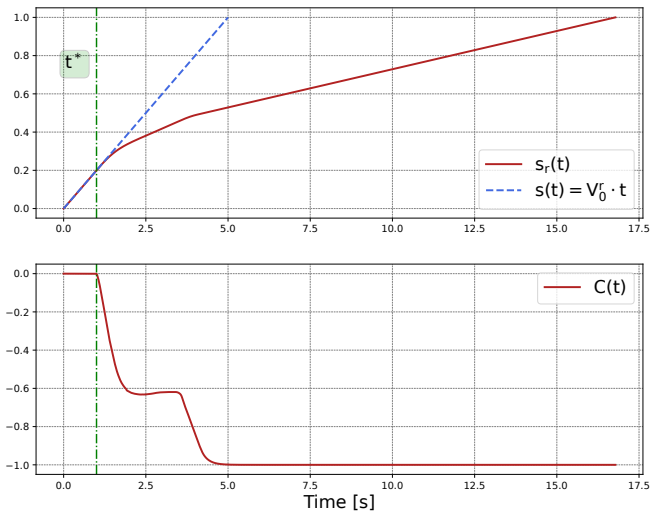
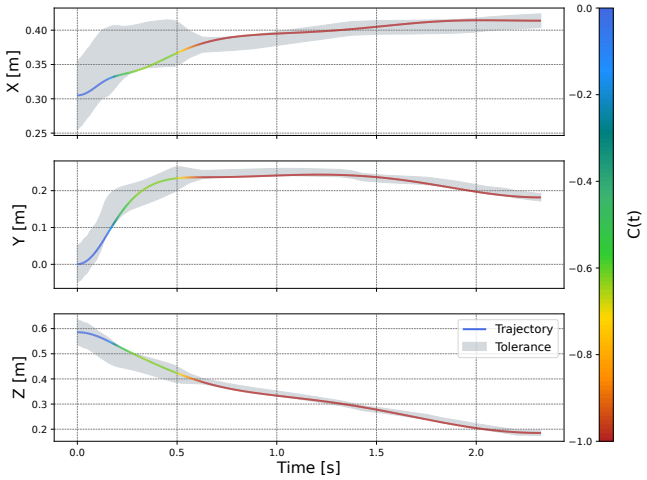


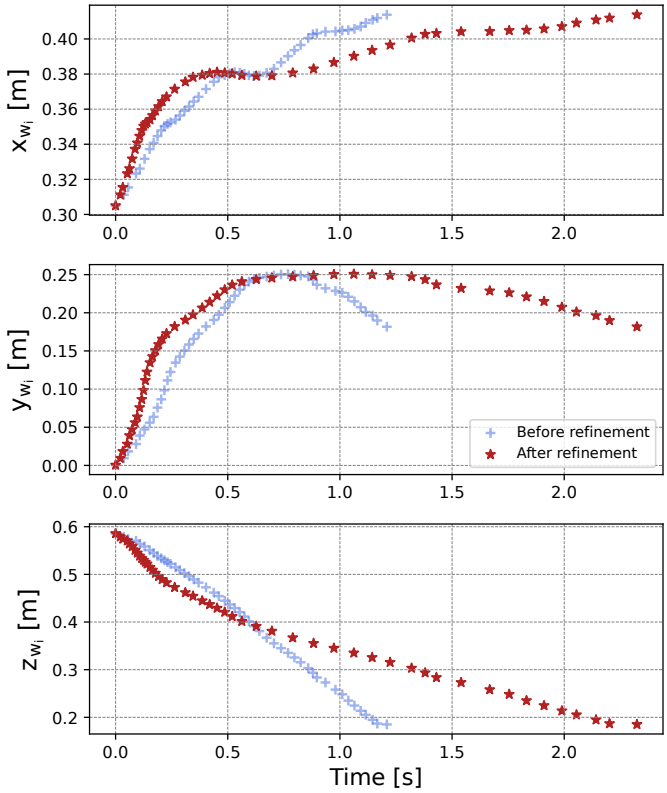
Figure 4.8: $s(t)$ and $s_r(t)$ during the Refinement Phase of RT1. t^* marks the moment the teacher starts giving command $C(t)$.

bution of $\mathbf{p}_{w_i} = [x_{w_i}, y_{w_i}, z_{w_i}]^T$ over the trajectory before and after refinement in Fig. 4.9(b). Evidently, the waypoints are stretched towards the end of the task. It is also noticeable that while the timing law of the beginning of the trajectory is left unchanged during refinement, the waypoint distribution does not overlap. This is due to the fact that since the tolerance values are increased from $\boldsymbol{\epsilon}_p^d$ to $\boldsymbol{\epsilon}_p^{max}$, as in (4.23), the optimizer has managed to find a faster trajectory to traverse these waypoints.

Providing higher tolerance bounds for the optimization will allow the Trajectory Generation module to optimize for a better solution in terms of its objective in (4.17). It means that higher tolerance bounds lead to lower execution time and lower jerk values. To further pinpoint this, we analyze this aspect by optimizing RT2 via different tolerance bounds of 2 cm and 5 cm across all its waypoints. Fig. 4.10 shows the effect of increasing tolerance values in RT2. We can see that via the tolerance bounds of 2 cm, the Optimization-based Smoothing solved for an execution time of 1.87 seconds and a maximum end-effector jerk of 73.55 m/s^3, whereas by increasing the tolerance bounds to 5 cm, the execution time is reduced by 15% to 1.61 seconds, and the maximum end-effector jerk dropped by 50% to 36.55 m/s^3.



(a) $\boldsymbol{p}_f^r(t)$ with tolerance bounds extracted based on $C(t)$



(b) Waypoints timing law before and after refinement.

Figure 4.9: RT1 fine-tuning after the Refinement Phase. As the intensity of $C(t)$ increases, the tolerance bounds are narrowed, and the timing of waypoints is extended accordingly.

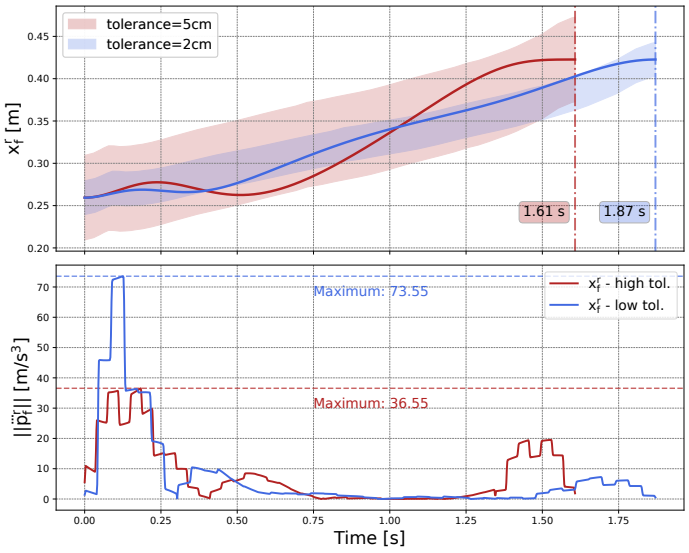


Figure 4.10: Comparison of $x_f^r(t)$ for RT2, considering optimized execution time and maximum end-effector jerk. Increasing the tolerance bounds enables the optimization to identify faster timing laws with reduced jerk profiles.

4.3.2.3 DMP Case Study

To show the benefits of incorporating DFL-TORO as an intermediate layer between the original demonstrations and the DMP algorithm, we train two sets of DMPs: DMP_o using the original trajectory $\boldsymbol{q}_o(t)$ and DMP_f using the optimized trajectory $\boldsymbol{q}_f(t)$. These are then used to reproduce the trajectories $\boldsymbol{q}_{o,dmp}(t)$ and $\boldsymbol{q}_{f,dmp}(t)$, respectively. By applying DFL-TORO prior to DMP training, we anticipate achieving smoother trajectories with a reduced jerk profile. Additionally, by eliminating noise from the original demonstration, we expect DMP_f to produce trajectories that not only adhere to kinematic limits but also have a shorter execution time compared to those generated by DMP_o. This setup highlights the enhanced trajectory quality and efficiency gained by using DFL-TORO in the learning process.

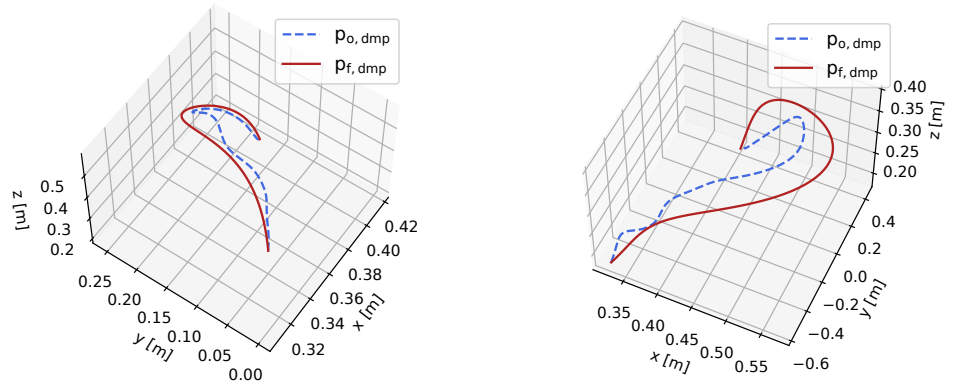
Since the original trajectory $\boldsymbol{q}_o(t)$ is slower than the optimized trajectory $\boldsymbol{q}_f(t)$, we utilize the inherent scaling factor τ from (4.1) in DMP_o to generate $\boldsymbol{q}_{s,dmp}(t)$, ensuring it matches the duration of $\boldsymbol{q}_{f,dmp}(t)$ for a meaningful comparison [148]. In this context, $\boldsymbol{p}_{f,dmp}(t)$, $\boldsymbol{p}_{o,dmp}(t)$, and $\boldsymbol{p}_{s,dmp}(t)$ denote the corresponding Cartesian position trajectories.

Figures 4.11(a) and 4.11(b) illustrate the paths of $\boldsymbol{p}_{o,dmp}(t)$ and $\boldsymbol{p}_{f,dmp}(t)$ for RT1 and

MT1, respectively. As shown, the path of $\boldsymbol{p}_{o,dmp}(t)$ contain unnecessary curves, which are artifacts from fitting over the noise of the original demonstration. Furthermore, the presence of noise results in $\boldsymbol{q}_{s,dmp}(t)$ becoming an infeasible trajectory in both cases, as highlighted by the kinematic violations shown in Figures 4.11(c) and 4.11(d). In contrast, $\boldsymbol{q}_{f,dmp}(t)$ remains feasible within the same execution time. This comparison underscores that removing path noise not only reduces execution time but also leads to a more accurate and feasible timing law, enhancing the overall trajectory quality.

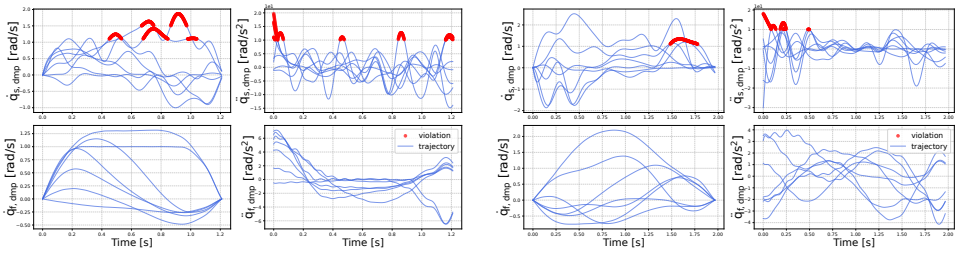
Furthermore, a comparison of the jerk values for $\boldsymbol{q}_{s,dmp}(t)$ and $\boldsymbol{q}_{f,dmp}(t)$ is presented in Fig. 4.11(e) and 4.11(f), showcasing that $\boldsymbol{q}_{f,dmp}(t)$ results in significantly lower jerk values. This improvement is a direct consequence of eliminating noise from the original demonstrations, thereby enhancing the outcomes of the DMP algorithm. Table 4.1 provides a summary of the execution time and jerk values (MANJ) for RT1-RT5 and MT1-MT5. The data clearly indicate that DFL-TORO effectively reduces both execution time and MANJ metrics. This reduction is further verified through the trajectories generated by DMP_o and DMP_f.

In addition to the benefits previously mentioned, it is crucial to note that the Gaussian basis functions used during the training process of DMPs, as described in (4.3), inherently provide a smoothing effect on the demonstrations. This smoothing can act as a filter, mitigating the impact of noise in the original demonstrations. However, this effect alone is often insufficient to completely eliminate noise, as residual artifacts can persist. These residual noise elements can still cause inefficiencies in the reproduced trajectories, as observed with DMP_o.



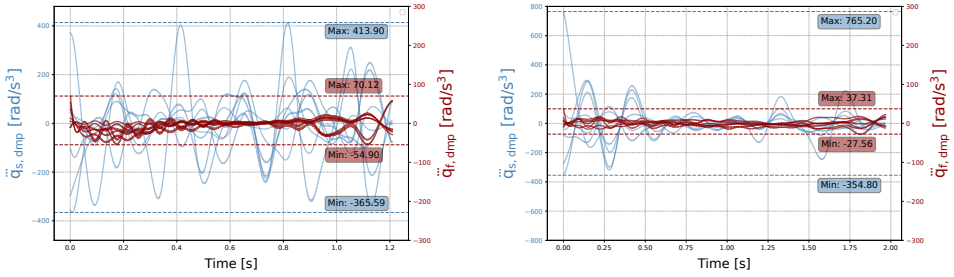
(a) 3D path of RT1.

(b) 3D path of MT1.



(c) Kinematic violations for RT1.

(d) Kinematic violations for MT1.



(e) Jerk profiles for RT1.

(f) Jerk profiles for MT1.

Figure 4.11: Performance comparison of DMP_o and DMP_f with regard to kinematic feasibility and jerk profiles. DMP_f not only surpasses DMP_o in kinematic feasibility but also provides a considerably smoother jerk profile.

Table 4.1: Comparison of time [s] and Maximum Absolute Normalized Jerk (MANJ) [rad/s^3] for RT1-5 and MT1-5.

Experiment	Time/Jerk	Demonstration		DMP	
		$\boldsymbol{q}_o(t)$	$\boldsymbol{q}_f(t)$	$\boldsymbol{q}_{o,dmp}(t)$	$\boldsymbol{q}_{f,dmp}(t)$
RT1	Time	8.88	**1.21**	8.88	**1.21**
	MANJ	15358.07	**78.31**	2305.95	**124.00**
RT2	Time	11.15	**1.87**	11.15	**1.87**
	MANJ	241452.27	**771.83**	2822.72	**582.82**
RT3	Time	11.82	**1.98**	11.82	**1.98**
	MANJ	62706.92	**82.25**	2028.04	**164.84**
RT4	Time	7.08	**1.22**	7.08	**1.22**
	MANJ	9048.28	**97.33**	678.59	**163.01**
RT5	Time	19.75	**2.21**	19.75	**2.21**
	MANJ	101934.88	**207.85**	2993.82	**358.07**
MT1	Time	10.55	**1.97**	10.55	**1.97**
	MANJ	216390.32	**132.40**	5781.34	**284.09**
MT2	Time	14.35	**1.49**	14.35	**1.49**
	MANJ	263475.13	**146.01**	2101.84	**152.09**
MT3	Time	9.82	**1.11**	9.82	**1.11**
	MANJ	22481.38	**115.51**	1366.84	**172.86**
MT4	Time	14.95	**1.31**	14.95	**1.31**
	MANJ	97762.75	**353.56**	10462.93	**315.19**
MT5	Time	8.08	**0.95**	8.08	**0.95**
	MANJ	23539.90	**102.12**	1533.07	**115.79**

Chapter 5

LfD Software Framework

This chapter introduces the proposed LfD software framework, providing a comprehensive high-level overview of its architecture and functionality. The framework is designed to enable non-expert users, such as process engineers, to program robotic systems intuitively without requiring prior knowledge of robotics.

5.1 High-level Overview

The key components of the software, their roles, and interactions are explained in detail, as depicted in Figure 5.1. The software modules are represented as rectangles, while interacting elements are shown as circles. Each element and module of this architecture is explained as follows:

- **User:** The primary actor in this framework is a non-expert user who interacts with the system to program the robot.
- **LFD Recorder:** This module records demonstrations provided by the user, capturing the robot's joint configurations and storing the demonstration trajectories for subsequent processing.
- **LFD Smoothing:** This module is the implementation of DFL-TORO, which refines raw demonstrations, transforming them into optimal trajectories suitable for deployment.

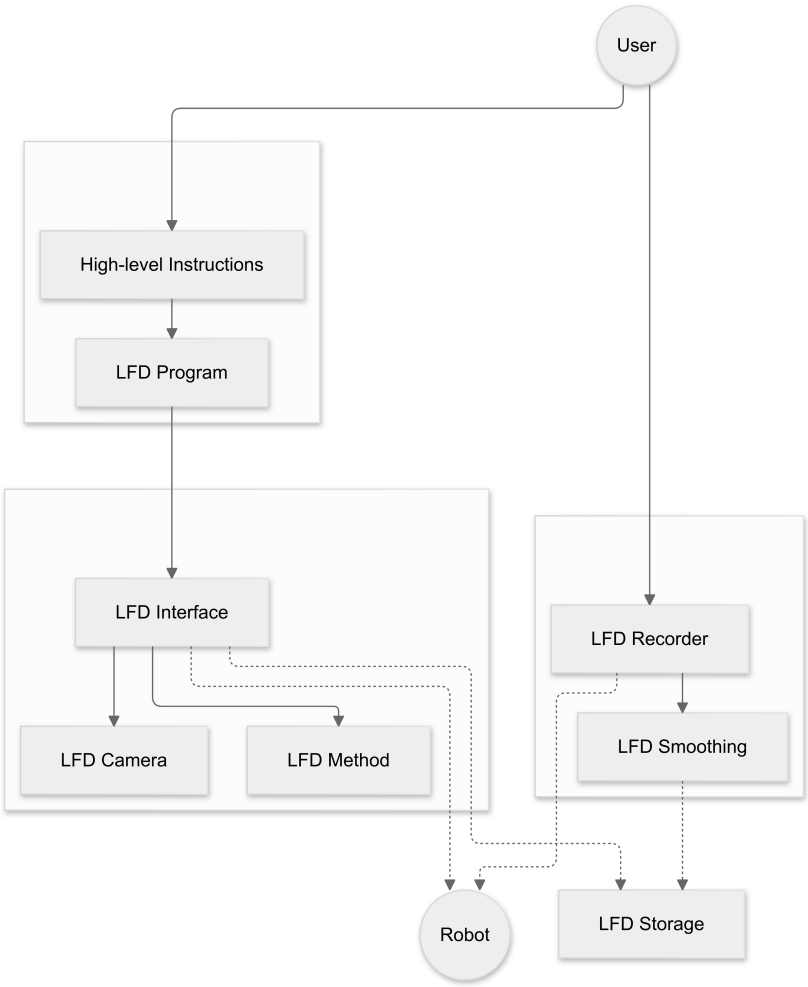


Figure 5.1: High-level architecture of the developed LfD software framework. The software modules are shown in rectangles, while the interacting elements are shown via circles.

- **LFD Interface:** Serving as the core of the framework, this module provides an interaction interface between the user and other software modules. It is designed to enhance modularity and extensibility by standardizing interactions.
- **LFD Method:** This module implements the desired LfD method, which is responsible for training on the recorded demonstrations and planning trajectories based on the required goal configurations.
- **LFD Camera:** This module interfaces with the perception system, enabling robot-camera coordination and fetching the pose of objects using integrated cameras.
- **LFD Program:** Acting as a wrapper for the entire framework, this module enables the user to program the robot using high-level instructions. These intuitive instructions allow users to assemble subtasks into complete tasks while controlling object perception and gripper functionalities.
- **LFD Storage:** This module handles the storage of all demonstration data, making it accessible on-demand across various components.

The full lifecycle of task implementation within the LfD software framework involves the following steps:

1. The user demonstrates the required subtasks through the LfD Recorder module. Each demonstration is named and stored uniquely. This process is repeated for all subtasks.
2. The LFD Smoothing module processes the recorded demonstrations to generate optimal trajectories, which are then stored for deployment.
3. The user creates high-level instructions to specify how the subtasks are combined to form the complete task. These instructions also define object aliases and gripper behaviors.
4. The LFD Program module executes the high-level instructions sequentially by interfacing with other modules via the LFD Interface module.

5. The LFD Interface module enables communication between the LFD Program and other modules, such as the LFD Camera for object localization, the LFD Method for motion planning, and the robotic system for task execution.

It is worthy to mention that the task implementation lifecycle is inherently iterative. The user can provide demonstrations, observe the robot's behavior during execution, and refine the instructions or demonstrations as needed. This iterative approach ensures that the robot's performance is optimized until the desired behavior is achieved.

5.1.1 Key Design Choices

The development of the LfD software framework was guided by several key design choices aimed at enhancing standardization, modularity, extensibility, and scalability. These choices ensure that the framework is adaptable to a wide range of robotic platforms and capable of integrating various methods and hardware components seamlessly. The following frameworks were incorporated to achieve the desired goals:

- **ROS:** ROS [13] serves as the middleware framework for the software, offering a standardized communication interface between different modules. By leveraging ROS, the framework benefits from an established and widely adopted ecosystem that supports modular component integration. ROS provides Standardized message passing and service calls, enabling seamless communication between heterogeneous subsystems. It also simplifies module or hardware integration, enhancing the adaptability and extensibility of the system. Moreover, ROS provides access to an extensive library of pre-existing tools and packages, accelerating development and ensuring compatibility with standard robotics platforms. The use of ROS fosters code reusability across various robotics applications and ensures the system remains scalable and flexible for future enhancements.

- **MoveIt:** MoveIt [14], an open-source software for robotic motion planning and manipulation, was integrated into the framework to achieve a robot-agnostic architecture. By

decoupling high-level motion planning and manipulation logic from hardware-specific implementations, MoveIt provides seamless application across different robotic platforms without requiring modifications to the core logic. It also includes kinematics, collision checking, and path planning functionalities, enabling flexible and efficient operation with various robotic arms. This abstraction simplifies adaptation to new robotic systems and greatly improves the framework's flexibility, making it suitable for a wide range of robotics applications.

5.1.2 Modularity

The framework's design emphasizes extensibility and modularity in two key areas:

- **Robotic Platform:** The integration of MoveIt enables the software to maintain a robot-agnostic architecture. This decoupled structure ensures that the framework can be extended to support any robotic platform with minimal effort, making it highly adaptable to diverse hardware configurations. The current implementation of the software includes integration with FR3 as well as ABB dual-arm Yumi.
- **LfD Method:** The framework is designed to accommodate arbitrary LfD methods. Its modular architecture allows for the implementation and integration of alternative LfD approaches, ensuring flexibility and adaptability for future advancements in the field. The current implementation of the software includes the integration of DMP as the LFD Method module.

5.2 Core Features and Modules

5.2.1 LFD Interface

The LFD Interface is designed as a modular system that facilitates the process of recording, training, and executing robot movements based on human demonstrations. The system is composed of several key modules that work together to provide a comprehensive LFD pipeline, depicted in Figure 5.2. The core features and modules are designed as follows:

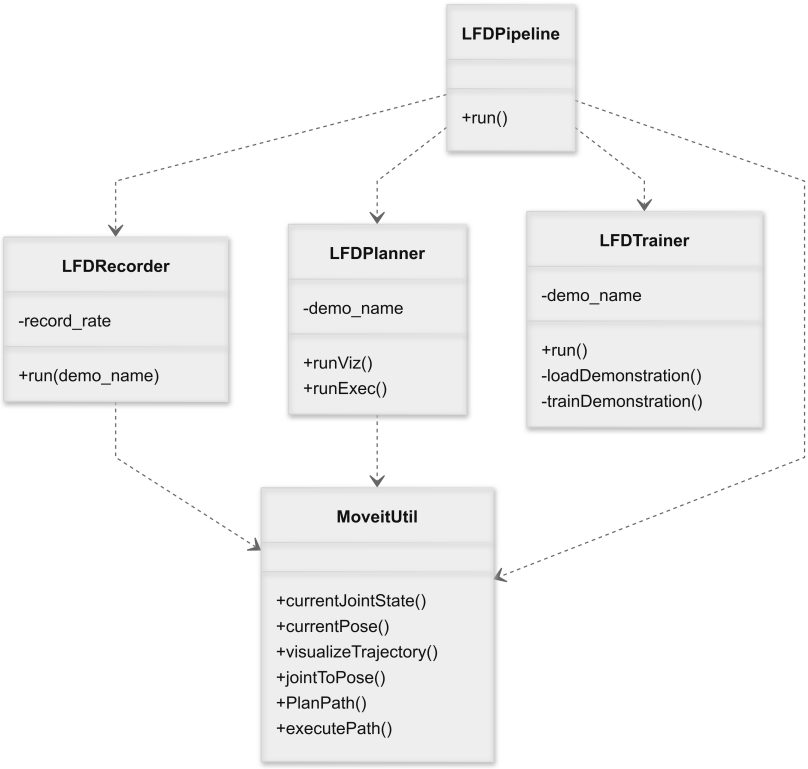


Figure 5.2: The LFD Interface class diagram.

- **LFD Pipeline:** This is the main orchestrator of the entire LfD process. It coordinates the interactions between different modules and manages the overall workflow. The LFD Pipeline class initiates the entire LfD process from demonstration recording to planning and execution.
- **LFD Recorder:** This module handles the recording of demonstration data. Its main functionalities include detecting when the robot starts moving to initiate recording, capturing the robot's joint states at specified intervals, visualizing the recorded trajectory in real-time, and storing the demonstration data for later use.
- **LFD Trainer:** Responsible for training the LfD model using the recorded demonstrations. The LFD Trainer module retrieves demonstrations stored in the LFD Storage and trains the LfD model with the loaded demonstration data, preparing it for trajectory planning and execution.
- **LFD Planner:** This module generates and executes trajectory plans based on the trained LfD model. Key features include interfacing with the LFD method to obtain trajectory plans, Visualizing planned trajectories, and executing the planned trajectories on the robot.
- **MoveitUtil:** A utility module that interfaces with MoveIt functionalities, providing real-time access to the robot's current state, visualization of trajectories and motion paths, and Moveit's internal path planning and execution capabilities.

The designed workflow of the LFD Pipeline is illustrated in Figure 5.3 and follows these steps:

1. **Initiating the Pipeline:** The user initiates the desired behavior—recording, training, or executing motions—via the LFD Pipeline. Communication between modules is facilitated through ROS Actions, ensuring the system remains non-blocking while the robot executes tasks.
2. **Recording Demonstrations:** The LFD Recorder module monitors the robot and begins recording as soon as it detects movement. Joint states are recorded through the

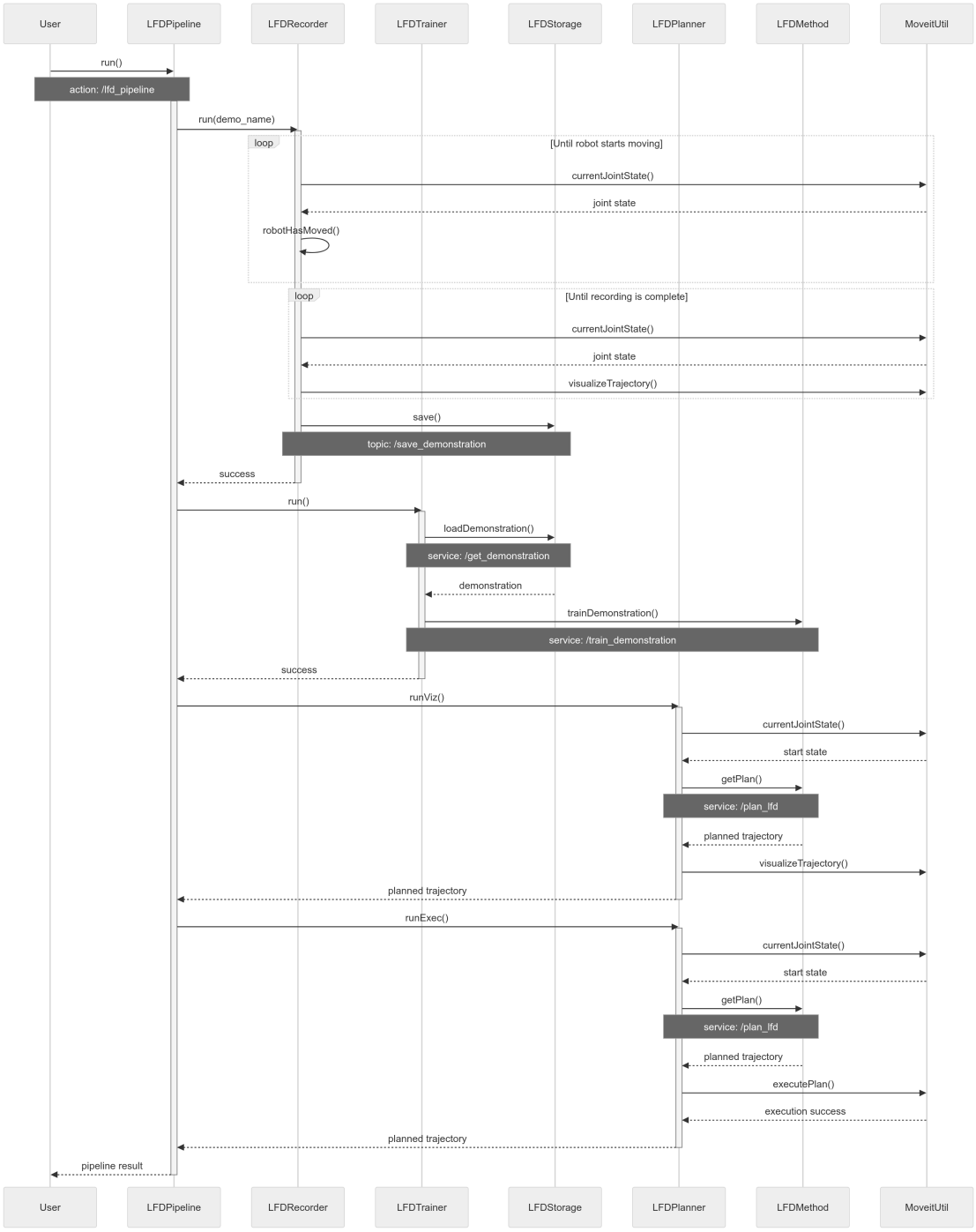


Figure 5.3: The LFD Interface sequence diagram.

MoveItUtil module and visualized in real-time. The completed demonstration is published via a ROS Topic and saved under a unique name in the LFD Storage module.

3. **Training Demonstrations:** The LFD Trainer retrieves the desired demonstration from the LFD Storage via a ROS Service. The demonstration data is transferred to the LFD Method (e.g., LFD DMP) via another ROS Service for training. The trained model is stored for subsequent use in planning and execution. The standard ROS communication with LFD Method, as highlighted in Figure 5.3, confirms the modularity and extensibility of this module as well.

4. **Planning and Executing Trajectories:** The LFD Planner fetches the robot's current state using the MoveItUtil module, setting it as the starting state for planning. A planning request, including the goal configuration and duration, is sent to the LFD Method via a ROS Service. The planned trajectory is either visualized for validation or directly executed on the robot.

This comprehensive pipeline ensures that the entire LfD process—from capturing demonstrations to executing complex tasks—is modular, extensible, and robust, meeting the needs of both research and practical applications.

5.2.2 LFD Method

The LFD Method module serves as the core component for learning and generalizing robot behavior based on demonstration data. This module is designed as an abstract interface, enabling the framework to support various LfD techniques. The key responsibilities of LFD Method are as follows:

- **Training Demonstrations:** The LFD Method module processes demonstration data received from the LFD Trainer module. It applies learning algorithms, such as DMPs, to extract motion patterns and encode them into reusable models. This responsibility is through a standard ROS service as shown in Figure 5.3.

- **Trajectory Generalization:** Using the trained model, the module generates trajectories that generalize the demonstrated motion to new start and goal configurations. This responsibility is through a standard ROS service as shown in Figure 5.3.

In this research, DMPs are used for case study. Therefore, the abstract implementation of the LFD Method module is replaced by the implementation of DMPs.

5.2.3 LFD Smoothing

The LfD Smoothing module is responsible for processing and optimizing recorded demonstrations to produce refined and optimal trajectories. This module incorporates the core implementation of DFL-TORO and integrates several components to filter, optimize, and smooth demonstration data effectively. The module's architecture, depicted in Figure 5.4, ensures that demonstration trajectories are optimized for accurate and efficient execution. The core components of LFD Smoothing are as follows:

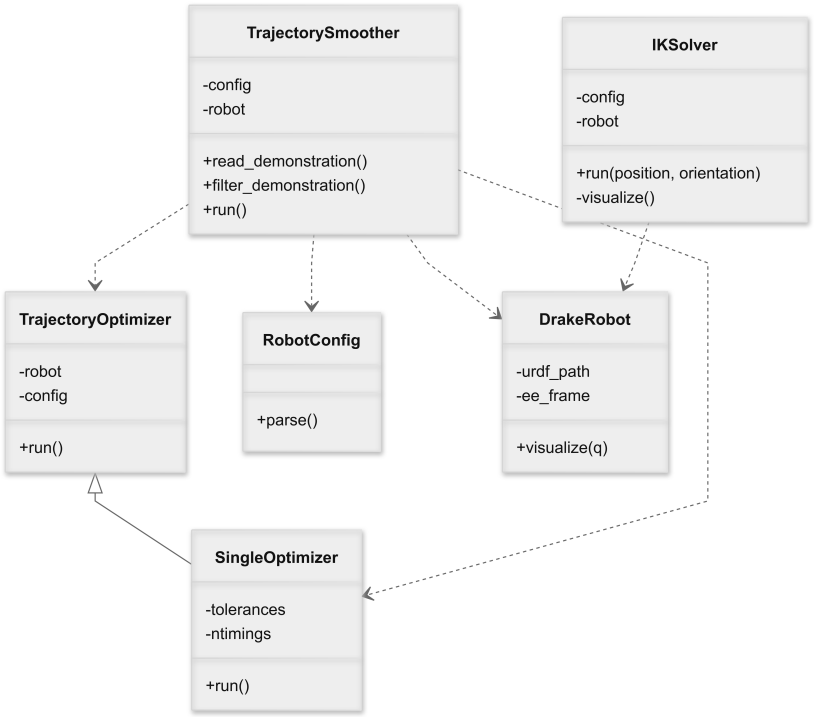


Figure 5.4: The LFD Smoothing class diagram.

- **Trajectory Smoother:** The central class that orchestrates the smoothing process. Key functionalities include loading recorded demonstration data, applying initial filtering to clean the data, and executing the complete smoothing pipeline.
- **Trajectory Optimizer:** Implements the time optimization process, refining the timing of the demonstration trajectories as detailed in Section 4.2.
- **Single Optimizer:** An extension of the Trajectory Optimizer, this class performs trajectory generation using the timing law extracted during time optimization. It ensures that the final trajectory adheres to specified tolerance values.
- **Robot Config:** A utility class that handles the parsing and management of robot configuration parameters required for the optimization process.
- **Drake Robot:** Represents the robot model used in the smoothing process, implemented via PyDrake [187]. This class integrates the robot's URDF with the PyDrake library and provides tools for visualizing optimization results. It also manages the designated end-effector frame for the trajectory.
- **IK Solver:** A utility class for computing inverse kinematics (IK) solutions. Based on PyDrake, this solver calculates joint configurations for a given end-effector pose.

The LfD Smoothing process follows a structured workflow, as illustrated in Figure 5.5, to generate smooth and optimal trajectories from recorded demonstrations:

1. The user initializes the Trajectory Smoother with the desired configuration for the selected demonstration.
2. The Trajectory Smoother loads the demonstration data and applies initial filtering to clean and prepare it for optimization.
3. Instances of DrakeRobot and RobotConfig are created based on the robot model and the provided configuration parameters.

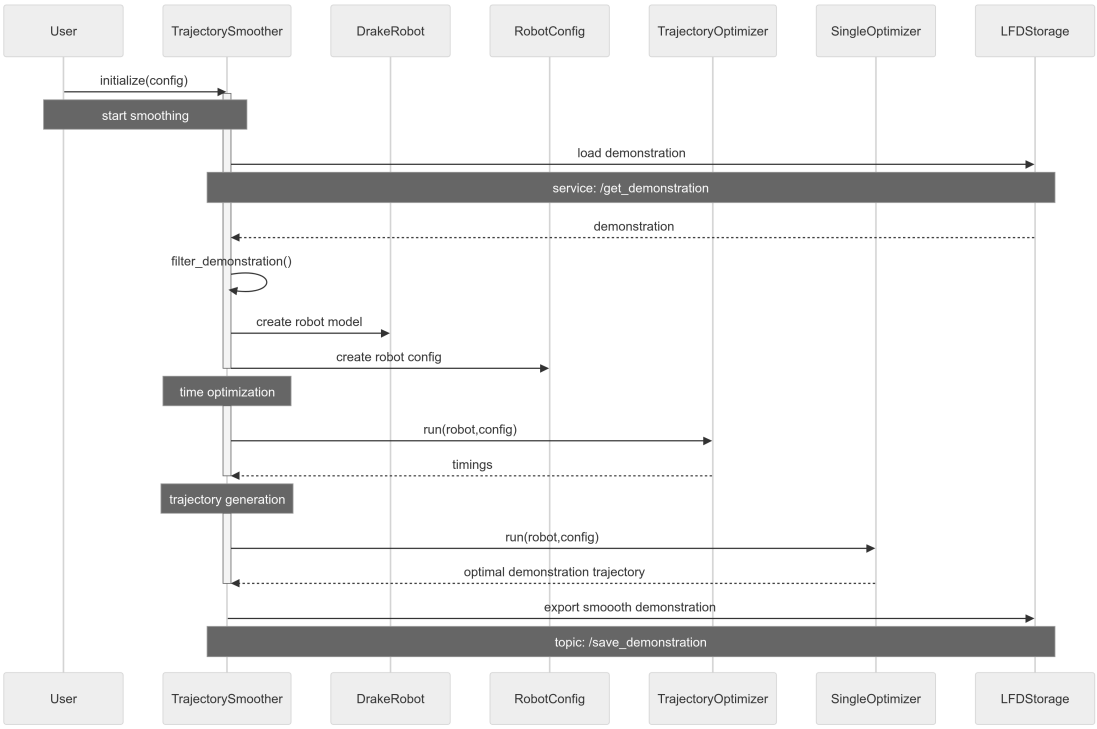


Figure 5.5: The LFD Smoothing sequence diagram.

4. The Trajectory Optimizer refines the timing of the demonstration trajectory, ensuring smooth transitions and efficient execution.

5. The Single Optimizer generates the final, optimal demonstration trajectory based on the output of the time optimization step.

6. The optimal demonstration trajectory is exported to the LfD Storage module, making it available for subsequent training or execution.

5.2.4 LFD Camera

The LFD Camera module is designed to handle camera integration and object localization within the LfD workflow. This module provides a comprehensive solution for integrating perception systems with robotic manipulators. The core features and components of this module are shown in Figure 5.6 and described as follows:

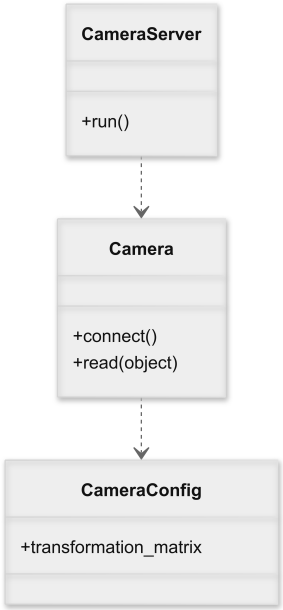


Figure 5.6: The LFD Camera class diagram.

- **Camera:** An abstract base class representing a generic camera interface with methods responsible for establishing a connection to the camera and reading the pose of a

specified object.

- **Camera Server:** Acts as the intermediary between the LfD workflow and the camera. It provides a unified interface for managing camera operations, ensuring seamless integration with other modules.
- **Camera Config:** Stores and manages camera configuration parameters, including transformation matrix and additional parameters such as symmetry parameters or angle offset values for pose calculations.

The workflow, as illustrated in Figure 5.7, follows these steps:

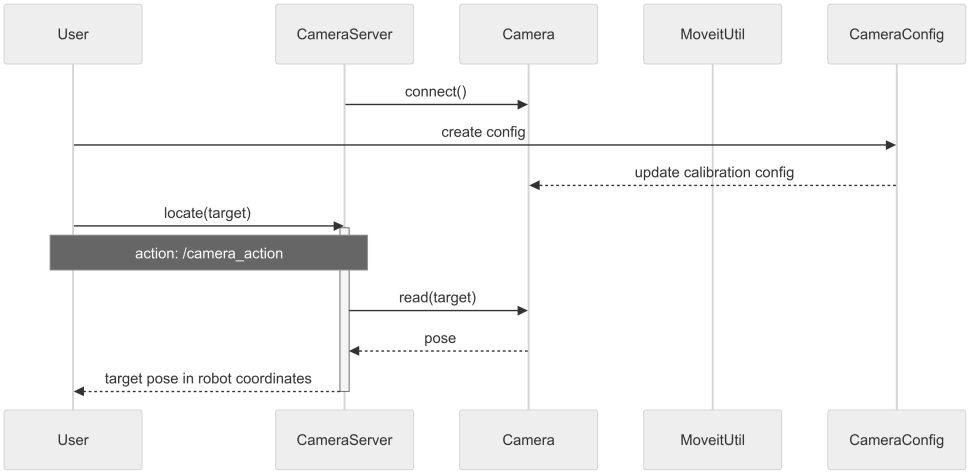


Figure 5.7: The LFD Camera sequence diagram.

1. The Camera Server establishes a connection with the camera hardware, initializing communication.
2. The user creates a new Camera Config, specifying the calculated transformation matrix and any additional parameters. This configuration is then updated in the Camera class to ensure accurate pose estimation.
3. When the user requests object localization, the Camera Server processes the request by interacting with the camera. The pose of the target object is calculated and returned in the robot's coordinate system, ready for use in the LfD workflow.

5.2.5 LFD Program

The LFD Program module serves as a high-level interface that integrates all core functionalities of the LfD framework, including the LFD Interface, LFD Method, and LFD Camera modules. It provides a centralized system for orchestrating the execution of learned demonstrations and composing advanced robotic tasks through a set of high-level instructions. As part of the modular design, DMPs are developed as the abstract LFD Method, serving as a foundational example of how any arbitrary LfD Method can be integrated into the LFD Program. The architecture of this module is shown in Figure 5.8, and its core components are described below.

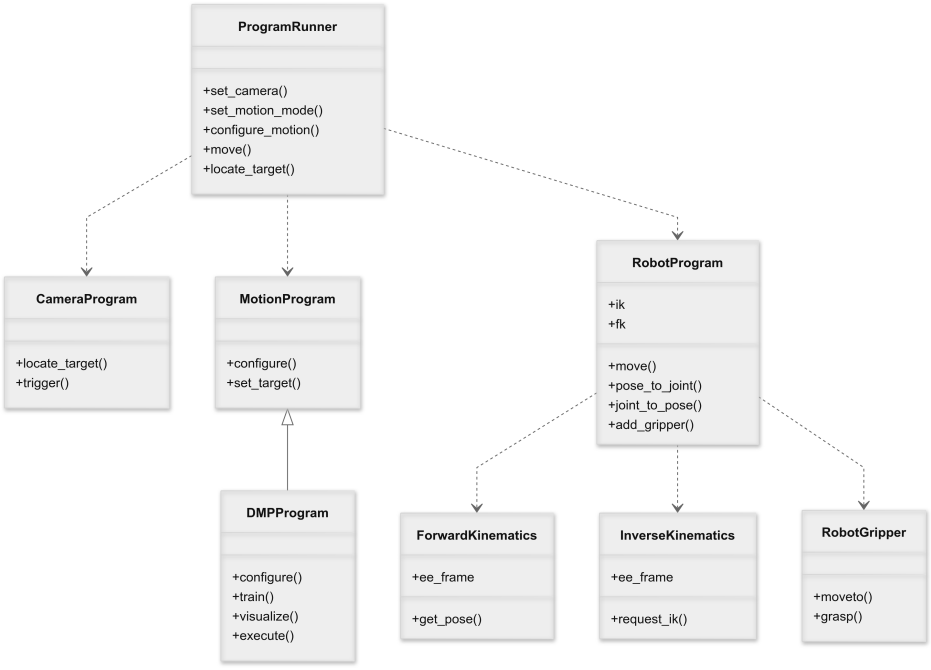


Figure 5.8: The LFD Program class diagram.

- **Program Runner:** The main class responsible for coordinating the execution of robotic programs. It includes functionalities for initializing the camera system, setting and configuring motion parameters, locating target objects through the camera interface, and executing planned motions on the robot.

- **Robot Program:** An abstract class that provides core utilities for robot control, including forward and inverse kinematics for converting between joint configurations and end-effector poses, executing planned trajectories on the robot, and commanding the robot's gripper for grasping tasks.
- **Motion Program:** An abstract interface for implementing different motion planning strategies. It provides methods for configuring motion parameters and defining goal configurations for planning.
- **DMP Program:** A concrete implementation of the Motion Program, specifically designed for DMPs. It interacts with LFD DMP functionalities to enable configuration of DMP parameters, training and planning based on demonstrations, and visualization and execution of planned trajectories. DMP Program serves as an example to show how other LfD Methods can be similarly developed and integrated into the LFD Program workflow.
- **Camera Program:** Handles operations related to the perception system, including triggering the camera to detect specific objects, localizing target objects and retrieving their poses in the robot's coordinate system.
- **Robot Gripper:** An abstract class defining gripper operations, such as moving the gripper to a specific position and performing grasping actions.
- **Forward Kinematics and Inverse Kinematics:** Utility classes for performing forward and inverse kinematics calculations to map between joint configurations and end-effector poses.

The workflow, as illustrated in Figure 5.9, follows these steps:

1. The Program Runner is initialized with the specific robot, motion mode (e.g., DMP), and the camera system.
2. The user configures motion parameters, which are passed to the DMP Program for further processing.

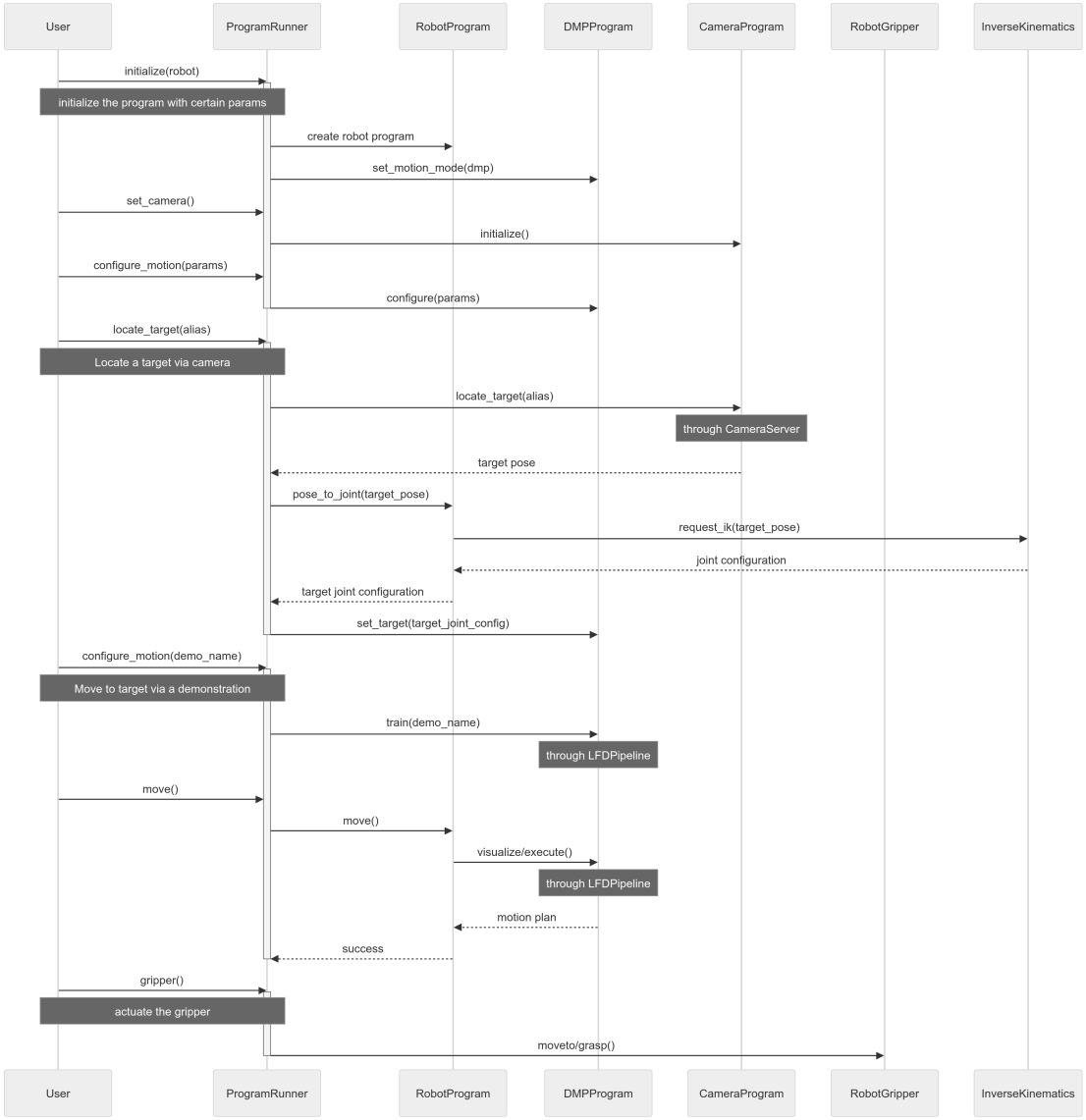


Figure 5.9: The LFD Program sequence diagram.

3. The user requests the localization of a target object. The Program Runner invokes the Camera Program to detect and retrieve the object's pose. The pose is then converted into a joint configuration using inverse kinematics.

4. The retrieved joint configuration is assigned as the goal for trajectory planning in the DMP Program.

5. The user specifies a demonstration for training. The DMP Program trains the motion model based on the demonstration and plans a trajectory toward the goal configuration. The user initiates the execution of the planned motion through the Robot Program and DMP Program.

6. The user can control the robot's gripper through the Program Runner, which interfaces with the Robot Gripper module for grasping and manipulation tasks.

5.2.6 High-Level Instructions

The LfD software framework offers a set of intuitive high-level instructions that allow users to easily control and program a robotic system, even without specialized robotics knowledge. These high-level instructions provide an abstract interface, enabling users to compose, modify, and execute complex tasks using simple commands. The key purpose of these instructions is to enable a process engineer, or any user with minimal robotics experience, to break down a task into manageable subtasks and specify the sequence of actions the robot must perform to complete the entire task.

High-level instructions are structured to provide the user with control over key aspects of robot operation, such as motion, object manipulation, and task sequencing. These instructions are designed to be easy to use while allowing the user to customize various parameters for specific task needs. Key components of high-level instructions include:

- **Motion Configuration:** This instruction allows the user to specify how the robot should move. The user can instruct the robot to move according to a given demonstration.

The configuration instruction supports various levels of customization, such as specifying the demonstration name (demo_name) or adjusting the movement duration (duration_scale). The user simply refers to pre-recorded demonstrations by name and the software handles the execution. Here is an example for providing such configuration:

```
configure_motion(demo_name,duration_scale)
move()
```

- **Target Object Localization:** The framework also provides the ability to locate objects in the robot's environment using a camera system. This command allows the user to specify a target object, and the system automatically integrates the object's location into the robot's planned trajectory, i.e., the specified demonstration is used to plan a trajectory to reach the target object. The alias of the object refers to the pre-configuration object in the camera system and signals the camera how to locate the object. Here is an example of how this instruction is used:

```
configure_motion(demo_name)
locate_target(object_alias) # Locate the object and set as target
move()
```

- **Gripper Control:** Beside the robot's motion, controlling the end-effector's gripper is also critical to enable robotic manipulation. High-level instructions enable the user to control the gripper's actions, including opening, closing, or moving to a predefined position. The user can issue simple commands to instruct the gripper to perform the required operations during task execution. Here is an example of how these instructions can be used:

```
gripper.moveto(4)           ## Move the fingers to position 4
gripper.grasp(close/open)   ## Perform grasping inwards our outwards
```

5.2.6.1 Creating Complex Tasks with High-Level Instructions

The main ability of the high-level instructions is to chain together multiple robot actions to form complete tasks. Users can specify a series of actions that the robot must perform in sequence. The task-creation process typically follows these steps:

1. **Configuring and executing motions:** The user specifies the trajectory or demonstration for the robot to follow. Each movement corresponds to a specific subtask, such as picking or placing an object, which the robot executes sequentially.
2. **Localizing target object:** If a specific object needs to be manipulated, the user can leverage the camera system and specify the object's alias to locate the object's coordinate. These coordinates are then used to plan the motion for reaching the object.
3. **Gripper manipulation:** During the task, the user can issue commands to open, close, or move the gripper, enabling object manipulation along with the motion execution.
4. **Iterative refinement:** The user can iteratively refine the task by providing feedback or modifying instructions until the robot's behavior is satisfactory.

For example, to perform a pick-and-place task, the user might issue the following sequence of instructions:

```
configure_motion(demo_name="pick_object_X")
locate_target(X)
move()
gripper.grasp(close)
configure_motion(demo_name="place_object_X")
move()
gripper.moveto(10)
```

Chapter 6

Industrial Case Study

In this chapter, an industrial case study is presented to validate the research contributions of this thesis. Section 6.1 provides a detailed overview of the industrial use case, outlining the manufacturing process. Section 6.2 applies the proposed roadmap to the use case, defining the scope of demonstration, the demonstrated mechanism, and the learning mechanism. In Section 6.3, the proposed DFL-TORO methodology is validated by showcasing its effectiveness in automating the sub-assembly process in a real manufacturing environment. Finally, the chapter concludes in Section 6.4 with the implementation of the LfD software framework to achieve full automation of the industrial use case using an LfD-based solution.

6.1 Overview

Rotarex S.A. is a world-leading producer of high-quality gas products made up of more than 100 components and a considerable number of sub-assembly steps. All the products have several variants with slightly different designs and assembly steps. This leads to “mass customization” with a large number of possible products but comparatively small batch sizes.

The required flexibility of the assembly is currently mainly achieved by the cognitive capabilities of human workers to perform different tasks and to adapt their actions quickly to different products. However, humans need to be well-trained and highly focused for a long time. Also, the high variation of speed and quality across different team members limits the

overall productivity. Therefore, a higher degree of automation of production is needed and the application of robots for flexible production is considered a key asset.

The existing robotic setup at Rotarex is ABB YuMi, a dual-arm collaborative robot, shown at the production site in Figure 6.1. The ABB YuMi features a dual-arm configuration with 7-DoF per arm, designed for precise and collaborative tasks in industrial settings. In the previous setup of Yumi in the production site, only one sub-assembly had been automated with a traditional programming mindset. The case study of this thesis is the transformation of this sub-assembly to an LfD-based solution, thanks to the contributions of this thesis.

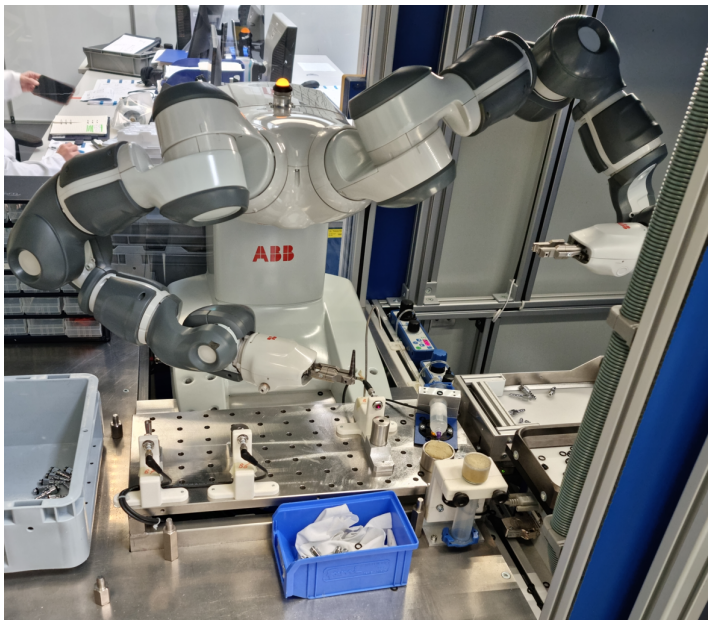
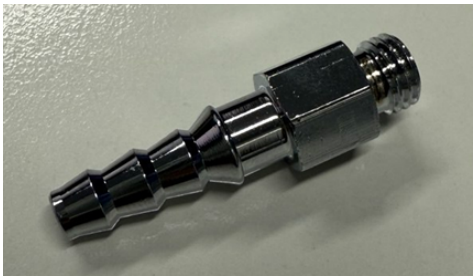


Figure 6.1: The ABB Yumi setup at the production site.

Illustrated in Figure 6.2, the sub-assembly process involves the installation of the O-ring (Figure 6.2(c)) onto the fitting (Figure 6.2(a)) using the auxiliary cone to ensure proper placement. The O-ring is first slid over the narrow end of the auxiliary cone (Figure 6.2(b)) and guided along the taper until it reaches the wider end. The fitting is then aligned with the wide end of the cone, and the O-ring is carefully pushed over the fitting and into its groove, leading to the final sub-assembly product (Figure 6.2(d)).



(a) Fitting.

(b) Auxiliary cone.

(c) O-ring.

(d) Final sub-assembly product.

Figure 6.2: Illustration of use-case sub-assembly steps.

6.2 Applying the Practical Roadmap

This section outlines the application of the roadmap's implementation stages—What to Demonstrate, How to Demonstrate, and How to Learn—in automating the industrial sub-assembly task using LfD.

6.2.1 What to Demonstrate

The sub-assembly task at hand is inherently challenging and complex, as evidenced by the significant training required for human workers to become skilled enough to perform it reliably. Given these complexities, the roadmap's recommendation to use subtask demonstrations instead of full-task demonstrations is particularly relevant. By focusing on individual subtasks, the robot can learn precise and reliable behaviors for each phase of the process, reducing the risk of errors during task execution.

Although the task involves contact—sliding the plastic ring onto the metallic fitting—the primary factor for success is the accurate demonstration and generalization of motion. The contact aspect is straightforward and well-structured, meaning that compliance is not a critical requirement for task success. Instead, a motion-based demonstration suffices, as it ensures the ring slides into place effectively when the motion is executed correctly.

6.2.2 How to Demonstrate

Given the lack of physical demands in this task and the minimal safety concerns associated with humans operating near the robot, kinesthetic teaching is a suitable demonstration mechanism. This method allows for intuitive, direct interaction with the robot to demonstrate complex motions while requiring minimal setup. Kinesthetic teaching is especially suitable for motion-based tasks, as it enables precise trajectory demonstrations without the need for additional equipment or significant technical expertise.

6.2.3 How to Learn

For this industrial use case, both joint space and Cartesian space learning methods were considered. Each has its advantages and trade-offs. Joint space offers lower implementation effort and is directly compatible with the robot's control system, making it easier to implement. Cartesian space provides a more intuitive representation of the task for human understanding and potentially better generalization but requires additional transformations for integration into the robot's control system.

Considering the need for a practical and straightforward implementation, DMPs are selected as the learning algorithm. DMPs are particularly suitable for their ability to perform one-shot demonstrations, high explainability, and low implementation effort, especially in joint space. While DMPs are not the only viable algorithm, they offer a reliable starting point for this task.

6.3 Applying DFL-TORO

This section describes the application of DFL-TORO to automate the industrial sub-assembly process, detailing the experimental setup used for implementation and validation.

6.3.1 Experimental Setup

The ABB Yumi is controlled using the ”abb robot driver” software stack developed by the ROS Industrial community [189]. This software facilitated interaction with ABB's internal controller, enabling state monitoring and providing position or velocity commands. To record demonstrations, YuMi operated in ”Lead-Through” mode, allowing manual manipulation of its arms. The ROS interface controlled YuMi during trajectory execution using position commands at a control frequency of 200 Hz. The rest of the experimental setup is similar to the validation experiments explained in Section 4.3.

6.3.2 Validation Experiments

The experiments with the ABB YuMi include automating a part of the sub-assembly conducted at a production site. This task involved using both arms of the ABB YuMi to assemble a component of a larger product. In these experiments, the sub-assembly is partially automated using DFL-TORO and DMPs. Figure 6.3 illustrates the steps of the subtasks automated via LfD, designated as YM1-YM4. The details of these tasks are described as follows:

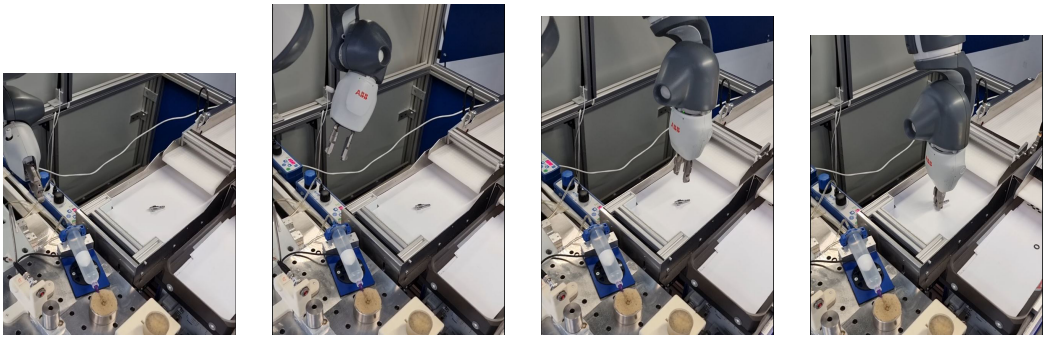
1. **YM1**: The left arm reaches for the metallic object to grasp it from the tray (Fig. 6.3(a) to Fig. 6.3(d)). The fitting's location is detected by an overhead camera and transformed into the robot's coordinate system. These coordinates are then converted into the desired joint configuration, which serves as the goal configuration for the DMP. Using this information, YM1 generalizes the trajectory to grasp the object.

2. **YM2**: The left arm moves the fitting to its desired location, releasing it into its mounting station (Fig. 6.3(e) to Fig. 6.3(h)).

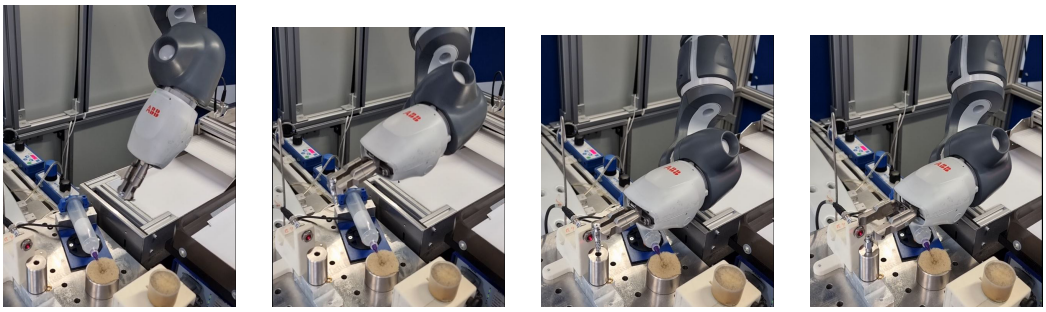
3. **YM3**: The left arm reaches for the O-ring on the other tray, grasping it from the inside (Fig. 6.3(i) to Fig. 6.3(l)). Similarly, the location of the ring is detected via the overhead camera.

4. **YM4**: The left arm moves the ring on top of the auxiliary cone, held by the right arm, releasing it to fall on the cone (Fig. 6.3(m) to Fig. 6.3(p)). From here, the right arm mounts the cone on top of the metallic object and inserts the ring onto it.

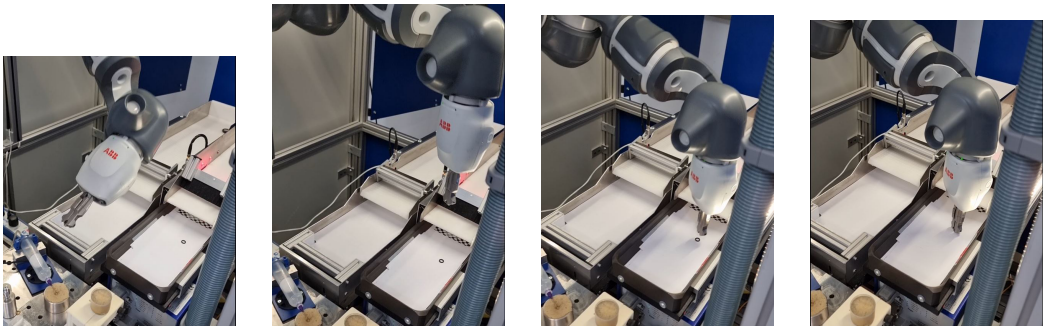
The task specifications of YM1-YM4 present a complex and precise manufacturing task in a small-part assembly process. Fig. 6.4(a) to 6.4(d) show the recorded demonstration trajectories. It is important to note that due to safety measures, the lead-through mode of industrial robots such as ABB YuMi inherently produces more noise in recorded demonstrations compared to the gravity compensation mode used in the FR3. This is because, in Lead-through mode, the robot's movements are guided by joint torque sensors that detect



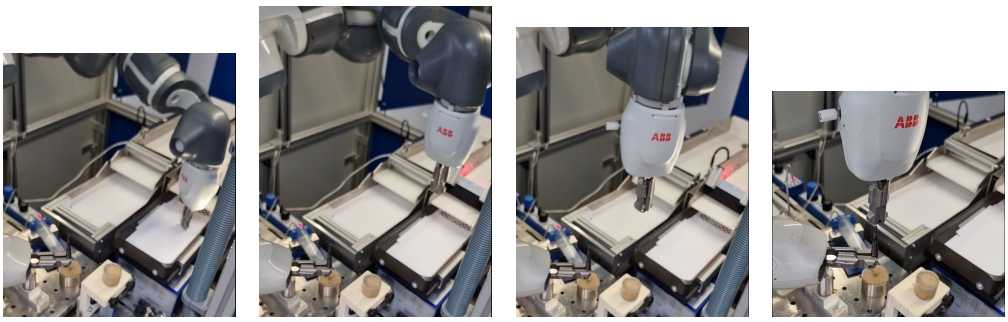
(a) YM1-1 (b) YM1-2 (c) YM1-3 (d) YM1-4



(e) YM2-1 (f) YM2-2 (g) YM2-3 (h) YM2-4



(i) YM3-1 (j) YM3-2 (k) YM3-3 (l) YM3-4



(m) YM4-1 (n) YM4-2 (o) YM4-3 (p) YM4-4

Figure 6.3: Visualization of YuMi tasks YM1 to YM4 in the production site.

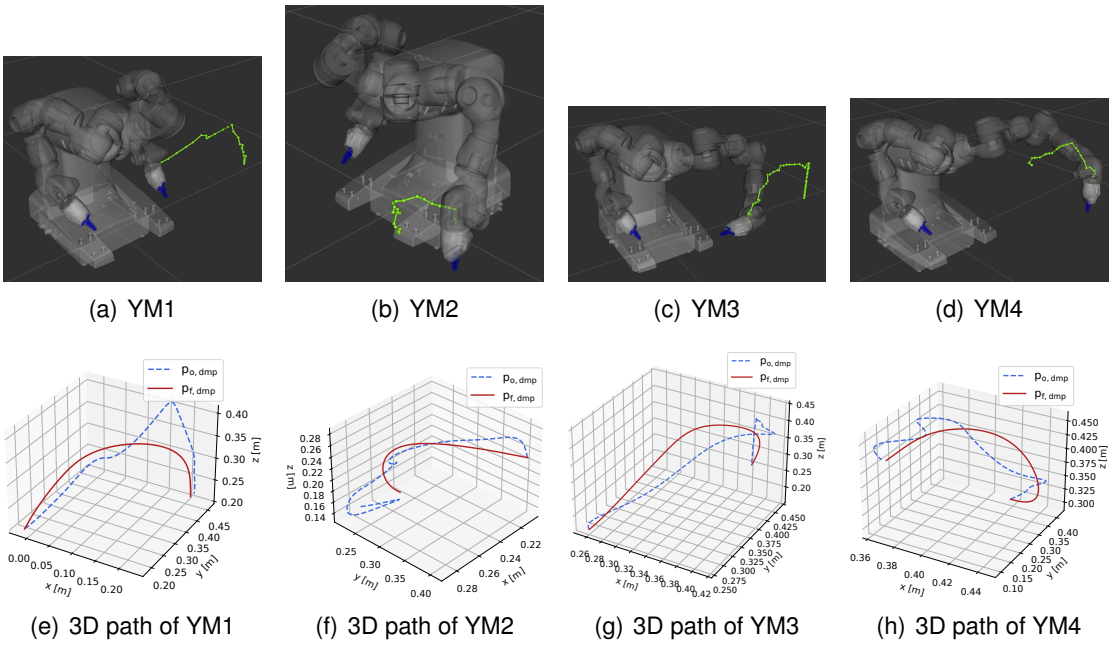
the forces applied by the operator. Unlike the gravity compensation mode, which can maintain smooth and steady movements by counteracting gravitational forces, the Lead-through mode can have sudden movements or stops based on human input. This can lead to inconsistencies and fluctuations in the trajectory, resulting in noisier data. Therefore, in these experiments, the role of noise removal becomes even more crucial.

6.3.2.1 DMP Case Study

Similar to Section 4.3.2.3 two sets of DMPs are trained: DMP_o using the original trajectories $\boldsymbol{q}_o(t)$ and DMP_f using the optimized trajectories $\boldsymbol{q}_f(t)$, reproducing $\boldsymbol{q}_{o,dmp}(t)$ and $\boldsymbol{q}_{f,dmp}(t)$, respectively, for the tasks YM1-YM4. Fig. 6.4(e) to 6.4(h) illustrate the paths of $\boldsymbol{p}_{o,dmp}(t)$ and $\boldsymbol{p}_{f,dmp}(t)$ for these tasks.

The presence of noise in the original demonstrations led to several reliability and safety issues, as evidenced by overfitting effects in each task. In YM1, $\boldsymbol{p}_{o,dmp}(t)$ shows an unnecessary movement by rising too far above the fitting's tray before approaching the object. In YM2, the noise significantly affects the motion towards the end, causing the robot to potentially collide with the mounting station for the fitting. Additionally, the path of $\boldsymbol{p}_{o,dmp}(t)$ fails to reach the goal configuration, requiring additional time for the DMP to converge to the goal. In YM3, the noise introduces an unnecessary curve in the end-effector's path before it descends to approach the ring, risking a collision with the back of the ring tray. Finally, in YM4, the noise results in several unnecessary curves, which negatively impact the efficiency of the motion execution.

Following a similar approach to Section 4.3.2.3, Table 6.1 provides a summary of the time and jerk values for YM1-YM4. The data highlight a significant improvement in both execution time and MANJ when DFL-TORO is incorporated into the LfD process. As reflected in the underlying path of $\boldsymbol{p}_{o,dmp}(t)$, the inherent noise-removal capabilities of standard LfD algorithms alone are insufficient to achieve optimal trajectories in a manufacturing context, where execution time and jerk are critical factors. This underscores the importance of using DFL-TORO, which significantly enhances the quality and efficiency of the generated trajectories by effectively filtering out noise and optimizing the motion.



(a) YM1 (b) YM2 (c) YM3 (d) YM4

(e) 3D path of YM1 (f) 3D path of YM2 (g) 3D path of YM3 (h) 3D path of YM4

Figure 6.4: Visualization of demonstration trajectories recorded using ABB YuMi, alongside the 3D paths generalized by DMP_o and DMP_f. The paths generated by DMP_o show noticeable effects of noise, which can cause the robot to collide with obstacles in the workspace or impact execution efficiency.

Table 6.1: Comparison of time [s] and Maximum Absolute Normalized Jerk (MANJ) [rad/s^3] for YM1-4.

Experiment	Time/Jerk	Demonstration		DMP	
		$\boldsymbol{q}_o(t)$	$\boldsymbol{q}_f(t)$	$\boldsymbol{q}_{o,dmp}(t)$	$\boldsymbol{q}_{f,dmp}(t)$
YM1	Time	9.75	**1.50**	9.75	**1.50**
	MANJ	139034.03	**492.11**	2386.13	**433.93**
YM2	Time	13.35	**1.10**	13.35	**1.10**
	MANJ	16148.34	**114.13**	3798.41	**168.37**
YM3	Time	79.75	**1.28**	79.75	**1.28**
	MANJ	5152541.33	**232.17**	16996.27	**227.79**
YM4	Time	14.15	**1.19**	14.15	**1.19**
	MANJ	153770.69	**190.80**	1839.68	**634.43**

6.4 Applying LfD Software

The final integration of the automation process utilized the LfD software framework. the software framework employed the DMP module to learn and generalize the demonstrated motion. DFL-TORO is applied to enhance and optimize the demonstrations to improve execution efficiency.

To enable perception capabilities, a custom camera module was developed to integrate Cognex cameras, which are deployed at the production site. These cameras acted as the perception device, providing object localization data critical for guiding the robot during task execution. Following the recording of required demonstrations, the LfD framework is used to translate these into high-level instructions for both arms of the ABB YuMi robot. These instructions construct the final program for the automation of the sub-assembly process.

6.4.1 Final Demo

The conclusion of this effort is presented in a video showcasing the automated sub-assembly process on the ABB YuMi robot at the production site. The video illustrates the successful application of the LfD software framework to achieve efficient automation based on LfD. The video of the final demo can be found here at https://drive.google.com

Chapter 7

Conclusion

This thesis has addressed the critical challenges of transitioning from mass production to mass customization in modern manufacturing, leveraging cobots and LfD as key enablers. By combining theoretical advancements and practical implementations, the research presented here offers a comprehensive approach to equipping industrial systems with the adaptability and efficiency required to meet evolving demands. The major contributions of this work can be summarized as follows:

1. **A Practical Roadmap for Deploying LfD in Manufacturing.** The proposed roadmap provides an actionable framework for practitioners to implement LfD solutions in industrial tasks. By addressing the key questions of ”What to Demonstrate,” ”How to Demonstrate,” ”How to Learn,” and ”How to Refine,” this roadmap guides users through the complexities of adapting robotic systems to mass customization. The questionnaire-style approach ensures accessibility for industry practitioners, bridging the gap between research and practical application.

2. **DFL-TORO.** This framework introduces innovative methodologies for capturing task requirements through one-shot demonstrations. By optimizing demonstration trajectories for time, jerk, and noise, DFL-TORO significantly improves operational efficiency while maintaining task accuracy. It represents a pivotal advancement in LfD by reducing the reliance on multiple demonstrations and addressing key limitations in existing

approaches.

3. **A Modular and Standardized LfD Software Framework.** The developed software framework integrates modularity, standardization, and accessibility to facilitate the deployment of LfD in industrial environments. Built on the ROS ecosystem and leveraging MoveIt, this framework provides a robot-agnostic, perception-integrated solution that simplifies the programming of collaborative robots. Its high-level interface enables non-experts to compose complex tasks intuitively, democratizing access to advanced robotic capabilities.

7.1 Achievements and Impact

This thesis has successfully validated its contributions through a combination of theoretical analysis, experimental evaluations, and an industrial case study. The roadmap demonstrated its utility in guiding the adaptation of robotic systems for flexible assembly processes. DFL-TORO showcased its effectiveness in improving task execution times and trajectory smoothness while reducing noise. The software framework proved to be a versatile and scalable solution, capable of supporting diverse LfD methods and robotic platforms.

Together, these contributions address critical gaps in current practices, providing a robust foundation for implementing LfD in manufacturing. By bridging the gap between research innovations and real-world deployment, this work enables industries to harness the full potential of collaborative robotics in the era of mass customization.

7.2 Future Directions

The future perspective for LfD in mass customization can be advanced by leveraging emerging technologies and addressing current limitations identified in this work.

- **Integration of LLMs for Human-Robot Interaction.** Integrating LLMs into the LfD framework can enhance human-robot interaction by facilitating real-time dialogue and

intelligent data analysis. LLM integration enables robots to better understand user inputs through natural language processing and optimize manufacturing processes. This capability allows for interactive learning, where robots continuously improve their performance based on user feedback, making them more adaptive and user-friendly.

- **LfD in Industry 5.0.** In the context of Industry 5.0, LfD can foster collaboration between humans and robots, supporting personalized assistance and enhancing workplace safety by enabling robots to perform hazardous tasks. Additionally, LfD facilitates adaptive manufacturing processes that can respond dynamically to changing requirements, aligning with the principles of resilience and human-centric production.

- **Practical Deployment and User Studies.** To validate the usability and practical benefits of the LfD framework, user studies involving industry practitioners such as process engineers should be conducted in real manufacturing settings. These studies would provide critical feedback for refining the system and ensuring its practical utility in diverse industrial applications.

- **Establishing Standardized Evaluation Metrics.** The variability and noisiness in human demonstrations present challenges for comparative evaluation with other LfD frameworks. Developing standardized metrics that account for these factors would enable more consistent benchmarking and objective assessment of the proposed framework.

Moreover, the DFL-TORO methodology has certain limitations that present opportunities for further improvement:

- **Automated Tolerance Assignment.** The reliance on default tolerance values for trajectory generation introduces challenges in accuracy and efficiency. Automating this process through semantic task understanding could improve adaptability and reduce the need for manual intervention.

- **Refinement Process with AR Integration:** The current refinement process, which requires human supervision at slower speeds, could be enhanced by using Augmented

Reality for virtual refinement. This approach mitigates safety risks and offers a more intuitive experience for human operators.

- **Online Learning and Optimization.** Extending DFL-TORO to support real-time planning and optimization would address its current offline constraints, improving learning speed and enabling adaptive responses in dynamic environments.

By addressing these directions, LfD can evolve into a more comprehensive and robust framework, integrating advanced technologies to enhance adaptability, user experience, and operational efficiency, paving the way for its wider adoption in mass customization and Industry 5.0 contexts.

Chapter 8

References

[1] Mikkel Rath Pedersen et al. “Robot skills for manufacturing: From concept to industrial deployment”. In: *Robot. Comput. Integr. Manuf.* 37 (2016), pp. 282–291.

[2] Yuval Cohen et al. “Assembly systems in Industry 4.0 era: a road map to understand Assembly 4.0”. In: *Int. J. Adv. Manuf. Technol.* 105 (2019), pp. 4037–4054.

[3] Jerry Wind and Arvind Rangaswamy. “Customerization: The next revolution in mass customization”. In: *Journal of interactive marketing* 15.1 (2001), pp. 13–32.

[4] Timotej Gašpar et al. “Smart hardware integration with advanced robot programming technologies for efficient reconfiguration of robot workcells”. In: *Robotics and Computer-Integrated Manufacturing* 66 (2020), p. 101979.

[5] Shirine El Zaatari et al. “Cobot programming for collaborative industrial tasks: An overview”. In: *Robotics and Autonomous Systems* 116 (2019), pp. 162–180.

[6] Christoffer Sloth, Aljaž Kramberger, and Iñigo Iturrate. “Towards easy setup of robotic assembly tasks”. In: *Adv. Robot* 34.7-8 (2020), pp. 499–513.

[7] Oliver Heimann and Jan Guhl. “Industrial robot programming methods: A scoping review”. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE. 2020, pp. 696–703.

[8] J Léger and J Angeles. “Off-line programming of six-axis robots for optimum five-dimensional tasks”. In: *Mechanism and Machine Theory* 100 (2016), pp. 155–169.

[9] Alireza Barekatain, Hamed Habibi, and Holger Voos. “A Practical Roadmap to Learning from Demonstration for Robotic Manipulators in Manufacturing”. In: *Robotics* 13.7 (2024), p. 100.

[10] Harish Ravichandar et al. “Recent advances in robot learning from demonstration”. In: *Annual review of control, robotics, and autonomous systems* 3 (2020), pp. 297–330.

[11] Emmanuel Dean-Leon et al. “Robotic technologies for fast deployment of industrial robot systems”. In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2016, pp. 6900–6907.

[12] Lindsay Sanneman, Christopher Fourie, Julie A Shah, et al. “The state of industrial robotics: Emerging technologies, challenges, and key research directions”. In: *Foundations and Trends® in Robotics* 8.3 (2021), pp. 225–306.

[13] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[14] Sachin Chitta. “MoveIt!: an introduction”. In: *Robot Operating System (ROS) The Complete Reference (Volume 1)* (2016), pp. 3–27.

[15] Bin Fang et al. “Survey of imitation learning for robotic manipulation”. In: *International Journal of Intelligent Robotics and Applications* 3 (2019), pp. 362–369.

[16] Zhihao Liu et al. “Robot learning towards smart robotic manufacturing: A review”. In: *Robotics and Computer-Integrated Manufacturing* 77 (2022), p. 102360.

[17] Zuyuan Zhu and Huosheng Hu. “Robot learning from demonstration in robotic assembly: A survey”. In: *Robotics* 7.2 (2018), p. 17.

[18] Arturo Daniel Sosa-Ceron, Hugo Gustavo Gonzalez-Hernandez, and Jorge Antonio Reyes-Avendaño. “Learning from Demonstrations in Human–Robot Collaborative Scenarios: A Survey”. In: *Robotics* 11.6 (2022), p. 126.

[19] Victor Hernandez Moreno et al. “Obstacles and opportunities for learning from demonstration in practical industrial assembly: A systematic literature review”. In: *Robotics and Computer-Integrated Manufacturing* 86 (2024), p. 102658.

[20] Weidong Li et al. “Learning from demonstration for autonomous generation of robotic trajectory: Status quo and forward-looking overview”. In: *Advanced Engineering Informatics* 62 (2024), p. 102625.

[21] Oliver Kroemer, Scott Niekum, and George Konidaris. “A review of robot learning for manipulation: Challenges, representations, and algorithms”. In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 1395–1476.

[22] Carlos Celemin et al. “Interactive imitation learning in robotics: A survey”. In: *Foundations and Trends® in Robotics* 10.1-2 (2022), pp. 1–197.

[23] ZongWu Xie et al. “Robot learning from demonstration for path planning: A review”. In: *Science China Technological Sciences* 63.8 (2020), pp. 1325–1334.

[24] Cristian C Beltran-Hernandez et al. “Accelerating Robot Learning of Contact-Rich Manipulations: A Curriculum Learning Study”. In: *arXiv preprint arXiv:2204.12844* (2022).

[25] Aude Billard et al. “Survey: Robot programming by demonstration”. In: *Springer handbook of robotics* (2008), pp. 1371–1394.

[26] Brenna D Argall et al. “A survey of robot learning from demonstration”. In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[27] Ya-Yen Tsai et al. “Constrained-space optimization and reinforcement learning for complex tasks”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 683–690.

[28] Sylvain Calinon. “A tutorial on task-parameterized movement learning and retrieval”. In: *Intell. Serv. Robot.* 9 (2016), pp. 1–29.

[29] Anna Mészáros, Giovanni Franzese, and Jens Kober. “Learning to Pick at Non-Zero-Velocity From Interactive Demonstrations”. In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 6052–6059.

[30] YQ Wang et al. “Optimised learning from demonstrations for collaborative robots”. In: *Robotics and Computer-Integrated Manufacturing* 71 (2021), p. 102169.

[31] Bojan Nemec et al. “An efficient pbd framework for fast deployment of bi-manual assembly tasks”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 166–173.

[32] Mihael Simonič et al. “Analysis of methods for incremental policy refinement by kinesthetic guidance”. In: *Journal of Intelligent & Robotic Systems* 102.1 (2021), p. 5.

[33] Jacopo Aleotti and Stefano Caselli. “Robust trajectory learning and approximation for robot programming by demonstration”. In: *Robotics and Autonomous Systems* 54.5 (2006), pp. 409–413.

[34] Luigi Biagiotti et al. “Robot Programming by Demonstration: Trajectory Learning Enhanced by sEMG-Based User Hand Stiffness Estimation”. In: *IEEE Transactions on Robotics* (2023).

[35] RB Ashith Shyam et al. “Improving local trajectory optimisation using probabilistic movement primitives”. In: *2019 IEEE/RSJ Int. Conf. Intel. Robots Syst. (IROS)*. IEEE. 2019, pp. 2666–2671.

[36] Carl De Boor. “On calculating with B-splines”. In: *J. Approx. Theory* 6.1 (1972), pp. 50–62.

[37] Hui Yang et al. “A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization”. In: *IEEE Trans. Indust. Elect.* 66.11 (2018), pp. 8557–8566.

[38] Yanqin Ma et al. “An efficient robot precision assembly skill learning framework based on several demonstrations”. In: *IEEE Transactions on Automation Science and Engineering* 20.1 (2022), pp. 124–136.

[39] Kun Qian et al. “Hierarchical and parameterized learning of pick-and-place manipulation from under-specified human demonstrations”. In: *Advanced Robotics* 34.13 (2020), pp. 858–872.

[40] Chen He et al. “Imitation Learning Study for Robotic Peg-in-hole Assembly”. In: *2021 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT)*. IEEE. 2021, pp. 339–344.

[41] Hongmin Wu et al. “A Framework of Improving Human Demonstration Efficiency for Goal-Directed Robot Skill Learning”. In: *IEEE Trans. Cogn. Develop. Syst.* 14.4 (2021), pp. 1743–1754.

[42] Shirine El Zaatari et al. “iTP-LfD: Improved task parametrised learning from demonstration for adaptive path generation of cobot”. In: *Robotics and Computer-Integrated Manufacturing* 69 (2021), p. 102109.

[43] Congcong Ye, Jixiang Yang, and Han Ding. “Bagging for Gaussian mixture regression in robot learning from demonstration”. In: *Journal of Intelligent Manufacturing* 33.3 (2022), pp. 867–879.

[44] Haopeng Hu, Xiansheng Yang, and Yunjiang Lou. “A robot learning from demonstration framework for skillful small parts assembly”. In: *The International Journal of Advanced Manufacturing Technology* 119.9-10 (2022), pp. 6775–6787.

[45] Chris Paxton, Gregory D Hager, Luca Bascetta, et al. “An incremental approach to learning generalizable robot tasks from human demonstration”. In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 5616–5621.

[46] Loris Roveda et al. “Assembly task learning and optimization through human’s demonstration and machine learning”. In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2020, pp. 1852–1859.

[47] Nathan Delson and Harry West. “Robot programming by human demonstration: The use of human inconsistency in improving 3D robot trajectories”. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’94)*. Vol. 2. IEEE. 1994, pp. 1248–1255.

[48] Yusuke Maeda, Tatsuya Ushioda, and Satoshi Makita. “Easy robot programming for industrial manipulators by manual volume sweeping”. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 2234–2239.

[49] Daniel Müller et al. “One-Shot kinesthetic programming by demonstration for soft collaborative robots”. In: *Mechatronics* 70 (2020), p. 102418.

[50] Ajay Mandlekar et al. “What matters in learning from offline human demonstrations for robot manipulation”. In: *arXiv preprint arXiv:2108.03298* (2021).

[51] Ankit Goyal et al. “RVT2: Learning Precise Manipulation from Few Demonstrations”. In: *RSS* (2024).

[52] Mingfei Sun and Xiaojuan Ma. “Adversarial imitation learning from incomplete demonstrations”. In: *arXiv preprint arXiv:1905.12310* (2019).

[53] Giovanni Franzese et al. “ILoSA: Interactive learning of stiffness and attractors”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 7778–7785.

[54] Wanxin Jin et al. “Learning from Sparse Demonstrations”. In: *arXiv preprint arXiv:2008.02159* (2020).

[55] Junjia Liu et al. *Rofunc: The Full Process Python Package for Robot Learning from Demonstration and Robot Manipulation*. 2023. DOI: 10.5281/zenodo.10016946. URL: https://doi.org/10.5281/zenodo.10016946.

[56] Lin Shao et al. “Concept2robot: Learning manipulation concepts from instructions and human demonstrations”. In: *The International Journal of Robotics Research* 40.12-14 (2021), pp. 1419–1434.

[57] Staffan Ekvall and Danica Kragic. “Robot learning from demonstration: a task-level planning approach”. In: *International Journal of Advanced Robotic Systems* 5.3 (2008), p. 33.

[58] Vamsi Krishna Origanti, Thomas Eiband, and Dongheui Lee. “Automatic parameterization of motion and force controlled robot skills”. In: *International Conference on Robot Intelligence Technology and Applications*. Springer. 2021, pp. 66–78.

[59] Scott Niekum et al. “Learning grounded finite-state representations from unstructured demonstrations”. In: *The International Journal of Robotics Research* 34.2 (2015), pp. 131–157.

[60] Franz Steinmetz, Verena Nitsch, and Freek Stulp. “Intuitive task-level programming by demonstration through semantic skill recognition”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3742–3749.

[61] Matteo Iovino et al. “A Framework for Learning Behavior Trees in Collaborative Robotic Applications”. In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2023, pp. 1–8.

[62] Kevin David French et al. “Super Intendo: Semantic Robot Programming from Multiple Demonstrations for taskable robots”. In: *Robotics and Autonomous Systems* 166 (2023), p. 104397.

[63] Christoph Willibald and Dongheui Lee. “Multi-level task learning based on intention and constraint inference for autonomous robotic manipulation”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 7688–7695.

[64] Luisa Mayershofer et al. “Task-Level Programming by Demonstration for Mobile Robotic Manipulators through Human Demonstrations based on Semantic Skill Recognition”. In: *ISR Europe 2023; 56th International Symposium on Robotics*. VDE. 2023, pp. 22–29.

[65] Simona Gugliermo et al. “Learning behavior trees from planning experts using decision tree and logic factorization”. In: *IEEE Robotics and Automation Letters* (2023).

[66] Lisa Scherf, Kevin Fröhlich, and Dorothea Koert. “Learning Action Conditions for Automatic Behavior Tree Generation from Human Demonstrations”. In: *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 2024, pp. 950–954.

[67] Thomas Eiband et al. “Online task segmentation by merging symbolic and data-driven skill recognition during kinesthetic teaching”. In: *Robotics and Autonomous Systems* 162 (2023), p. 104367.

[68] Jonathan Feng-Shun Lin, Michelle Karg, and Dana Kulić. “Movement primitive segmentation for human motion modeling: A framework for analysis”. In: *IEEE Transactions on Human-Machine Systems* 46.3 (2016), pp. 325–339.

[69] Simon Lyck Bjært Sørensen, Thiusius Rajeeth Savarimuthu, and Iñigo Iturrate. “Robot Task Primitive Segmentation from Demonstrations Using Only Built-in Kinematic State and Force-Torque Sensor Data”. In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2023, pp. 1–7.

[70] Christian RG Dreher and Tamim Asfour. “Learning temporal task models from human bimanual demonstrations”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 7664–7671.

[71] Feng Zhou, Fernando De la Torre, and Jessica K Hodgins. “Hierarchical aligned cluster analysis for temporal clustering of human motion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.3 (2012), pp. 582–596.

[72] Caiming Xiong et al. “Robot learning with a spatial, temporal, and causal and-or graph”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 2144–2151.

[73] Estuardo Carpio, Madison Clark-Turner, and Momotaz Begum. “Learning sequential human-robot interaction tasks from demonstrations: The role of temporal reasoning”. In: *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2019, pp. 1–8.

[74] Oscar Gustavsson et al. “Combining context awareness and planning to learn behavior trees from demonstration”. In: *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2022, pp. 1153–1160.

[75] Marco Ewerton et al. “Incremental imitation learning of context-dependent motor skills”. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2016, pp. 351–358.

[76] Emmanuel Pignat and Sylvain Calinon. “Bayesian Gaussian mixture model for robotic policy imitation”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4452–4458.

[77] Guilherme Maeda et al. “Active incremental learning of robot movement primitives”. In: *Conference on Robot Learning*. PMLR. 2017, pp. 37–46.

[78] Kaimeng Wang, Yongxiang Fan, and Ichiro Sakuma. “Robot Grasp Planning: A Learning from Demonstration-Based Approach”. In: *Sensors* 24.2 (2024), p. 618.

[79] Dorothea Koert et al. “Learning intention aware online adaptation of movement primitives”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3719–3726.

[80] Gennaro Raiola, Xavier Lamy, and Freek Stulp. “Co-manipulation with multiple probabilistic virtual guides”. In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2015, pp. 7–13.

[81] Anahita Mohseni-Kabir et al. “Simultaneous learning of hierarchy and primitives for complex robot tasks”. In: *Autonomous Robots* 43 (2019), pp. 859–874.

[82] Andreea Bobu et al. “Aligning Human and Robot Representations”. In: *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 2024, pp. 42–54.

[83] Zhipeng Dong et al. “Passive bimanual skills learning from demonstration with motion graph attention networks”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4917–4923.

[84] Dong Liu et al. “Robotic manipulation skill acquisition via demonstration policy learning”. In: *IEEE Transactions on Cognitive and Developmental Systems* 14.3 (2021), pp. 1054–1065.

[85] Yaqiang Mo et al. “Multi-step motion learning by combining learning-from-demonstration and policy-search”. In: *Advanced Robotics* 37.9 (2023), pp. 560–575.

[86] Felix Frank et al. “Constrained probabilistic movement primitives for robot trajectory adaptation”. In: *IEEE Transactions on Robotics* 38.4 (2021), pp. 2276–2294.

[87] Di-Hua Zhai et al. “A motion planning method for robots based on DMPS and modified obstacle-avoiding algorithm”. In: *IEEE Transactions on Automation Science and Engineering* (2022).

[88] Sayantan Auddy et al. “Continual learning from demonstration of robotics skills”. In: *Robotics and Autonomous Systems* 165 (2023), p. 104427.

[89] Sipu Ruan et al. “PRIMP: PRobabilistically-Informed Motion Primitives for Efficient Affordance Learning from Demonstration”. In: *IEEE Transactions on Robotics* (2024).

[90] Nghia Vuong, Hung Pham, and Quang-Cuong Pham. “Learning sequences of manipulation primitives for robotic assembly”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4086–4092.

[91] Zheng Wu et al. “Prim-lafd: A framework to learn and adapt primitive-based skills from demonstrations for insertion tasks”. In: *IFAC-PapersOnLine* 56.2 (2023), pp. 4120–4125.

[92] Lars Johannsmeier, Malkin Gerchow, and Sami Haddadin. “A framework for robot manipulation: Skill formalism, meta learning and adaptive control”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 5844–5850.

[93] Zheng Wu et al. “Learning dense rewards for contact-rich manipulation tasks”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6214–6221.

[94] Michelle A Lee et al. “Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8943–8950.

[95] Todor Davchev et al. “Residual learning from demonstration: Adapting dmps for contact-rich manipulation”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4488–4495.

[96] Mel Vecerik et al. “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards”. In: *arXiv preprint arXiv:1707.08817* (2017).

[97] Josip Vidaković et al. “Accelerating robot trajectory learning for stochastic tasks”. In: *IEEE access* 8 (2020), pp. 71993–72006.

[98] Cristian Alejandro Vergara Perico, Joris De Schutter, and Erwin Aertbeliën. “Combining imitation learning with constraint-based task specification and control”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1892–1899.

[99] Weiyong Si, Yuan Guan, and Ning Wang. “Adaptive compliant skill learning for contact-rich manipulation with human in the loop”. In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 5834–5841.

[100] Wei Wang, Robert NK Loh, and Edward Y Gu. “Passive compliance versus active compliance in robot-based automated assembly systems”. In: *Industrial Robot: An International Journal* 25.1 (1998), pp. 48–57.

[101] Peng Song, Yueqing Yu, and Xuping Zhang. “A tutorial survey and comparison of impedance control on robotic manipulation”. In: *Robotica* 37.5 (2019), pp. 801–836.

[102] Neville Hogan. “Impedance control of industrial robots”. In: *Robotics and computer-integrated manufacturing* 1.1 (1984), pp. 97–113.

[103] Joohwan Seo et al. “Contact-rich SE (3)-Equivariant Robot Manipulation Task Learning via Geometric Impedance Control”. In: *IEEE Robotics and Automation Letters* (2023).

[104] Theodora Kastritsi, Fotios Dimeas, and Zoe Doulgeri. “Progressive automation with dmp synchronization and variable stiffness control”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3789–3796.

[105] Shangshang Yang et al. “Learning Pose Dynamical System for Contact Tasks under Human Interaction”. In: *Actuators*. Vol. 12. 4. MDPI. 2023, p. 179.

[106] Weitian Wang et al. “Facilitating human–robot collaborative tasks by teaching-learning-collaboration from human demonstrations”. In: *IEEE Transactions on Automation Science and Engineering* 16.2 (2018), pp. 640–653.

[107] Bojan Nemec et al. “Human robot cooperation with compliance adaptation along the motion trajectory”. In: *Autonomous robots* 42 (2018), pp. 1023–1035.

[108] Thomas Eiband et al. “Collaborative programming of robotic task decisions and recovery behaviors”. In: *Autonomous Robots* 47.2 (2023), pp. 229–247.

[109] Leonel Rozo et al. “Learning physical collaborative robot behaviors from human demonstrations”. In: *IEEE Transactions on Robotics* 32.3 (2016), pp. 513–527.

[110] Devesh K Jha et al. “Generalizable human-robot collaborative assembly using imitation learning and force control”. In: *2023 European Control Conference (ECC)*. IEEE. 2023, pp. 1–8.

[111] Mahdi Khoramshahi and Aude Billard. “A dynamical system approach to task-adaptation in physical human–robot interaction”. In: *Autonomous Robots* 43 (2019), pp. 927–946.

[112] Roohollah Jahanmahin et al. “Human-robot interactions in manufacturing: A survey of human behavior modeling”. In: *Robotics and Computer-Integrated Manufacturing* 78 (2022), p. 102404.

[113] Xueyan Xing et al. “Dynamic Motion Primitives-based Trajectory Learning for Physical Human-Robot Interaction Force Control”. In: *IEEE Transactions on Industrial Informatics* (2023).

[114] Giovanni Franzese et al. “Interactive imitation learning of bimanual movement primitives”. In: *IEEE/ASME Transactions on Mechatronics* (2023).

[115] Franziska Krebs et al. “The kit bimanual manipulation dataset”. In: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2021, pp. 499–506.

[116] Simon Stepputtis et al. “A system for imitation learning of contact-rich bimanual manipulation policies”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 11810–11817.

[117] Junjia Liu et al. “Birp: Learning robot generalized bimanual coordination using relative parameterization method on human demonstration”. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE. 2023, pp. 8300–8305.

[118] Xiaofeng Mao et al. “Learning fine pinch-grasp skills using tactile sensing from real demonstration data”. In: *arXiv preprint arXiv:2307.04619* (2023).

[119] Noémie Jaquier, David Ginsbourger, and Sylvain Calinon. “Learning from demonstration with model-based Gaussian process”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 247–257.

[120] Miguel Arduengo et al. “Gaussian-process-based robot learning from demonstration”. In: *Journal of Ambient Intelligence and Humanized Computing* (2023), pp. 1–14.

[121] Guanwen Ding et al. “A task-learning strategy for robotic assembly tasks from human demonstrations”. In: *Sensors* 20.19 (2020), p. 5505.

[122] Thibaut Kulak et al. “Active learning of Bayesian probabilistic movement primitives”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2163–2170.

[123] Adrian Prados, Santiago Garrido, and Ramon Barber. “Learning and generalization of task-parameterized skills through few human demonstrations”. In: *Engineering Applications of Artificial Intelligence* 133 (2024), p. 108310.

[124] Isacco Zappa et al. “Parameterization of Robotic Welding Trajectories from Demonstration”. In: *2023 11th International Conference on Control, Mechatronics and Automation (ICCMA)*. IEEE. 2023, pp. 146–151.

[125] Zhenxi Cui et al. “Coupled multiple dynamic movement primitives generalization for deformable object manipulation”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5381–5388.

[126] Xing Li and Oliver Brock. “Learning from demonstration based on environmental constraints”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10938–10945.

[127] Edward Johns. “Coarse-to-fine imitation learning: Robot manipulation from a single demonstration”. In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 4613–4619.

[128] Yunlei Shi et al. “Combining learning from demonstration with learning by exploration to facilitate contact-rich tasks”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1062–1069.

[129] Felix Wohlgemuth et al. “Electromyography-based Kinesthetic Teaching of Industrial Collaborative Robots”. In: *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 2024, pp. 1124–1128.

[130] Adrián Prados et al. “Kinesthetic learning based on fast marching square method for manipulation”. In: *Applied Sciences* 13.4 (2023), p. 2028.

[131] Alireza Barekatain, Hamed Habibi, and Holger Voos. “DFL-TORO: A One-Shot Demonstration Framework for Learning Time-Optimal Robotic Manufacturing Tasks”. In: *arXiv preprint arXiv:2309.09802* (2023).

[132] Weiyong Si, Ning Wang, and Chenguang Yang. “A review on manipulation skill acquisition through teleoperation-based learning from demonstration”. In: *Cognitive Computation and Systems* 3.1 (2021), pp. 1–16.

[133] Marc Rigter, Bruno Lacerda, and Nick Hawes. “A framework for learning from demonstration with minimal human effort”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2023–2030.

[134] Albert Tung et al. “Learning multi-arm manipulation through collaborative teleoperation”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 9212–9219.

[135] Jing Luo et al. “A vision-based virtual fixture with robot learning for teleoperation”. In: *Robotics and Autonomous Systems* 164 (2023), p. 104414.

[136] Başak Güleçyüz et al. “NetLfD: Network-Aware Learning from Demonstration for In-Contact Skills via Teleoperation”. In: *IEEE Robotics and Automation Letters* (2023).

[137] Congcong Yin and Qiuju Zhang. “A multi-modal framework for robots to learn manipulation tasks from human demonstrations”. In: *Journal of Intelligent & Robotic Systems* 107.4 (2023), p. 56.

[138] Xinghao Zhu et al. “Diff-lfd: Contact-aware model-based learning from visual demonstration for robotic manipulation via differentiable physics-based simulation and rendering”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 499–512.

[139] Shuo Yang et al. “Watch and act: Learning robotic manipulation from visual demonstration”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2023).

[140] Xuanhui Xu et al. “Robot imitation learning from image-only observation without real-world interaction”. In: *IEEE/ASME Transactions on Mechatronics* (2022).

[141] Erdem Bıyık et al. “Active preference-based Gaussian process regression for reward learning and optimization”. In: *The International Journal of Robotics Research* (2023), p. 02783649231208729.

[142] Carlos Celemin and Javier Ruiz-del-Solar. “An interactive framework for learning continuous actions policies based on corrective feedback”. In: *Journal of Intelligent & Robotic Systems* 95 (2019), pp. 77–97.

[143] Peter Pastor et al. “Learning and generalization of motor skills by learning from demonstration”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 763–768.

[144] Mehrdad Tavassoli et al. “Learning skills from demonstrations: A trend from motion primitives to experience abstraction”. In: *IEEE Transactions on Cognitive and Developmental Systems* (2023).

[145] Eugenio Chisari et al. “Correct me if i am wrong: Interactive learning for robotic manipulation”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3695–3702.

[146] Hang Yin et al. “An ensemble inverse optimal control approach for robotic task learning and adaptation”. In: *Autonomous Robots* 43 (2019), pp. 875–896.

[147] You Zhou, Jianfeng Gao, and Tamim Asfour. “Movement primitive learning and generalization: Using mixture density networks”. In: *IEEE Robotics & Automation Magazine* 27.2 (2020), pp. 22–32.

[148] Auke Jan Ijspeert et al. “Dynamical movement primitives: learning attractor models for motor behaviors”. In: *Neural computation* 25.2 (2013), pp. 328–373.

[149] Matteo Saveriano et al. “Dynamic movement primitives in robotics: A tutorial survey”. In: *The International Journal of Robotics Research* 42.13 (2023), pp. 1133–1184.

[150] Bojan Nemec, Andrej Gams, and Aleš Ude. “Velocity adaptation for self-improvement of skills learned from user demonstrations”. In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2013, pp. 423–428.

[151] Seiji Shaw et al. “Constrained dynamic movement primitives for safe learning of motor skills”. In: *arXiv preprint arXiv:2209.14461* (2022).

[152] Antonis Sidiropoulos, Dimitrios Papageorgiou, and Zoe Doulgeri. “A novel framework for generalizing dynamic movement primitives under kinematic constraints”. In: *Autonomous Robots* 47.1 (2023), pp. 37–50.

[153] Antonis Sidiropoulos and Zoe Doulgeri. “A reversible dynamic movement primitive formulation”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3147–3153.

[154] Fares J Abu-Dakka, Matteo Saveriano, and Ville Kyrki. “A Unified Formulation of Geometry-aware Dynamic Movement Primitives”. In: *arXiv preprint arXiv:2203.03374* (2022).

[155] Matteo Saveriano, Felix Franzel, and Dongheui Lee. “Merging position and orientation motion primitives”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7041–7047.

[156] Liang Han et al. “Modified dynamic movement primitives: robot trajectory planning and force control under curved surface constraints”. In: *IEEE transactions on cybernetics* (2022).

[157] Chunyang Chang et al. “Impedance adaptation by reinforcement learning with contact dynamic movement primitives”. In: *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2022, pp. 1185–1191.

[158] Zhiwei Liao et al. “Dynamic skill learning from human demonstration based on the human arm stiffness estimation model and Riemannian DMP”. In: *IEEE/ASME Transactions on Mechatronics* 28.2 (2022), pp. 1149–1160.

[159] Emre Ugur and Hakan Girgin. “Compliant parametric dynamic movement primitives”. In: *Robotica* 38.3 (2020), pp. 457–474.

[160] Antonis Sidiropoulos and Zoe Doulgeri. “Dynamic via-points and improved spatial generalization for online trajectory planning with Dynamic Movement Primitives”. In: *arXiv preprint arXiv:2212.13473* (2022).

[161] William S Cleveland and Susan J Devlin. “Locally weighted regression: an approach to regression analysis by local fitting”. In: *Journal of the American statistical association* 83.403 (1988), pp. 596–610.

[162] Jan Peters et al. “Towards motor skill learning for robotics”. In: *Robotics Research: The 14th International Symposium ISRR*. Springer. 2011, pp. 469–482.

[163] Kaimeng Wang, Yu Zhao, and Ichiro Sakuma. “Learning robotic insertion tasks from human demonstration”. In: *IEEE Robotics and Automation Letters* (2023).

[164] Yanqin Ma, De Xu, and Fangbo Qin. “Efficient insertion control for precision assembly based on demonstration learning and reinforcement learning”. In: *IEEE Transactions on Industrial Informatics* 17.7 (2020), pp. 4492–4502.

[165] Neha Das et al. “Model-based inverse reinforcement learning from visual demonstrations”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 1930–1942.

[166] Minttu Alakuijala et al. “Learning reward functions for robotic manipulation by observing humans”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5006–5012.

[167] Tu Trinh, Haoyu Chen, and Daniel S Brown. “Autonomous assessment of demonstration sufficiency via bayesian inverse reinforcement learning”. In: *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 2024, pp. 725–733.

[168] Alejandro Escontrela et al. “Video prediction models as rewards for reinforcement learning”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[169] Jihong Zhu, Michael Gienger, and Jens Kober. “Learning task-parameterized skills from few demonstrations”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4063–4070.

[170] Alexandros Paraschos et al. “Probabilistic movement primitives”. In: *Advances in neural information processing systems* 26 (2013).

[171] Chengfei Yue et al. “Probabilistic movement primitives based multi-task learning framework”. In: *Computers & Industrial Engineering* (2024), p. 110144.

[172] *Robots and robotic devices – Collaborative robots*. Technical Specification ISO/TS 15066:2016. International Organization for Standardization, 2016. URL: https://www.iso.org/standard/62996.html.

[173] Peter Chemweno, Liliane Pintelon, and Wilm Decre. “Orienting safety assurance with outcomes of hazard analysis and risk assessment: A review of the ISO 15066 standard for collaborative robot systems”. In: *Safety Science* 129 (2020), p. 104832.

[174] Yue Yang et al. “Enhancing Safety in Learning from Demonstration Algorithms via Control Barrier Function Shielding”. In: *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 2024, pp. 820–829.

[175] Jiaqi Wang et al. “Large language models for robotics: Opportunities, challenges, and perspectives”. In: *arXiv preprint arXiv:2401.04334* (2024).

[176] Chao Zhang et al. “Towards new-generation human-centric smart manufacturing in Industry 5.0: A systematic review”. In: *Advanced Engineering Informatics* 57 (2023), p. 102121.

[177] Shanhe Lou et al. “Human-Cyber-Physical System for Industry 5.0: A Review From a Human-Centric Perspective”. In: *IEEE Transactions on Automation Science and Engineering* (2024), pp. 1–18.

[178] Yan Liu et al. “Analyzing the robotic behavior in a smart city with deep enforcement and imitation learning using IoRT”. In: *Computer Communications* 150 (2020), pp. 346–356.

[179] Laura Romeo et al. “Internet of robotic things in smart domains: Applications and challenges”. In: *Sensors* 20.12 (2020), p. 3355.

[180] Milan Groshev et al. “Edge robotics: Are we ready? An experimental evaluation of current vision and future directions”. In: *Digital Communications and Networks* 9.1 (2023), pp. 166–174.

[181] Changchun Liu et al. “An augmented reality-assisted interaction approach using deep reinforcement learning and cloud-edge orchestration for user-friendly robot teaching”. In: *Robotics and Computer-Integrated Manufacturing* 85 (2024), p. 102638.

[182] Xi Vincent Wang and Lihui Wang. “Augmented reality enabled human–robot collaboration”. In: *Advanced Human-Robot Collaboration in Manufacturing*. Springer, 2021, pp. 395–411.

[183] Ludovic Hamon. “Virtual reality and programming by demonstration: Teaching a robot to grasp a dynamic object by the generalization of human demonstrations”. In: *Presence* 20.3 (2011), pp. 241–253.

[184] Jonatan S Dyrstad and John Reidar Mathiassen. “Grasping virtual fish: A step towards robotic deep learning from demonstration in virtual reality”. In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2017, pp. 1181–1187.

[185] J Michael McCarthy. *Introduction to theoretical kinematics*. MIT press, 1990.

[186] Franka Emika. *Robot and interface specifications — Franka Control Interface (FCI) documentation*. 2023. URL: https://frankaemika.github.io/docs/control_parameters.html.

[187] Russ Tedrake and the Drake Development Team. *Drake: Model-based design and verification for robotics*. 2019. URL: https://drake.mit.edu.

[188] Freek Stulp and Gennaro Raiola. “DmpBbo: A versatile Python/C++ library for Function Approximation, Dynamical Movement Primitives, and Black-Box Optimization”. In: *J. Open Source Softw.* (2019). DOI: 10.21105/joss.01225. URL: https://www.theoj.org/joss-papers/joss.01225/10.21105.joss.01225.pdf.

[189] ROS Industrial. *The new ROS driver for ABB robots*. 2024. URL: https://github.com/ros-industrial/abb_robot_driver.