Reliable Autonomy at the Intersection of Constrained Motion Planning, Learning from Demonstration, and Augmented Reality

by

C. L. Mueller

M.S., University of Colorado Boulder, 2021

B.S., University of California Santa Barbara, 2011

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2023

Committee Members:

Bradley Hayes, Chair

Christoffer Heckman

Alessandro Roncone

Nisar Ahmed

Laura Hiatt

Mueller, C. L. (Ph.D., Computer Science)

Reliable Autonomy at the Intersection of Constrained Motion Planning, Learning from Demonstration, and Augmented Reality

Thesis directed by Prof. Bradley Hayes

Historically, robot automation has targeted applications that require consistency, precision, and long-term repeatability. Tasks that are dynamic or require operation in close proximity to human users reveal traditional robot controllers to be inflexible, costly, and unsafe. In response, Robot Learning from Demonstration (LfD) methods enable users to teach robots through actions, forgoing the need for programming expertise and providing a mechanism for flexibility. However, one limitation of traditional LfD methods is that they often utilize limited or context-independent information modalities, such as robot configurations, that inhibit the capture of pertinent skill information. This thesis presents a set of algorithms and user interaction systems that focus on enabling human users to communicate additional information in the form of constraints to a robot learning system. This results in more robust, generalizable, and safe skill execution. It will outline how constraints based on abstract, high-level concepts can be integrated into existing LfD methods, how unique interfaces can further enhance the communication of such constraints, and how the grounding of these constraints requires novel constrained motion planning techniques.

Dedication

> "Success is not final, failure is not fatal: It is the courage to continue that counts."
> – Winston Churchill

This document is dedicated to all the friends, family, loved ones, and colleagues who encouraged me to persevere, and taught me that my graduate studies are worthwhile means themselves.

Acknowledgements

First and foremost, I would like to acknowledge my advisor, Professor Bradley Hayes. Without his unwavering and optimistic support, I would likely not be where I am today. Thank you!

I would also like to acknowledge the greater CU Boulder community, which includes my committee members, colleagues, administrators, and staff, all of whom helped to create an encouraging environment to pursue graduate studies.

The work presented in this dissertation was conducted with the support of the National Science Foundation (award # 1830686).

Contents

Chapter

1 Introduction **1**

1.1 Technical Motivation 4

1.2 Thesis Statement 5

1.3 Contributions 5

1.4 Dissertation Outline 7

2 Constrained Robot Learning from Demonstration **9**

2.1 Robot Learning from Demonstration Preliminaries 10

2.1.1 What is Learning from Demonstration? 10

2.1.2 Modes of Interaction 11

2.1.3 Data Used for Learning 11

2.1.4 Characterization of Robot Learning from Demonstration 12

2.1.5 Incorporating Context into LfD 14

2.2 Concept Constrained Learning from Demonstration 15

2.2.1 Conceptual Constraints 16

2.2.2 The CC-LfD Algorithm and Model 17

2.2.3 Evaluation 27

2.2.4 Implemented Conceptual Constraints 29

2.3 Results 30

2.3.1 Evaluation Tasks . . . 30
2.3.2 Evaluation Criteria . . . 31
2.3.3 Results and Discussion . . . 31
2.3.4 Contributions: . . . 33
2.4 Maintaining Constraint-Compliance Introduces New Challenges . . . 34
2.4.1 Challenge I: Interface Design and Model Insight . . . 34
2.4.2 Challenge II: Keyframe Sparsity and Constraint-Compliant Motion Plans . . . 35

3 Augmented Reality Interfaces for Learning from Demonstration 38

3.1 Preliminaries . . . 40
3.1.1 Utility of Augmented Reality for Learning from Demonstration . . . 41
3.1.2 Revisiting Interaction Modes for LfD . . . 44
3.1.3 Task-space to Configuration Space Optimization . . . 45
3.1.4 Feedback to Foster Self-Correction . . . 47
3.2 System I: ARC-LfD . . . 47
3.2.1 System Design . . . 49
3.2.2 Interaction Flow . . . 49
3.2.3 Skill & Constraint Representation . . . 51
3.2.4 Constraint Editing & Application . . . 52
3.2.5 System Validation . . . 53
3.2.6 Case Study I (Precise Placement): A Placement Task with Orientation Change at Goal Pose . . . 55
3.2.7 Case Study II (Changing Environment): Introducing New Obstacles in a Pick-and-Place Task . . . 55
3.2.8 Case Study III (Changing Goal): Moving the Receptacle for a Pouring Task . 56
3.2.9 Benefits . . . 57
3.3 System II: ARPOC-LfD . . . 58

3.3.1 System Design . . . 58
3.3.2 Interaction Flow . . . 60
3.3.3 Hypotheses . . . 62
3.3.4 Experiment Protocol . . . 62
3.3.5 Evaluation Protocol . . . 69

4 Combining Constrained Motion Planning and Learning from Demonstration 71

4.1 Constrained Motion Planning Preliminaries . . . 72
4.1.1 Sampling-based Motion Planning . . . 72
4.1.2 Constrained Motion Planning . . . 74
4.1.3 Biased Sampling in Motion Planning . . . 80
4.1.4 Sequential Manifold Planning Problems . . . 81
4.2 Intersection Point Dependence Relaxation . . . 85
4.2.1 Information Needed to Solve SMPPs . . . 86
4.2.2 ρ-usefulness and the Ω-set . . . 86
4.2.3 IPD-Relaxation Formulation . . . 87
4.2.4 Constrained-LfD Keyframe Distribution Taxonomy . . . 88
4.2.5 The IPD-Relaxation Algorithm . . . 89
4.3 Evaluation . . . 91
4.3.1 Evaluation Domains . . . 91
4.3.2 Metrics . . . 92
4.3.3 Experimental Conditions to Evaluate IPD-Relaxation . . . 92
4.3.4 Intersection Point Generation Mechanism Details . . . 93
4.4 Evaluation Experiments . . . 95
4.4.1 Domain I - Constraint Demonstration for Biasing . . . 95
4.4.2 Domain II - 2D Navigation with Explicit Constraints . . . 95

4.4.3 Evaluation Domain 3: Simulated Manipulator Arm with Implicit Manifold Constraints 97

4.5 Results and Discussion 101

4.5.1 Domain I Results - Biased Sampling 101

4.5.2 Domain II Results - 2D Navigation with Explicit Constraints 101

4.5.3 Domain III Results - Simulated Manipulator Arm with Implicit Manifold Constraints 103

5 Conclusion **106**

5.1 Summary of Contributions 107

5.1.1 Concept Constrained Learning from Demonstration (CC-LfD) 107

5.1.2 Augmented Reality Systems for Constrained Robot Learning from Demonstration 108

5.1.3 Intersection Point Dependency Relaxation 109

5.2 Implications for Future Work 110

Bibliography **113**

Tables

Table

2.1 CC-LfD Evaluation Critera 30

2.2 Pouring Task Evaluation Results 32

2.3 Placement Task Evaluation Results. 33

3.1 Implemented Constraints for ARC-LfD System Validation 52

4.1 Description of Metrics for each Evaluation Domain 92

4.2 IPD Relaxation Results for Domain II 102

4.3 IPD Relaxation Results for Domain III 103

Figures

Figure

1.1 Contribution Overview 6

2.1 The Learning from Demonstration Pipeline 10
2.2 CC-LfD Contribution Overview 15
2.3 Examples of an 'Upright' constraint's reference pose, tilt tolerance, and violation. . . 17
2.4 CC-LfD Algorithm - Data Collection and Constraint Annotation 19
2.5 Dynamic Time Warping (DTW) example 20
2.6 CC-LfD Algorithm - Alignment 21
2.7 CC-LfD Algoirthm - Modeling 22
2.8 CC-LfD Algorithm - Rejection Sampling and Model Relearning 22
2.9 Demonstration Trajectory Clustering 24
2.10 CC-LfD Algorithm - Final Skill Execution 26
2.11 Kinesthetic Demonstration Example 27
2.12 CC-LfD Evaluation Tasks 31

3.1 Augmented Reality Interfaces for Constrained LfD 38
3.2 Contributions of AR Interfaces to LfD 39
3.3 The Human-Action Cycle adapted for LfD 41
3.4 ARC-LfD Visualization Examples 48
3.5 ARC-LfD System Diagram 50

3.6 ARC-LfD Interaction Flowchart . . . 51
3.7 ARC-LfD Editing Menu Example . . . 54
3.8 ARC-LfD Evaluation - Case Study I . . . 55
3.9 ARC-LfD Evaluation - Case Study II . . . 56
3.10 ARC-LfD Evaluation - Case Study III . . . 57
3.11 Instrumented Tong for ARPOC-LfD . . . 59
3.12 ARPOC-LfD Architecture Diagram . . . 60
3.13 ARPOC-LfD Interaction Flow Diagram . . . 61
3.14 ARPOC-LfD Evaluation Task I: Mailbox Opening . . . 65
3.15 ARPOC-LfD Evaluation Task II: Glue Tracing . . . 66
3.16 ARPOC-LfD Evaluation Task II: Stacking . . . 67

4.1 IPD-Relaxation Contribution Overview . . . 72
4.2 Topological Configuration Space . . . 73
4.3 Manifold Projection Method . . . 76
4.4 Charts Creating a Sphere . . . 77
4.5 Atlas-Chart Mappings . . . 78
4.6 Task Space Region Constraint Framework . . . 80
4.7 Sequential Manifold Planning Example . . . 82
4.8 Differentiating Intersection Point Independence from Dependence . . . 84
4.9 How CC-LfD Solves an SMPP . . . 89
4.10 2D Navigation Environment . . . 96
4.11 Evaluation Domain III for IPD-Relexation . . . 98
4.12 Results: Sampling Efficiency Graphs . . . 100

5.1 Contribution Overview . . . 107

Chapter 1

Introduction

> "Expert knowledge is intuitive; it is not necessarily accessible to the expert himself."
> - Donald Michie [87]

The late Donald Michie, a lesser-known forefather of Artificial Intelligence breaking code alongside Alan Turing at Bletchley Park, established a remarkable career that ranged across computer science, biology, and genetics. At the University of Edinburgh, he formed the team that developed the Freddy I and II robots, pioneering computer-vision-assisted manipulation arms that used object recognition for sorting and sequenced assembly. More than 50 years ago, his team introduced a robot system capable of responsive sequenced behavior to object recognition; a remarkable feat given the limitations in computing technology at the time. Despite decades of technological progress, modern-day vision-assisted manipulation systems have not radically improved in their ability to identify, sort, and assemble objects compared with these pioneering Freddy systems. No doubt, such systems are more capable as evidenced by the proliferation of practical and useful robotics companies worldwide. However, computation power has nearly doubled annually since the time of Freddy. Robotic control systems, software, and hardware have substantially improved enabling impressive physical feats like the legged robots of Boston Dynamics. The rise of applicable statistical and machine learning methods has greatly improved vision systems facilitating the proliferation of autonomous vehicles. Given all these impressive advances, one might expect dramatic paradigm-shifting advancement of manipulation systems. This begs the question, *"Why have robot manipulation systems not experienced the same level of technological improvement afforded to other*

areas of information technology?"

One potential answer lies within the idea that implicit intuition and latent intellectual requirements behind simple tasks belie capability that should be reachable to modern-day autonomous robotic systems. Years ago Michie's team came to a similar sort of understanding: they realized that the ability to convey the high-level intuition of a task to the Freddy system required a higher-level form of programming language (RAPT or Robot Automatically Programmed Tool) that better enabled these researchers to outline the required robot behavior for successful assembly. In other words, they needed a new language to better capture the intent and goal of the task. In the same spirit, robot systems intended to perform complex tasks in context-rich environments need mechanisms to integrate richer and more human-intuitive forms of information. Without such information, robot systems will struggle to improve in capability in a way that differentiates their abilities from the ancestral Freddy robots and to advance autonomous systems in profound and broadly applicable ways.

At face value, this idea of integrating and curating information into machine intelligence is not a strictly new concept. To date, foundational machine learning methods center around selected features chosen for training data that intuitively capture the essence of what must be learned. As summarized in an article by Scholkopf et. al.,"...the majority of current successes of machine learning boil down to large scale pattern recognition on suitably collected independent and identically distributed (i.i.d.) data."[127]. But such an approach, while powerful, presents limits to the performance capabilities of such models. Much effort must be made in order to avoid biased machine learning models [94], motivated by the desire to generalize to broader applications. Conditioning data is another approach to generating more widely usable models, often through a Bayesian approach. In these approaches, the choice of prior depends on broad assumptions about the nature of the distribution of your data (i.i.d., normal, binomial, etc,.). Learning priors can avoid reliance on such assumptions, but requires large-scale data once again, ultimately resulting in a different flavor of the models summarized by Scholkopf (perhaps with some performance gains).

Through comparison with intelligent animals' ability to generalize, Scholkopf emphasizes

that "...machine learning often disregards information that animals use heavily: interventions in the world, domain shifts, temporal structure – by and large, we consider these factors a nuisance and try to engineer them away." In other words, Scholkopf makes the argument that the nature by which machine learning models discover patterns in data often ignores context and the causal structure of a given problem domain. Fortunately, recent advances in machine learning have started to incorporate some of the types of structures Scholkopf argues machine learning historically has avoided. For example, temporal attention structures in large language models enable GPT-3 [25] and Google's PaLM [35] to produce sensational text-generation abilities. Retrieval-Augmented Generation methods [89] rely on situational context drawing on knowledge graphs to condition models. Xie et. al [153] integrates parameterized physical models for model-based reinforcement learning in robotics.

The integration of other forms of information (be it structural, temporal, etc,.) could have broad implications for the capability of autonomous robotics that rely on statistical and machine learning methods for behavior generation. A persistent challenge with the application of traditional machine learning methods in robotics is their inherent limitations in generating large-scale data from which to learn performant behavior models. This limitation is either physical, as the robot cannot feasibly explore a problem space (say for the purposes of Reinforcement Learning) in a time-efficient manner, or translational, as simulation environments that enable large-scale trials can fail to transfer to the physical world. As hinted by Xie et. al, the introduction of structure, in the form of models or conditions on behavior, might enable autonomous robots to avoid this problem of data sparsity and allow for more generalized behavior models that can adapt to situational context.

For autonomous robotics systems, and particularly those meant to collaborate closely with human users, this additional context becomes paramount. Human users often possess knowledge about a task that is not known nor communicated to the collaborating robotic agent [140]. Sharing context between humans and robotic systems in such settings has the potential to improve both human and robot performance alike [139]. This requires the robot to share information with the human teammate and vice-versa.

As such, this dissertation presents a series of novel works involving algorithm design and interface development that supports the transfer of information drawn from human expertise to expand upon the abilities of autonomous robotic agents to quickly acquire and generalize skills. More specifically, it describes how the communication and encoding of behavioral constraints serve as a new approach to integrate additional structured information used to create a novel statistical robot learning methodology that utilizes human-provided examples as training data. It introduces new interfaces through which human experts more easily provide examples and communicate these behavioral constraints. And lastly, it describes how this learning algorithm both defines and helps to solve a challenging form of constrained robotic motion planning.

1.1 Technical Motivation

One of the most fundamental capabilities of robotic agents in the automation of task execution is their ability to produce feasible and useful motion. However, this motion is only useful if it achieves the behavior intended by the human operator/programmer/user. Traditional robotics have met behavior criteria through explicit programming. For example, in robotic car manufacturing, physical constraints on arms are well-scoped such that programmed behavior stays within safe limits. While this often suffices, an explicitly programmed behavior essentially acts as a single instance drawn from the space of possible solutions. However, when faced with dynamic task requirements, accommodating constraints and task goals becomes an expensive endeavor as the robot may require extensive reprogramming for a given change. An alternative approach is to utilize large-scale simulation to generate the data needed for data-intensive models. However, simulation environments do not always model the intended execution environment due to physics engine inaccuracies (e.g. the friction needed to carry an item) or the complexity of the environment itself (e.g. safely modeling human beings).

Robot Learning from Demonstration (LfD) techniques strives to make the training and re-training of robots accessible to non-experts by removing the need for real programming in lieu of layman-friendly modes of interaction and teaching. The underlying narrative of this dissertation is

that by integrating the communication of behavioral constraints into robot LfD methods, we can enable users to specify the task, make the resulting learned model more robust to dynamic changes in requirements, open up new modes of information exchange through state-of-the-art interfaces, and fuse LfD with constrained motion planning methods. Likewise, by relying on the intrinsic domain expertise of the demonstrator, the ideal outcome is a model that requires little data since the user is providing highly accurate examples of correct behavior while bounding the space of possible learned models with constraints.

1.2 Thesis Statement

With this motivation in mind, this dissertation argues that the incorporation of human-provided behavioral constraints into robot Learning from Demonstration techniques serves as a mechanism to achieve context-rich robot automation through the fusion of Robot Learning from Demonstration, novel interface design, and Constrained Motion Planning. More specifically, the dissertation outlines how additional task context in the form of these behavioral constraints acts as a basis for novel Learning from Demonstration methods, offers avenues to expand LfD interface design, and fosters a fusion of constrained LfD with constrained motion planning. The benefits include a reduced reliance of autonomous robotic systems on programming expertise, and the acceleration of robotic capability to more quickly acquire learned skills that generalize to broader applications.

1.3 Contributions

The summarized contributions contained within this dissertation are as follows:

(1) A novel robot Learning from Demonstration method called **Concept Constrained Learning from Demonstration (CC-LfD)** that enables users to provide problem context through the form of behavioral constraints.

(2) Two augmented reality systems, **Augmented Reality for Constrained Learning from**

Demonstration (ARC-LfD) and **Augmented Reality-based Pose Optimization for Constrained Learning from Demonstration (ARPOC-LfD)**. The first enables users to visualize, edit, and annotate keyframe LfD models with constraints in a mixed-reality setting. The second enables the use of a data-tong device for online pose-optimization-based teleoperation that provides feedback as a means to produce more optimal constrained demonstrations.

(3) An algorithm called **Intersection Point Dependency Relaxation (IPD-Relaxation)** for solving constraint-varying motion planning problems (such as those defined by the CC-LfD model) by utilizing keyframe distributions from demonstrations as a heuristic in an optimization process, thereby generating constraint-compliant motion for the entire learned task.

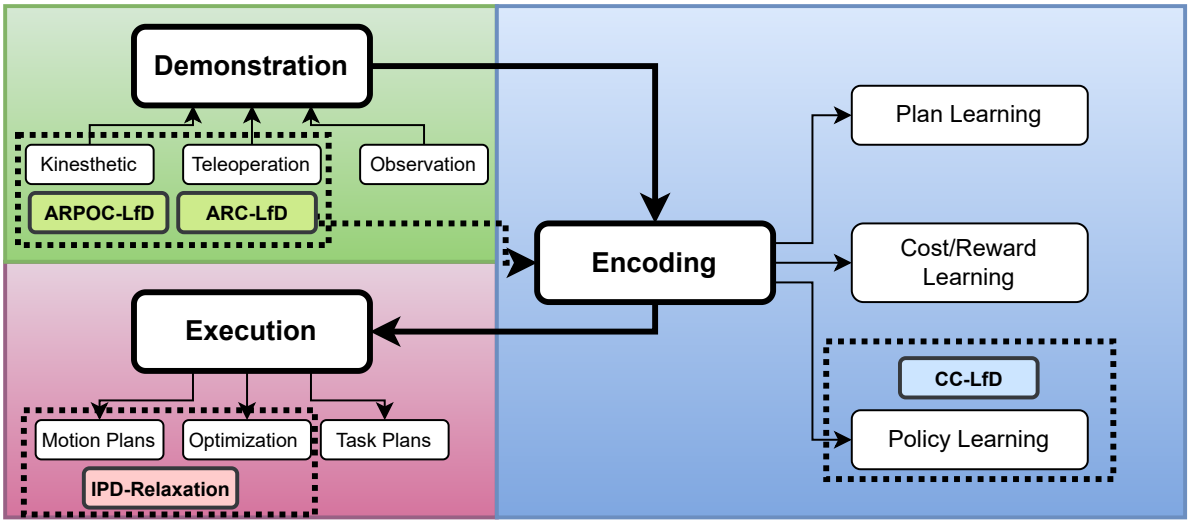


Figure 1.1: An overview of where the contributions presented in this dissertation fit into the general Learning from Demonstration pipeline introduced by Bakker et al. [13]. Learning from Demonstration requires a mechanism for collecting demonstration data (green region), creating an encoding of that data (blue region), and using the encoding to generate robot execution (pink region). The dotted lines group/link the system and algorithm contributions of this dissertation with corresponding major topic areas outlined by Ravichandar et al. [120].

As such, Figure 1.1 presents a broad overview of the Learning from Demonstration pipeline defined by [13]. In this pipeline, demonstrations are provided to an encoding scheme that generates

a model of the learned skill. This encoding provides the means for agent execution of behavior. The figure indicates how the systems and algorithms above contribute to the three main areas of this pipeline: 1) Demonstration, 2) Encoding, and 3) Execution. The coming chapters will detail how these algorithms and systems each contribute to these respective areas.

1.4 Dissertation Outline

This document consists of six chapters. The middle three chapters constitute the bulk of the dissertation and outline the three major contributions listed above. The chapters are described below:

(1) Chapter 1 provides an introduction, motivation, and outline of the dissertation.

(2) Chapter 2 describes related work surrounding Robot Learning from Demonstration methods with a specific focus on how such methods have been augmented by additional context to enrich resulting models. It then continues to describe an algorithm called *Concept Constrained Learning from Demonstration*, or CC-LfD, that incorporates concept-representative behavioral constraints into an LfD technique called Keyframe LfD.

(3) Chapter 3 provides the appropriate background necessary for the two Augmented Reality systems and then describes how these two systems create a holistic interface that supports human-provided demonstration, behavior constraint communication, visual feedback to human users for model verification, and informed pose-optimized teleoperation.

(4) Chapter 4 provides the necessary background and related research on constrained motion planning leading to a discussion about how CC-LfD enables users to define a challenging constrained motion planning problem called *Sequential Manifold Planning* and provides a means to solve that problem through an algorithm called *Intersection Point Dependence Relaxation*.

(5) Lastly, Chapter 5 concludes the document by summarizing the contributions of these works

and with a discussion about the avenues of future work that these novel contributions unlock.

Chapter 2

Constrained Robot Learning from Demonstration

Learning from Demonstration (LfD) methods enable robots to rapidly gain new skills and capabilities by leveraging examples provided by human operators [13]. While effective, this training mechanism presents the potential for sub-optimal demonstrations to negatively impact performance due to unintentional operator error. Similarly, even quality demonstrations might not properly encode import features of the task intended for learning. In either case, the resulting model may be deficient and produce agent behavior that violates the expectation and intent of the human trainer. To overcome this deficiency, the novel algorithm introduced in this chapter, *Concept Constrained Learning from Demonstration (CC-LfD)*[1] , enables robust skill learning and skill repair that incorporates annotations of conceptually-grounded constraints (in the form of planning predicates) during live demonstrations into the LfD process [98].

This chapter will first supply the necessary background information for an understanding of this algorithm. It will then show through an evaluation that CC-LfD can be used to quickly repair skills with as little as a single annotated demonstration without the need to identify and remove low-quality demonstrations. It will also show evidence for potential applications to transfer learning, whereby constraints can be used to adapt demonstrations from a related task to achieve proficiency with few new demonstrations required. Lastly, it will outline how this algorithm necessitates research into novel Constrained Motion Planning algorithms and novel interface design for communicating constraints and visualizing learned models.

[1] Note that the information presented in this chapter draws upon, paraphrases, and uses text verbatim from Mueller et al. [98].

2.1 Robot Learning from Demonstration Preliminaries

2.1.1 What is Learning from Demonstration?

Robot Learning from Demonstration (LfD) encompasses an extensive set of techniques that enable robots to learn skills from human guidance [11]. The terms learning from demonstration [10, 32], programming by demonstration [20], apprenticeship learning [1], and imitation learning [60] all tend to refer to this same high-level concept within robotics literature. Figure 2.1 outlines the general LfD pipeline defined by [13] and refined with categorizations from [120]. The general sequence involves 1) Demonstration, 2) Encoding, and 3) Execution. A human teacher typically performs 'ground truth' *demonstrations* in the form of state trajectories or goal states that convey the 'what' and/or 'how' of a skill. Ideally, demonstration communicates the nature of the skill to the robot system, which *encodes* a model that approximates some latent ground truth model held by the demonstrator [10]. This encoding is then used to generate a trajectory of controls or states by which the robot *executes* behavior that ideally recreates the intended behavior as demonstrated by the user.

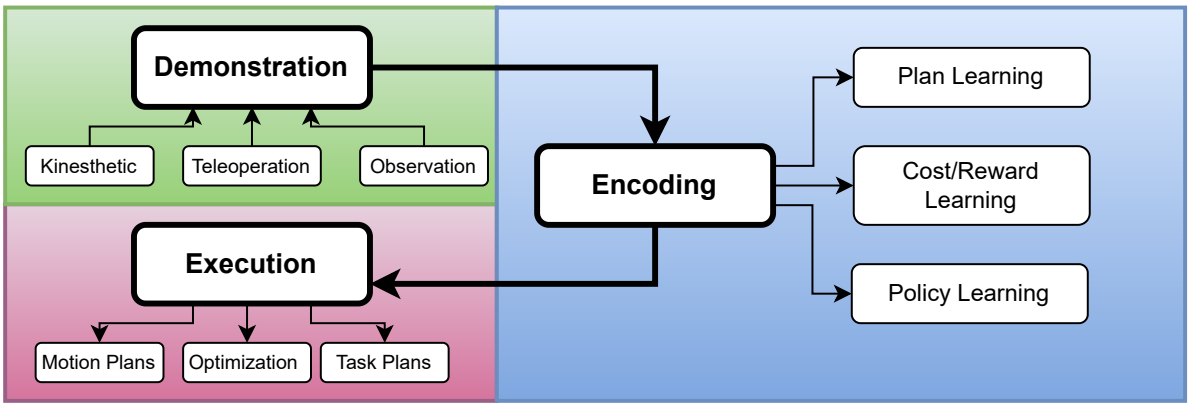


Figure 2.1: Adopting the general pipeline from [13] and incorporating specifics from [120], this diagram outlines the major components of the Learning From Demonstration pipeline: 1) demonstration of data, 2) an encoding that produces a model, and 3) the execution of that model.

2.1.2 Modes of Interaction

This first step of the LfD pipeline outlined above is for a user to provide demonstrations. There are many modalities of human-robot interaction with respect to robot skill learning from human users, the most popular being passive observation, teleoperation, and kinesthetic demonstration. Passive observation requires that the robotic learner possess the ability to externally observe their human teacher without direct physical interaction [142]. This introduces the correspondence problem [100, 22] where the learning system must discover, or be given, a useful mapping between the external state of the human or device (e.g., motion tracking markers) and the control space of the robot (e.g., configuration space, task space, accelerations, torques, etc,.). Robots and their human teacher cohabit in a shared environment, but they perceive and interact with this environment differently, which necessitates this mapping between the state of the human (perhaps perceptually, physically, or both) and the robot. Teleoperation is a more direct means of interaction where users control the robot through some device (e.g., a joystick). While this mode bypasses the correspondence problem, it does introduce the difficulty of designing a human-intuitive device to control high degree-of-freedom robots [6]. The most direct mode of interaction is kinesthetic demonstration where human operators physically manipulate the robotic device while the robotic learning system passively captures state information. With respect to redundant robot manipulators, this mode of demonstration has been shown to be preferable by users as it offers the most intuitive control over the robot [6]. Kinesthetic demonstration also bypasses the correspondence problem and does not require a difficult-to-use controller [152].

2.1.3 Data Used for Learning

LfD methods use a wide variety of state data for learning dependent on the desired learning outcome. Imitation learning and inverse reinforcement learning methods for mobile robotics generally utilize steering angle and other control inputs such as acceleration and braking forces [130]. For manipulators, force [28] and torque [41] are possible forms, however, a large number of LfD methods

utilize joint angles as the predominant data type [3, 98, 4, 131, 101, 8]. Joint angles often represent the configuration space (and therefore planning space) of robot manipulator arms. Most methods collect trajectories, but this is not a strict requirement. For example, in [3] single points serve as user-supplied keyframes or via-points. Another form of data used in LfD is robot end-effector pose data [106, 37, 130], often called task space. The work presented in this dissertation predominantly uses joint state and pose data as well as environment object pose data.

2.1.4 Characterization of Robot Learning from Demonstration

Ultimately, these modes of demonstration all serve the same purpose: to facilitate the transfer of information from a human user with some expert intuition about the skill to the robotic learning system. This information is used by robot skill learning methods to produce usable learned models of the task (i.e. the encoding step of Figure 2.1). It is helpful to understand that while this is broadly the goal, the mechanisms by which systems learn from human demonstrations utilize various concepts and methods in statistical and machine learning. Ravichandar et.al., [120] outline three major learning outcomes that generally categorize the type of learning methods and representation: 1) plan, 2) cost/reward, and 3) policy learning.

Plan Learning Methods Plan learning methods attempt to learn models that operate at high levels of task abstraction, either learning a primitive sequence or hierarchy [120]. These sequences and hierarchies are often representations of small subtasks or skills that when executed together, create an overall behavior. Assembly tasks often require hierarchical representation. As exemplified by Hayes et al. [58], novel hierarchical task networks are generated by leveraging ordering constraints and demonstration of subtasks from users in order to learn assembly tasks. An alternative approach to hierarchies is to learn robust sequences of skills that enable variations in the final executed behavior. For example, Konidaris et al. [80] presents *CST* (Constructing Skill Trees) which segments demonstration trajectories into skills or task primitives. Sequences of small behaviors on branches of the skill model tree account for variations in the skill itself.

Cost/Reward Learning Methods Cost/Reward methods generate a model through the optimization of a learned or predetermined function, usually called a 'cost/loss' or 'reward' function [10, 120]. As outlined in Ravichandar et al. this approach comes in two major forms: trajectory optimization and Inverse Reinforcement Learning (IRL) [120]. For example, [12] formalizes learning from human-robot interactions as a dynamical system where demonstrations serve as useful observations for learning an objective's functional parameters. This acts as a form of inverse reinforcement learning whereby a reward function is learned from human demonstration.

Policy Learning Methods Policy learning methods expect that there exists a function (i.e. the policy) that produces desired robot behavior. Human demonstrations provide input as some combination of time, state, and/or raw observation. A learning process produces a policy π from these demonstrations. While similar to IRL-based methods, policy outcome learning methods directly learn a policy, not a reward function. This policy outputs either a trajectory of states (often in the same space as the input) or a low-level action plan. Some examples of policy methods include Probabilistic Movement Primitives [103], using a maximum likelihood learning process to generate distributions of demonstration trajectories, employing other model features through methods like primitive coupling through mean and covariance modulation through distribution conditioning.

Perhaps the most pertinent example to this dissertation of a policy learning method is that of Keyframe Learning from Demonstration [3]. This model encodes trajectories into a series of keyframes (sometimes referred to as via-points) [146, 7]. The keyframe model is a directed graph of waypoints (or distributions) for a motion planner to traverse through. These keyframes are either single points directly sourced from demonstration or represented as distributions of state data called Sequential Pose Distributions [3]. Keyframe models provide the flexibility to push more skill execution to the motion planner as the inter-keyframe distance increases (relying on a search process), which allows for automated planning around obstacles, but still retains the stylistic intent of users through learned distributions [146, 109].

2.1.5 Incorporating Context into LfD

Most LfD methods successfully capture the low-level style of demonstration data i.e. they capture well the positional and orientational style of the demonstrations. However, as mentioned above, a major challenge to learning from demonstration is the verification that a learned model will produce behavior according to a user's intent. It is difficult to determine if sufficient training data has been received to cover the breadth of scenarios a skill is expected to perform. One approach is to introduce reparative demonstrations in order to fix the model. For example, Jain et al. [65] introduce an iterative learning method for trajectory improvements where a human demonstrator introduces incremental improvements to retrain the model. Rather than introducing new trajectory data, alternative approaches employ human-in-the-loop skill learning utilizing robot-induced propositions (e.g., labels and feature queries) to repair and more quickly learn skills [27]. Extending this concept, Basu et al. [15] showed that feature queries serve best to facilitate human augmentation of a robot's objective function. Perez et al. introduce a method called C-LEARN that uses demonstration to infer geometric constraints for future model refinement [108].

The above approaches can extend the idea that additional information from human demonstrators can boost learned models. For example, Chao et al. [31] show that encoding human-grounded concepts equips robots to reason better about unfamiliar tasks. This chapter focuses on the algorithm Concept Constrained Learning from Demonstration that introduces the idea of incorporating behavioral constraints into Keyframe LfD [98]. These constraints serve to represent a contextual behavior restriction specific to the task, hence they're called 'Conceptual Constraints'. Conceptual constraints are encoded as planning predicates (e.g., "is_upright(cup)") that filter candidate waypoints sampled from keyframe distributions such that the generated sequence adheres to these constraints. This approach enables users to communicate multiple constraints in addition to the provided demonstration data, providing a richer set of information than trajectories alone can encode. This enables the creation of a model that more closely approximates the 'ground truth' representation of the task.

The learning of behavioral constraints from demonstration within the human-robot interaction community has predominantly focused on task ordering constraints [57, 58, 43]. However, CC-LfD enables users to provide multiple grounded constraints that can both represent low-level motion constraints (e.g., "keep the end-effector parallel to the table") as well as high-level concepts (e.g., "don't let the cup of water and the laptop be near each other") that are pertinent to a successful behavior producing model. CC-LfD generates such models be enforcing constraint adherence into the keyframe sampling process. It should be noted that the motion planning between constrained keyframe waypoints relies on constrained motion planning methods. As will be shown in upcoming chapters, the ability for users to specify sets of changing constraints over the length of the skill introduces a challenging constrained motion planning problem.

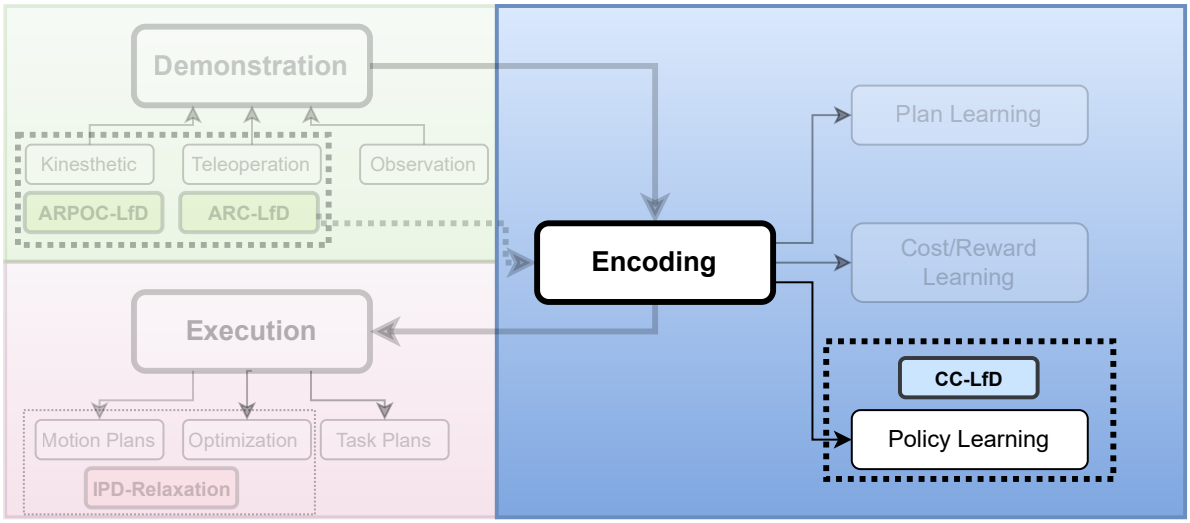


Figure 2.2: Referring back to Figure 1.1, the CC-LfD algorithm is a novel method to encode demonstration data akin to Policy Learning methods.

2.2 Concept Constrained Learning from Demonstration

This section outlines the first major contribution of this dissertation: *Concept Constrained Learning from Demonstration* (CC-LfD). CC-LfD is a method for learning and repairing skill policies with minimal additional demonstrations. As an extension of Keyframe LfD, CC-LfD classifies as a policy learning method of encoding in the general LfD pipeline (see Figure 2.2). Central to this

method's success is the hypothesis that physical trajectory demonstrations alone are a relatively low-bandwidth signal as compared to the fusion of trajectories with abstract concepts in the form of planning predicates [124]. By introducing a method to propagate constraints from constraint-annotated demonstrations into the entirety of a skill's training set, the approach achieves rapid skill repair and robust skill learning from demonstration, even if the initial training data or skill model contains errors. Through our experiments we show that our method facilitates robust skill learning from demonstration, providing a dramatic reduction in the training data required for skill repair as compared to introducing additional high-quality trajectories.

The three primary contributions of this work are:

- Concept Constrained Learning from Demonstration (CC-LfD), an algorithm for few-shot robust skill learning from demonstration
- An application of CC-LfD to skill repair, enabling CC-LfD to make existing skills more robust to failure with minor additional effort.
- An experimental validation of CC-LfD for skill repair and transfer learning implemented on a manufacturing robot.

2.2.1 Conceptual Constraints

CC-LfD employs behavioral constraints called *Conceptual Constraints* in the form of logical formulae of Boolean state classifiers (e.g., on_table(cup)). These constraints are generally represented as task-space (i.e. position and orientation) constraints that represent problem-specific concepts that are important to the execution of the skill (see Figure 2.3). Their associated Boolean classifiers are used as rejection sampling filters to bias keyframe distributions. This chapter will outline the training and execution phase of CC-LfD by providing a general intuition for each.

(a) *Reference Pose* **(b)** *At the Tilt Tolerance* **(c)** *Constraint Violated*

Figure 2.3: Examples of an 'Upright' constraint's reference pose, tilt tolerance, and violation.

2.2.2 The CC-LfD Algorithm and Model

2.2.2.1 Demonstration Trajectory Preproccessing

The first phase of the CC-LfD algorithm is a demonstration trajectory preprocessing step. In this step, trajectory demonstrations are collected from a user on which constraints are annotated. These trajectories undergo an alignment step to link like segments of the trajectory. These aligned trajectories will serve as the basis for proceeding with clustering and distribution fitting steps.

***Demonstration Collection and Constraint Annotation*:** The first phase in any LfD method is the collection of a training data-set that is representative of a skill a user intends the robot to learn. To accomplish this CC-LfD requires a collection of trajectories $C = \{T_0, T_1, \ldots, T_{|C|}\}$, which serves as the input for Algorithm 1. A trajectory is defined as a sequence of tuples (or observations) called frames $T = \{f_0, f_1, \ldots, f_{|T|-1}\}$, each containing a time-stamp (t), a vector of world state data (s), and the set of annotated constraints c, such that $f = (t, s, c)$. The world state vector consists of the robots joint configuration, end-effector position and orientation. This state vector can include any pertinent information about the task such as the position of other objects in the environment relative to the end-effector.

Algorithm 1: Trajectory Preprocessing

```
   Input: Collection of recorded trajectories C
   Output: Labeled and Aligned Trajectories L
 1 L, trajectoryLength ← alignWithDTW(C);
 2 for i in range(trajectoryLength) do
 3 |   activeConstraints ← {};
 4 |   for T in L do
 5 |   |   activeConstraints ← activeConstraints ∪ getConstraints(L[i]);
 6 |   end
 7 |   for T in L do
 8 |   |   applyConstraints(L[i], activeConstraints);
 9 |   end
10 end
   /* All demonstration trajectories now have same constraint sequence.      */
11 return L
```

During demonstration, CC-LfD allows users to annotate a trajectory with constraint information. Annotation means that a user communicates when a constraint should hold true in the form of Boolean variable on the observations of a trajectory T. These constraint variables combine together to create Boolean expression (e.g., "is_upright(cup) AND over_spillway(cup)"). Each variable in these expressions represents a predefined concept and the combination of these concepts generates multi-constraint conditions on the associated segment of the demonstrated skill. Annotation can be accomplished through any interface that can enable a user to communicate when constraints must apply to the current segment of the demonstration. Chapter 3 will describe an Augmented Reality system that enables annotation, editing, and visualization of constraints.

CC-LfD supports two modes of constraint annotation: 1) constraint propagation and 2) constraint toggling. Constraint propagation means that when the behavior of the demonstration violates the constraint, the system turns that constraint to false. As an example, a constraint '*is_upright(cup)*' applied at frame f_a would propagate to all proceeding frames until it is found to be violated by the demonstration at f_b (where $b > a$). An advantage to this approach is that the user only is concerned with assigning constraints at the start of their application, and can demonstrate a violation to turn them off. The disadvantage is that this requires users to successfully demonstrate the skill without premature violation.

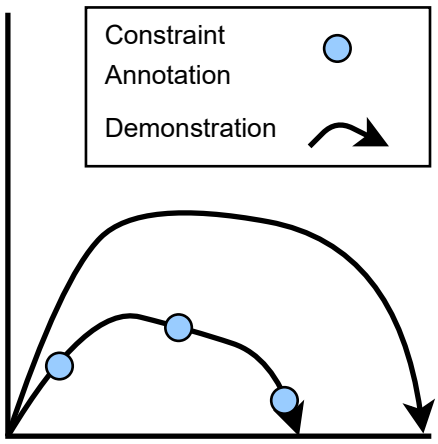


Figure 2.4: The data collection phase of CC-LfD acquires constraint annotated trajectories from human users.

Constraint toggling requires the users explicitly toggle constraints on and off, and frames will be annotated with a constraint until a user toggles that specific constraint off. The advantage of this approach is that users can explicitly decide when constraints apply and their application is no longer dependent on the quality of the demonstration, but with a potentially higher cognitive load. Regardless of the method of annotation, the result is the production of intervals on the demonstration trajectory where Boolean expressions of conceptual constraints must be satisfied by robot behavior.

***Trajectory Alignment*:** Before generating any model, it should be noted that trajectories are not guaranteed to maintain a temporal alignment [143, 59, 36] (e.g., $f_{15} \in T_0$ may not represent the same point in skill execution as $f_{15} \in T_1$). To overcome this issue, a trajectory alignment step (Algorithm 1, line 1) that utilizes Dynamic Time Warping (DTW) [125] is used to link 'equivalent' points between trajectories. Figure 2.5 provides an example of this alignment process, showing how the shift in each trajectory is captured by the alignment indicators. One important consideration in the alignment approach of CC-LfD is the alignment and ordering preservation of constraint boundaries or constraint transitions. A constraint boundary is a frame $f_i \in T_n$ where the currently applied set of constraints changes. This could mean a new constraint is added or an existing one is removed or altered at that frame.

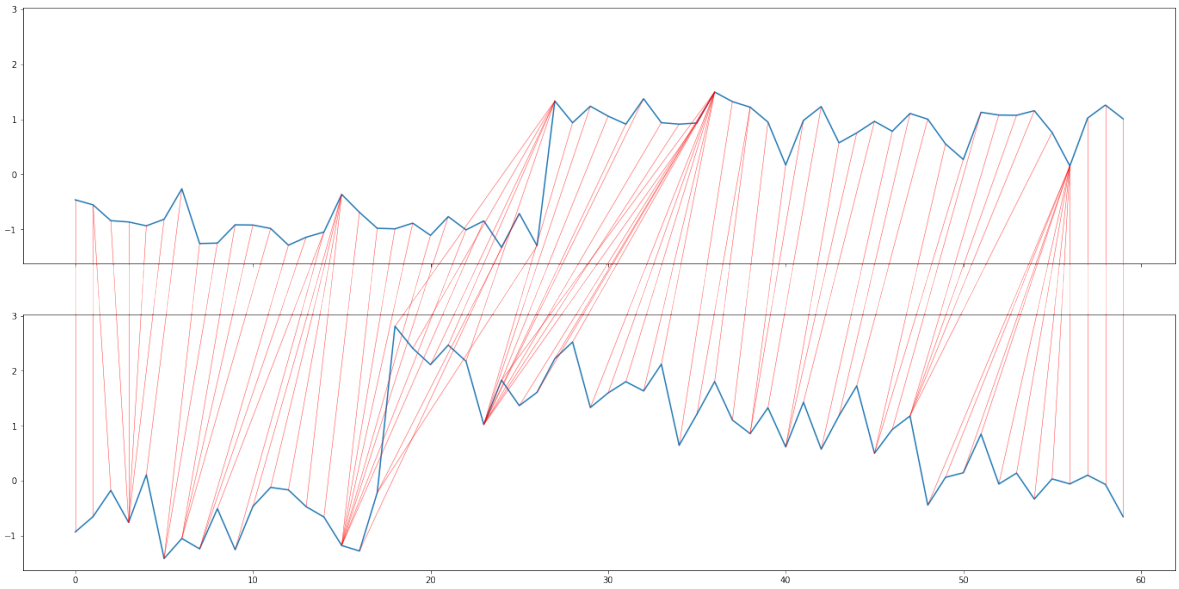


Figure 2.5: An example of 2D trajectory data aligned using Dynamic Time Warping. As evident in this figure, the step-up in each trajectory is well-aligned via this algorithm. This alignment process is used to align demonstration trajectories in the space that CC-LfD models (e.g. task space or configuration space).

Demonstration trajectories are aligned against a chosen reference demonstration using standard DTW [69] with a Euclidean distance cost function. An iterative alignment process where each trajectory is rebuilt based on the warping path provided by the DTW algorithm results in repeated points for certain trajectories that may align with multiple points in another trajectory. This iterative process repeats the DTW alignment procedure on the extended trajectories until those trajectories have been extended to an equivalent length (i.e. the same number of samples in each aligned trajectory). This simplifies the clustering of data points into groups. Repeated points can be tracked and discarded if desired.

Once the collection of trajectories are aligned, constraints are combined across trajectories as a Boolean expression consisting of the logical *AND* of all constraints (both applied and propagated) occurring at that frame index (see Algorithm 1, lines 2-10 and Figure 2.6). Thus, for a Boolean expression of constraints $b_{m,n}$ occurring at frame f_n of trajectory T_m, we apply [98]:

$$b_{m,n} = \bigwedge b_{p,n} \forall p \in \{0, 1, .., |C| - 1\}$$

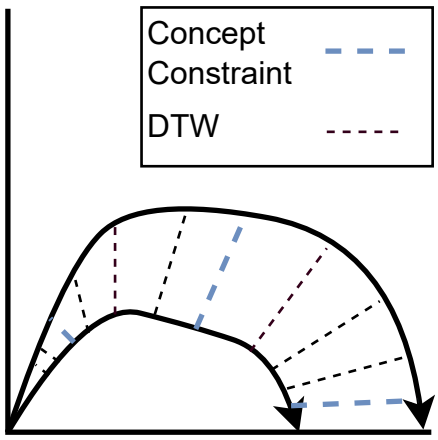


Figure 2.6: A DTW alignment process links like portions of the demonstrations, preserving and propagating constraint annotations across all demonstrations.

2.2.2.2 Model Formation

The next step in the CC-LfD algorithm is the generation of an intial model learned over the aligned and labeled trajectory data. This step first requires an automated clustering step where sequential groups of data across trajectories (linked via the DTW alignment) are used to learn distributions. This distribution, along with metadata like assigned constraints, represents a single keyframe. These sequential keyframes are linked together in a sequential graph structure to create what is called a *Sequential Pose Distribution* graph as introduced in [3]. Each keyframe undergoes a rejection-sampling process to generate a new population of constraint-compliant points sampled from the original learn distribution. A new distribution is fitted to this population, shifting the keyframe model towards constraint compliance. To avoid too much distributional overlap, a keyframe culling process provides sparsity to the graph model.

Keyframe Clustering: This collection of aligned trajectories, also annotated with constraint transition information, is used to produce a directed graph where each vertex represents a keyframe: a distribution over state space used to generate waypoints for skill execution (Algorithm 2, Lines 1-2), . To learn these distributions, the CC-LfD algorithm utilizes the alignment to create clusters taken orthogonally across trajectories, grouping linked segments of data according to the

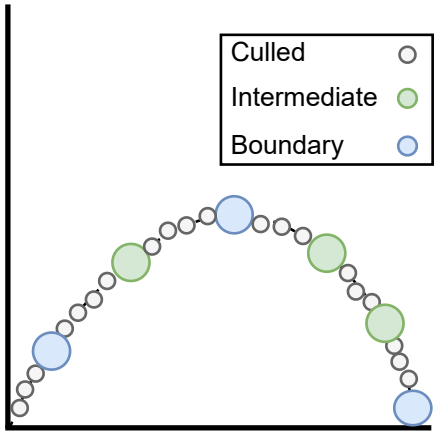


Figure 2.7: Once aligned, demonstration data is grouped sequentially to form Sequential Pose Distributions. A sparsification step culls adjacent intermediate pose distributions to avoid backtracking behavior, but never culls boundary or constraint transition keyframes.

DTW alignment. This approach extends traditional clustering by accounting for annotated constraint expressions through the creation of *boundary keyframes* consisting of trajectory data from a fixed-size window around frames that lie on a *constraint transition* boundary. As mentioned previously, constraint transitions occur when the applied Boolean constraint expressions for consecutive frames differ.

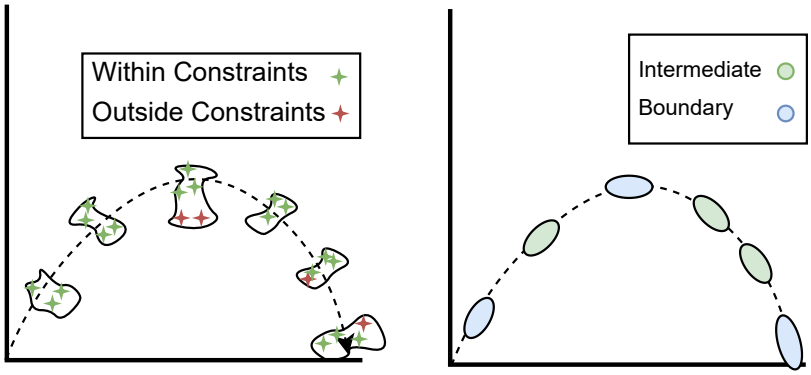


Figure 2.8: A rejection sampling procedure (left) shifts the sequential pose distributions by collecting constraint valid points according to the assigned constraints. Relearning new distributions from these points better reflects constraint compliance and produces a more robust skill representation (right).

Algorithm 2: Model Formation

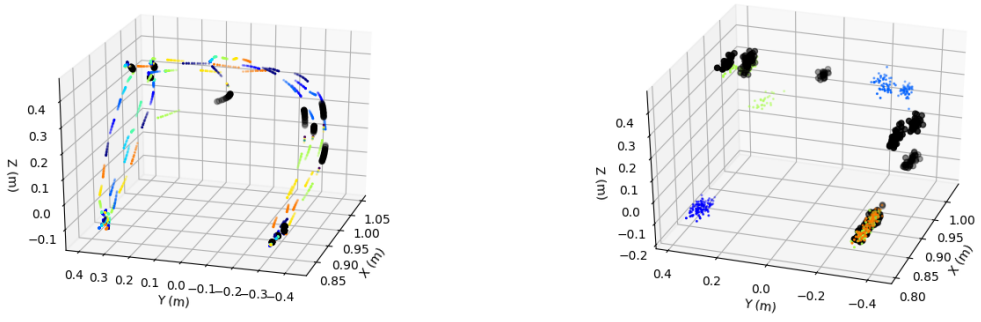
Input: Labeled and Aligned Trajectories L, Variational Distance Threshold α
Output: Keyframe Graph G

```
1  kf_groups ← clusterObservationsIntoKeyframe(L);
2  kfs ← buildGaussianKernelDensityModels(kf_groups);
3  for kf ∈ kfs do
4  |   p ← generateSamplePointsFromModels(kf);
5  |   p ← discardConstraintViolations(p, getConstraints(kf));
6  |   kf ← buildGaussianKernelDensityModel(p);
7  end
8  G ← buildDirectedGraph(kfs, L);
   /* Build directed graph from keyframes using ordering information in L    */
9  p ← generateSamplePointsFromModels(G);
10 G ← cullAdjacentOverlappingKeyframes(G,p,α);
   /* Remove intermediate keyframes that have a variational distance < α with
      a preceeding neighbor                                                  */
11 return G
```

It is important to note that with the creation of constraint transition keyframes, the keyframe model is afforded the flexibility to push representation towards constraint changes by utilizing only boundary keyframes or continue to capture the stylistic aspects of the demonstration through the inclusion of intermediate keyframes (see Figure 2.8, right, and Figure 2.13). Intermediate keyframes are produced by clustering frames at uniform intervals between boundary keyframes. Each intermediate keyframe inherits the constraint requirements of the most recent prior boundary keyframe so that constraint adherence is still maintained on a keyframe-by-keyframe basis until another boundary keyframe is reached, indicating a different set of constraints will apply moving forward down the keyframe chain.

As previously mentioned, CC-LfD models each keyframe cluster as a probability distribution over state space that is trained on the cluster's frames/observations. In the reference implementation introduced in Mueller et al. [98], the distributions are generated using Gaussian Kernel Density Estimation [104, 122]. The chosen bandwidth parameter value [98] maintains the majority of probability mass in close proximity to the observed frames. However, this kernel bandwidth value serves to balance the model between fitness to the cluster data and a corresponding increase in sample variation to increase the efficiency of rejection sampling. This increases the flexibility of

the distributions but at the expense of an increased likelihood of poor skill reconstruction during execution.



(a) Grouping of trajectory points by keyframe. Black points indicate constraint transitions, while colored points indicate intermediate keyframes.

(b) Keyframe clustering showing sparseness after the culling process. These clusters serve as the data for learning Sequential Pose Distributions.

Figure 2.9: *Left:* Visualization of robot trajectory data (projected into XYZ-space) being clustered into keyframes (varying colors show grouped data). *Right:* The remaining keyframe data after culling and augmentation.

Keyframe Sparsification: One of the unfortunate behavioral side effects of Sequential Pose Distributions (i.e. distribution-based keyframes) is potential backtracking where neighboring sampled waypoints result in backward or unintended robot behavior during skill execution. To overcome this, the CC-LfD performs a culling step that deletes intermediate keyframes whose distributions are too close to an immediately preceding neighbor in the keyframe graph (Fig. 2.9 and Algorithm 2, Line 11). This overlap is determined by a prior minimum threshold of variational distance ($\delta(k_i, k_{i-1}) > \alpha$) between the two distributions over an equally sized set of points sampled from each. For a set of n points sampled from Keyframe i ($P_{k_i} \sim k_i$) and n points sampled from Keyframe $i+1$ ($P_{k_{i+1}} \sim k_{i+1}$), the variational distance between keyframe distributions is [98]:

$$\delta(k_i, k_{i+1}) = \sum_{p \in P_{k_i} \cup P_{k_{i+1}}} |k_i(p) - k_{i+1}(p)|$$

.

Constraint-based Relearning: Once the initial keyframe graph is built, a secondary process serves to create better-fitting keyframe distributions to the constraints that apply to them. For

each keyframe distribution, n points are sampled from the original learned distribution (Algorithm 3, Line 4). If the sampled point p satisfies the constraints assigned to the current keyframe, it is accepted and added as training data for a new keyframe, otherwise, it is discarded (Algorithm 3, Line 6). There are two main benefits from this relearning process: 1) the need for high-variance bandwidth values is reduced, and 2) the new distribution will more closely approximate the configuration space manifold where the constraint expression is true, increasing the sampling efficiency during skill execution.

Algorithm 3: Skill Reconstruction

Input: Keyframe Graph Vertex Sequence V, Samples per keyframe n
Output: Motion Plan M

```
1 waypoints ← [];
2 for v ∈ V do
3 |   p ← sampleValidPointsFromKeyframe(v,n);
4 |   p ← discardConstraintViolations(p,getConstraints(v).);
5 |   if p == ∅ then return ERROR;
6 |   waypoints.add(maxLikelihoodPointFromSet(p,v));
7 end
8 M ← createMotionPlan(waypoints);
9 return M
```

2.2.2.3 Skill Reconstruction

To reproduce a skill, waypoints from the ordered sequence of keyframes are sampled. Motion planning between these waypoints results in a trajectory for execution. The resulting keyframe model is a learned approximation of a skill, and thus a path through the keyframe graph produces a rough sequential distributional representation that the robot traverses to execute this skill correctly. Once a sequence has been obtained, sampled waypoints from each keyframe are obtained by sampling from each of these distributions (see Figure 2.10), specifically samples that also adhere to the constraints assigned to each keyframe.

One of the motivations for using sequential pose distributions as a keyframe representation is that an environment may not exactly match the training environment. The execution environment could be one with different obstacles, or different target locations. Thus sampled waypoints might

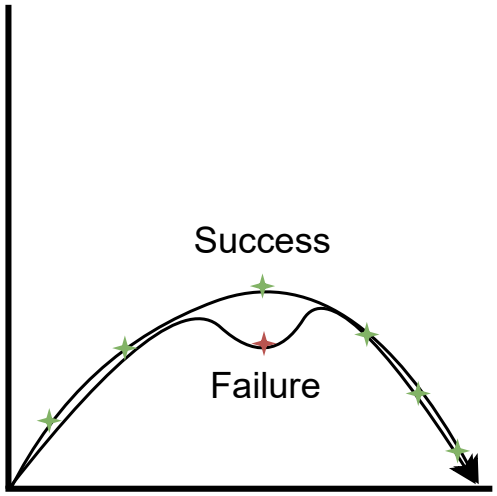


Figure 2.10: A final rejection sampling step produces a sequence of constraint valid waypoints through which to motion plan. Without such constraint compliance, a solution trajectory is apt to fail (lower trajectory).

be invalid (e.g., in collision with an obstacle), be infeasible to use in a motion plan (e.g., no valid solution), or violate the required set of constraints for the keyframe [98]. To overcome this limitation, CC-LfD performs a rejection process of points sampled from the keyframe's distribution, discarding those that violate one of the aforementioned validity criteria (constraints, collision, targeting, etc,.). The valid point with the highest likelihood (relative to its sampled distribution) is retained as a waypoint for the final motion plan (Algorithm 3, Lines 3-6)

If a situation arises where all of the sampled states from an *intermediate* keyframe are rejected, this keyframe is ignored and the next keyframe in sequence becomes the subsequent keyframe. This provides execution flexibility but may potentially sacrifice motion cues encoded by these intermediate keyframes. This approach burdens the motion planner (as it must now plan across a greater distance) while providing more freedom to the final motion plan (still honoring the imposed constraints). An important caveat is that *boundary* keyframes are treated as essential to the skill, and thus if no valid point can be sampled from such keyframes, the skill cannot be executed as the model cannot honor these important constraint transition regions during plan generation. As will be discussed in upcoming chapters, the honoring of constraints throughout the entirety of

the motion plan requires the usage of constrained motion planning methods as standard motion planning algorithms do not necessarily adhere to constraints.

2.2.3 Evaluation

2.2.3.1 Robot Platform

We evaluate CC-LfD using the Rethink Robotics Sawyer robot (see Figure 2.11. Sawyer is a 7-degree-of-freedom robotic arm, with a working envelope of 1260 millimeters and a maximum payload of 4 kilograms. Our CC-LfD reference implementation, including both skill learning and robot control software, is implemented as a node within Robot Operating System [111] utilizing the MoveIt! motion planning framework [33]. [2]

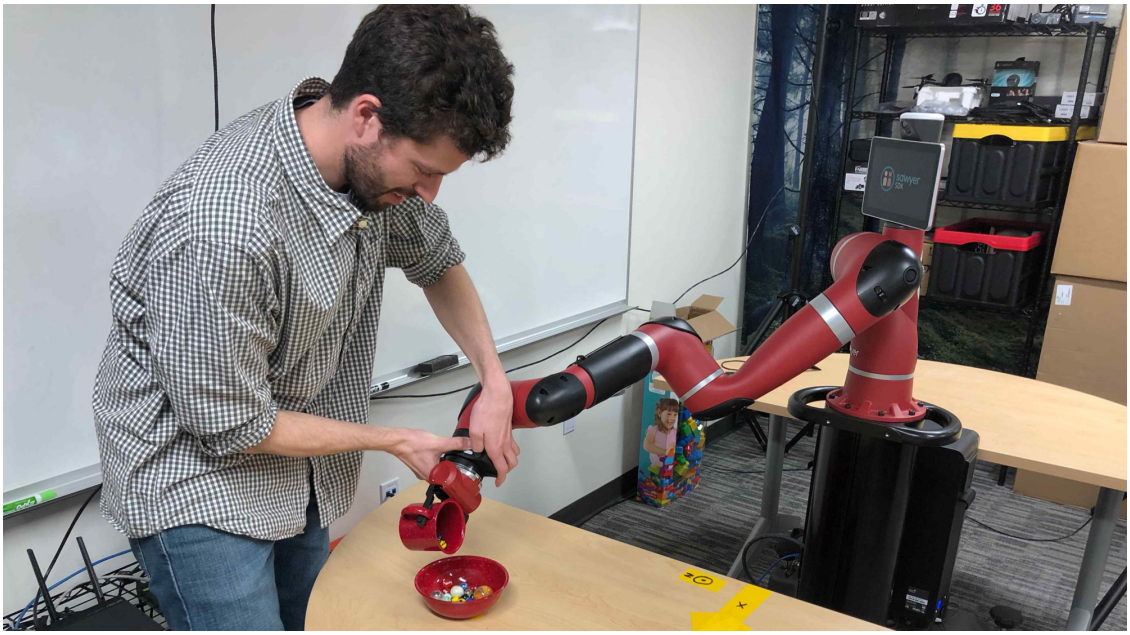


Figure 2.11: A user demonstrating a skill on a ReThink Robotics Sawyer collaborative robot.

[2] CC-LfD reference implementation is available at: https://github.com/cairo-robotics/.

2.2.3.2 Skill Repair from Poor Initial Demonstrations

We demonstrate the utility of CC-LfD through an evaluation involving skill repair, a domain in which the goal is to make a brittle or ineffective skill model more effective and robust to varying environmental conditions. Poor demonstrations may contain useful information to help dictate the skill or they might purposefully provide negative signals i.e. what not to do [53]. Thus there is motivation to show that CC-LfD capably operates over poorly demonstrated skills. This evaluation tests the ability of a learning method to absorb positive aspects of sub-optimal or noisy demonstrations while rejecting aspects harmful to the model, requiring a minimum of additional information.

To demonstrate skill repair with CC-LfD, we utilize two tasks common across skill learning from demonstration literature: a compound pouring task and a precision placement task. The first task involves picking up a cup, moving it over a target region (another cup), and pouring its contents into the target vessel without spilling along the way. This task was chosen due to the complexity of having a constraint that must only be enforced for part of the trajectory (e.g., keep the cup upright) and is violated in other parts. The second task involves picking up a cup, maneuvering around an obstacle not modeled by the motion planner, and resting the cup atop the obstacle. This task was chosen because of its ability to illustrate adherence to motion constraints for part of the task, requiring certain keyframes to obey spatial rules captured by demonstrations without the benefit of analytical models to assist (e.g., the motion planner's collision avoidance). Both tasks are evaluated for success according to their objective behavior which is defined by the intent of the task (e.g. pouring contents of the cup into the target, placing the cup in a resting place) and the expected constraints on the task itself (e.g. cup must remain upright, the cup must not collide with a hidden obstacle).

For each task, three low-quality demonstrations are provided as a baseline of poor performance to which additional demonstrations must be added (these demonstrations may not be excluded from model training) to repair the learned skill model. Generally, a low-quality demon-

stration is one that explicitly violates the chosen constraints for a given task. As an example, in the cup pouring task, a poor demonstration might tip the cup too far before it is over a target, prematurely spilling contents. Other low-quality demonstrations may add harmful variance in the keyframes of a learned skill that is likely to degrade execution quality in naïve LFD solutions. We test two approaches to skill repair:

CC-LfD: Train a CC-LfD model with a single constraint-annotated demonstration and the initial low-quality demonstration dataset.

Naïve LfD: Train a skill model with the addition of successful high-quality demonstrations to the initial low-quality dataset, but with constraints omitted.

In other words, we add high-quality training data to each skill's initial low-quality training set, build a new skill model, and test the model by evaluating its skill executions for success. In the CC-LfD condition, we add a single trajectory annotated with Boolean constraint expressions during demonstration by the human, while in naïve LfD condition we add a number of high-quality demonstration trajectories without constraint information.

2.2.4 Implemented Conceptual Constraints

The evaluation system employs two conceptual constraints to showcase the effectiveness of CC-LfD: an object being 'upright' and a minimum end-effector height. The upright constraint (see Figure 2.3) dictates that an object must be upright according to a predefined upright orientation, reference axis, and a threshold angle of deviation that is object-specific. The upright orientation uses a quaternion representation of the end-effector of Sawyer. In other words, we use a specific grasping orientation to represent the 'uprightness' of the cup rather than the orientation of the cup itself. The environment reference axis defines the axis against which rotation deviations are measured. This axis is generally the z-axis relative to the frame of reference of Sawyer. Axis-angle rotations around the reference axis have no bearing on the 'uprightness' of the object. The threshold angle is the limit within which an object is upright compared with its current angle of

Task Evaluation Criteria	
Type	**# Poor Demos**
Baseline	3 Poor Quality Demonstrations
7 Repairs	Baseline + 7 High Quality Demonstrations
14 Repairs	Baseline + 14 High Quality Demonstrations
21 Repairs	Baseline + 21 High Quality Demonstrations
28 Repairs	Baseline + 28 High Quality Demonstrations
Constrained	Baseline + 1 Constraint Annotated Demonstration

Table 2.1: Evaluation criteria used for both the pouring task and placement task.

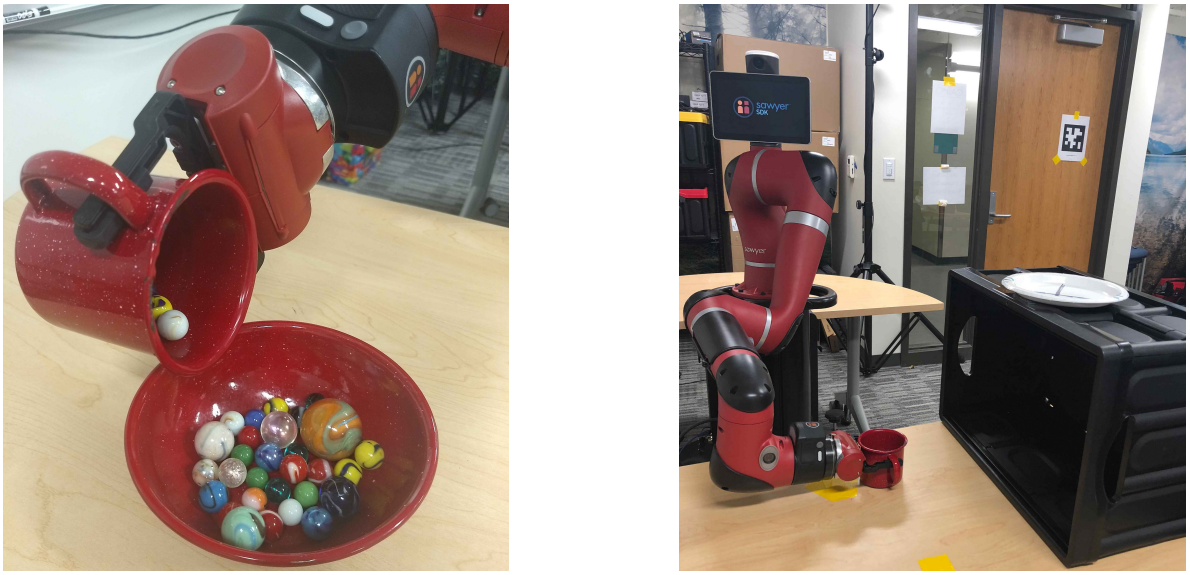
deviation. In the pouring task and placement task, the upright constraint is used to ensure that the cup is not tilted past its upright threshold angle (see Figure 2.3, right-most).

2.3 Results

As the robot can only manipulate its 7-DoF arm, an 'upright' constraint applied to a keyframe forces configurations to be sampled where the object is upright within the robot's grasp, while a 'minimum height' concept forces configurations to be sampled where the end effector pose is a minimum height above the table underneath it. The CC-LfD framework supports any type of concept that can be encoded as a Boolean classifier over state space.

2.3.1 Evaluation Tasks

The CC-LfD algorithm was evaluated using two tasks: 1) A cup pouring task, and 2) A pick and place task. In the cup pouring task, the robot must lift the cup off the table, carry the cup above a certain height until it is over top of a target, then lower and pour its contents into the target without spilling along the way. The task is considered a failure if at any point the robot violates these conditions (see Figure 2.12a a). In the placement task, the robot must lift the cup off the table and place the cup on top of a sideways crate (Fig. 2.12b b). Success for this task requires the robot to place the object on top of the crate without collisions or spills, carrying the cup above a safe height over the crate until it is over top of the placement zone. If at any point before placement, a disqualifying event occurs, the entire task is considered a failure.



(a) Pouring Task **(b)** Placement Task

Figure 2.12: The two evaluation tasks used for the CC-LfD algorithm. *a)*: The Pouring Task required users to carry a cup full of marbles to a target and pour the marbles without spilling. *b)*: The Placement Task required users to move a cup and place it on top of a shelf.

2.3.2 Evaluation Criteria

For each task, we evaluate the success or failure of the robot's performance based on the criteria presented in Table 2.1. We present results from six experimental conditions investigating different levels of skill repair, each consisting of ten trials per task with skill models trained using varying amounts of low-quality (LQ) and high-quality (HQ) demonstration trajectories. An LQ demonstration fails to successfully meet the given task evaluation criteria. An HQ demonstration properly executes the task according to the given task evaluation criteria. A constraint-annotated trajectory is one that is performed correctly while also being annotated with concept constraints.

2.3.3 Results and Discussion

Skill Repair: Our results (see Tables 2.2 and 2.3) show that a single constrained demonstration is enough to repair the poorly trained baseline skill for both tasks, with sufficient variation across executions to guarantee that this is not an artifact of model overfitting. The low-quality baseline demonstrations result in incorrect skill performance nearly always, while a single constrained

Pouring Results			
Type	**# Poor Demos**	**# Repairs**	**Success %**
Baseline	3	0	0
7 Repairs	3	7	40
14 Repairs	3	14	60
21 Repairs	3	21	70
28 Repairs	3	28	60
Constrained	**3**	**1**	**100**

Table 2.2: Pouring Task Evaluation Results. Percentage of successful task executions out of 10 attempts. A single constrained demonstration is adequate to repair the skill given the baseline of three poor-quality demonstrations.

repairing demonstration results in nearly perfect skill performance, maintaining the (allowable) feature variances provided by the baseline training trajectories. The single failure during the constrained condition for the placement task occurred due to a minor obstacle collision that, while not affecting the final placement, violated the collision avoidance success criteria.

Importantly, we observe that while the introduction of additional high-quality demonstrations shows a trend of improvement over the baseline, it does not quickly converge to a high level of success. For both tasks, even twenty-eight unconstrained HQ repairing demonstrations are not enough to overcome the problems the model inherits from the initial LQ training data. In all cases, states are sampled from distributions that contain the LQ trajectories, but by using CC-LfD the harmful aspects of these demonstrations are successfully discarded while the potentially informative signal is maintained.

Applications to Transfer Learning: The keyframe constraint optimization performed by CC-LfD can also be used to effectively generalize skills across contexts where interpretations of constraints differ. As an empirical proof-of-concept, CC-LfD is able to generalize the pouring task to a new cup requiring a grasp that is orthogonal to the grasp encountered during training (and thus, requiring a very different trajectory through configuration space). The application of the upright constraint to the new cup universally results in skill failure, as at least one of the boundary keyframe distributions is unable to produce any viable samples that conform to the

Placement Results			
Type	**# Poor Demos**	**# Repairs**	**Success %**
Baseline	3	0	10
7 Repairs	3	7	50
14 Repairs	3	14	40
21 Repairs	3	21	70
28 Repairs	3	28	70
Constrained	**3**	**1**	**90**

Table 2.3: Placement Task Evaluation Results. Percentage of successful task executions out of 10 attempts. A single constrained demonstration is adequate to repair the skill given the baseline of three poor quality demonstrations.

required constraints.

By introducing a single demonstration performed under the new conditions, CC-LfD is able to learn a model that can perform the skill correctly, despite the fact that the rest of its training data was performed such that it never exhibited the correct upright behavior. This example suggests skill transfer as a promising application of future work extending CC-LfD, as will be shown in the upcoming chapters.

2.3.4 Contributions:

Given the above results and discussion, the novel contributions of the CC-LfD algorithm are as follows:

- CC-LfD enables few-shot skill learning by utilizing constraint information specific to a skill that results in a better keyframe model representation of the desired skill to be learned by the robot system.
- One-shot repair by introducing constraint information from annotated trajectories into existing unannotated training data to overcome potentially poor demonstrations that equate to a poor-performing model.
- A preliminary result showing how CC-LfD facilitates skill augmentation by updating constraint parameters to produce a new constrained skill model.

2.4 Maintaining Constraint-Compliance Introduces New Challenges

While the CC-LfD algorithm presents a number of benefits in terms of user interaction with an LfD system and the robustness increases of the model itself, it introduces two challenges that warrant further investigation.

(1) *Challenge I*: During demonstration, constraint annotation is difficult for novice demonstrators, and there is no obvious insight into whether or not the system learned the skill appropriately nor an easy way to update a learned skill without demonstration.

(2) *Challenge II*: It is desirable to push model representation towards a constraint transition keyframe-only representation, but this introduces the need for appropriate constrained motion planning to guarantee constraint compliance during execution.

2.4.1 Challenge I: Interface Design and Model Insight

CC-LfD provides a mechanism to integrate constraints according to the user's discretion, and it builds a keyframe model shaped by those constraints and corresponding trajectory data. However, there are some limitations introduced by the CC-LfD related to interface design and model insight, as outlined below.

Interface Design: Constraint annotation is challenging to perform during demonstration with existing interfaces. With kinesthetic demonstration, a user often has to use both hands in order to successfully guide a robot through a motion. This makes constraint annotation challenging to perform as the most common interfaces utilize triggers either on the robot arm itself or through a toggling interface on a tablet, to indicate when constraints apply. This naturally limits the number of accessible constraints available to the user during the demonstration itself, and when demonstrating kinesthetically, the user often has to pause. Narration through a natural language interface is one possible alternative approach, although it still makes it difficult to annotate multiple constraints concurrently during the demonstration.

Model Awareness: Another limitation with the standard CC-LfD algorithm, is that there is no obvious indication as to how successful the robot agent will be at executing the skill. The only insight into the model that the user has is previewing the generated motion plan either in simulation or on the actual robot platform. In isolation, this preview might not provide enough information for the user to validate whether or not the skill was learned as intended. A simulated environment might not be physically situated in the space the demonstrations occurred. The actual execution of the skill puts the cart before the horse in that a poorly learned skill results in a potentially dangerous behavior preview. Relatedly, a user might want to verify whether or not the constraints they annotated are appropriate for the model or are in the location they desire.

Augmented Reality for CC-LfD: Chapter 3 outlines a holistic augmented reality approach to overcoming these limitations. It will describe how keyframe previews of a learned skill combined with visualizations of assigned constraints enable users to demonstrate without the need for real-time annotation. This allows for the post-hoc shaping of learned skills through constraint editing. In a similar vein, this same augmented reality system can be used to visualize robot configurations in lieu of kinesthetic demonstration. By utilizing an instrumented tong to provide pose targets for a real-time feasible pose-optimization, a user is free to focus on the task at hand. To assist in the demonstration of constraints, this pose optimization mechanism can also optimize for task-space constraints.

2.4.2 Challenge II: Keyframe Sparsity and Constraint-Compliant Motion Plans

The second major challenge relates to shifting model representation towards constraint compliance (i.e. a boundary keyframe dominant representation) and less on a highly granular keyframe model (i.e. reliance on intermediate keyframes). The more and more intermediate keyframes are culled, the model representation shifts towards a representation anchored by the constraint transition keyframes. The benefit of this sparser representation is that it provides a means to push execution toward automated motion planning. However, this requires that the planning between one constraint transition keyframe to the next constraint transition keyframe requires constraint

compliance. This introduces the need for constrained motion planning methods in order to guarantee constraint-compliant robot execution of learned skills.

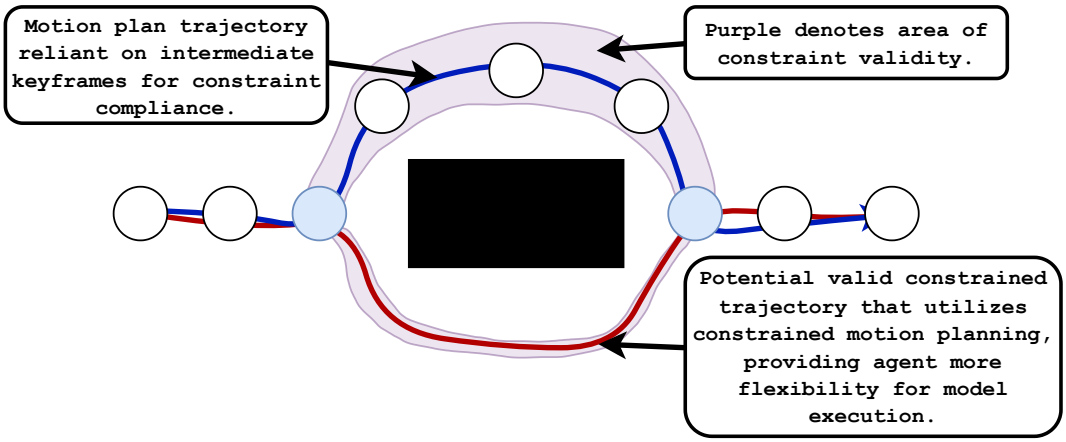


Figure 2.13: CC-LfD relies on intermediate keyframes to produce a constraint-compliant trajectory (blue). This limits the potential for alternative constraint-compliant trajectories that diverge away from intermediate trajectory distributions (red), but requires the use of constrained motion planning algorithms.

Keyframe Sparsity The CC-LfD algorithm produces a sequence of constraint-compliant waypoints but relies on traditional sampling-based motion planning algorithms to move from point to point. Such algorithms do not necessarily produce constraint-compliant motion plans. Given that the initial implementation and evaluation of the CC-LfD algorithm utilized permissive tolerances for the concept constraints used in each evaluation task, ensuring that each intermediate trajectory was compliant with the given segments assigned constraints proved feasible. However, should the model need to accommodate changes in the environment that occlude intermediate keyframes or the user simply want to push execution onto algorithmic motion planning, there is a need for constrained motion planning.

One of the essential procedures in the CC-LfD algorithm is the keyframe culling process. Intermediate keyframes are removed from the graph so that during sampling for skill execution, adjacent waypoints do not result in undesirable behavior like backtracking. Depending on the CC-LfD parameters from grouping data and generating keyframes, this process generally produces a large number of intermediate keyframes relative to the number of boundary keyframes. Since these

intermediate keyframes must adhere to the constraints assigned to a prior boundary keyframe, they ultimately produce a sequence of constraint-compliant waypoints that are granular enough such that traditional motion planners are able to adhere to the set of constraints. This is possible as the divergence from one point to another is minimal, generally enabling the behavior resulting from executing skill trajectory to adhere to constraints throughout the motion of the robot agent. However, this is not strictly guaranteed as the motion from one keyframe to another still relies on motion planning algorithms that are unaware of constraints.

One benefit of automated motion planning is that it drives behavior execution away from reliance on explicit programming or, in the case of CC-LfD, away from relying on intermediate keyframe waypoints. Such intermediate keyframes act as a form of model over-fitting and might only represent a single instance of a skill as demonstrated by the user. In certain cases, the most pertinent keyframes of the skill might be the constraint transition keyframes. These keyframes encode important change-point locations about the behavior of the robot. A keyframe model that uses only constraint transition keyframes results in a representation that depends more on constraints and their change locations and far less on the specific instance of the demonstration trajectories. As will be shown in Chapter 4, this representation defines a challenging constrained motion planning problem in which constraints change throughout the required plan. Knowing when and where these changes occur is an ill-formed problem that the CC-LfD sparse keyframe representation provides the means to solve.

Chapter 3

Augmented Reality Interfaces for Learning from Demonstration

This chapter describes two interactive systems for robot Learning from Demonstration that utilize augmented reality (AR) as a means to facilitate novel modes of demonstration, to view learned robot behavior, and to update a learned model's assigned constraints. The first is called *Augmented Relation for Constrained Learning from Demonstration, (ARC-LfD)*[1] . The second system is called *Augmented Reality-based Pose Optimization for Constrained Learning from Demonstration (ARPOC-LfD).*

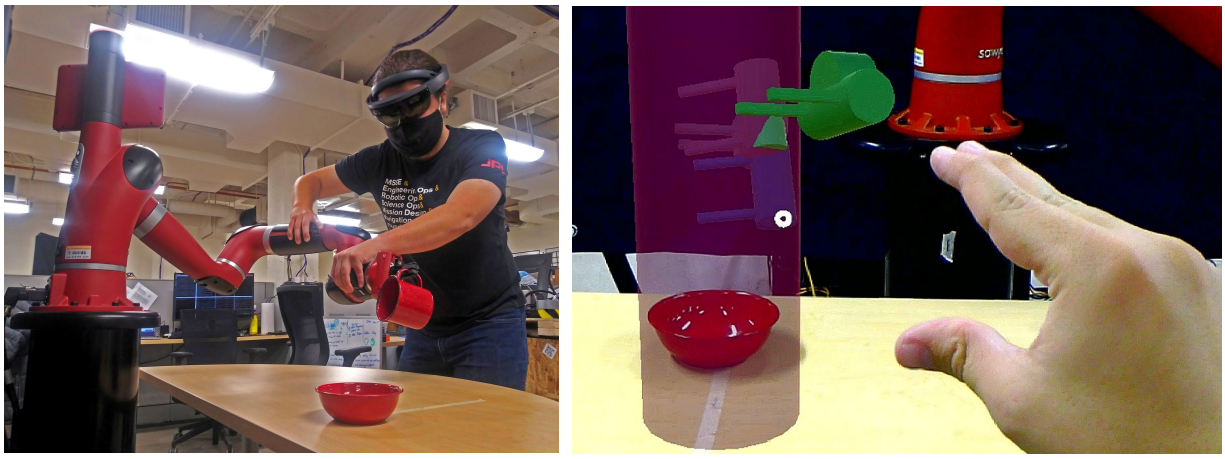


Figure 3.1: Augmented Reality combined with constrained Learning from Demonstration creates novel interfaces that enable the teaching, verification, editing, and updating of robot skills using in-situ visualizations and interaction. The above picture shows a user kinesthetically demonstrating (left) and editing skills afterward (right).

[1] Note that the information presented in this chapter draws upon, paraphrases, and uses text verbatim from Luebbers et al. [90]

The motivation driving the development of these two systems is as such:

(1) Most LfD systems provide little insight into the quality and capability of the learned skill, including the previously discussed CC-LfD model (see Chapter 2).

(2) Traditional modes of demonstration (kinesthetic and teleoperation) have specific shortcomings that make one preferable over the other, and these modes do not readily support constraint communication in their standard form.

(3) These systems enable user-driven skill correction and enable constraint-compliance assistance during demonstration.

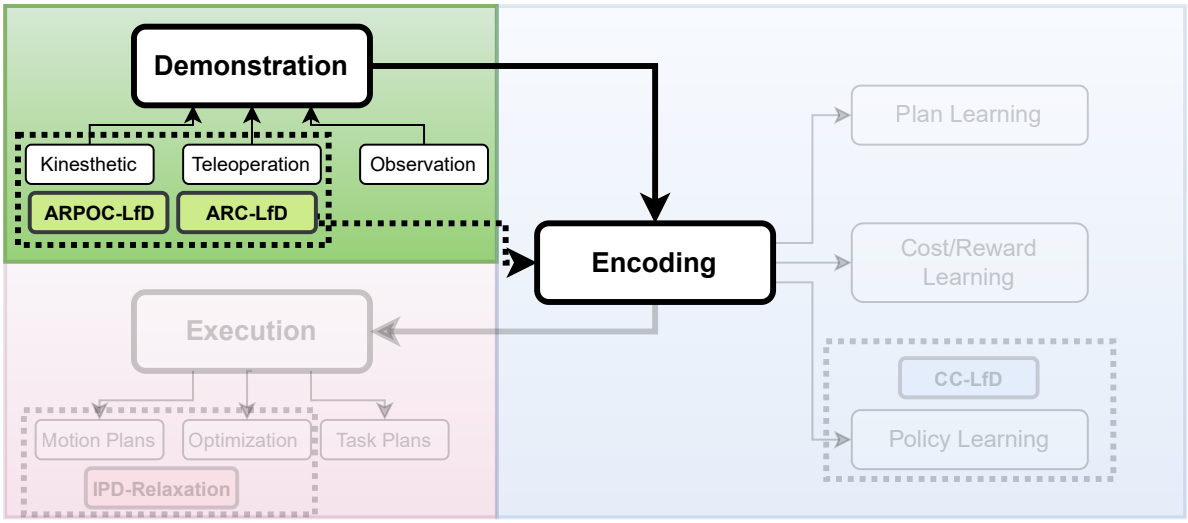


Figure 3.2: Referring to Figure 3.2, ARC-LFD and ARPOC-LfD support kinesthetic and teleoperation-based modes of demonstration while providing visualizations of constraints and the resulting learned model.

These two systems contribute predominantly to the demonstration phase of the LfD pipeline (see Figure 3.2). However, ARC-LfD does directly interact with the CC-LfD model, thereby providing a direct association with the encoding portion of the pipeline. ARC-LfD provides in situ visualizations representing constraints, end-effector holographics, and feedback regarding skill capability. This includes visualizations of end-effector holograms that represent candidate samples from the model's keyframes. Additionally, on each keyframe, there are visual indicators specific

to each assigned constraint that indicate if the model is capable of adhering to the constraints as well as a means to update constraint parameters. This results in an ability to adapt skills post-hoc without repeating the entire learning process.

ARPOC-LfD is a teleoperation system that employs real-time feasible pose generation for teleoperation with the added ability to assist in constraint compliance during demonstration. The system supports the use of an instrumented tong that provides an easy mechanism for users to supply pose-targets during demonstration. Similarly to ARC-LfD, the system utilizes AR to provide a virtual holographic manipulator arm to provide in-situ concurrent visualizations of the online pose optimization output to facilitate better situational awareness of robot capability during demonstration.

3.1 Preliminaries

A useful framing for the utility of these two AR systems lies in what is called the Human-Action Cycle. Adapted for the human-robot interaction setting in [138], this cycle describes important components from the user perspective when it comes to human-robot interaction. Figure 3.3 provides a new variant of the cycle using a Learning from Demonstration perspective (gray box) and discusses how ARC-LfD and ARPOC-LfD integrate into this cycle (blue and pink boxes). In this adapted cycle, a human user is responsible for three major components in their interaction with a robot learning system. A user must first consider the goals of the LfD interaction, perhaps wishing for the system to learn the stylistic nature of the demonstration data, or they might be more concerned with adherence to task constraints. The next consideration is the actual process of demonstration during which they must ideally provide adequate quality demonstrations. Finally, a user must evaluate whether or not the robot system has correctly learned the skill as communicated through demonstration. As mentioned in [138], each of these components of the cycle could result in a breakdown in successful human-robot interaction. In the case of Learning from Demonstration, such breakdowns could result in poorly performing learned models, unsafe robot execution, and less trust in the system.

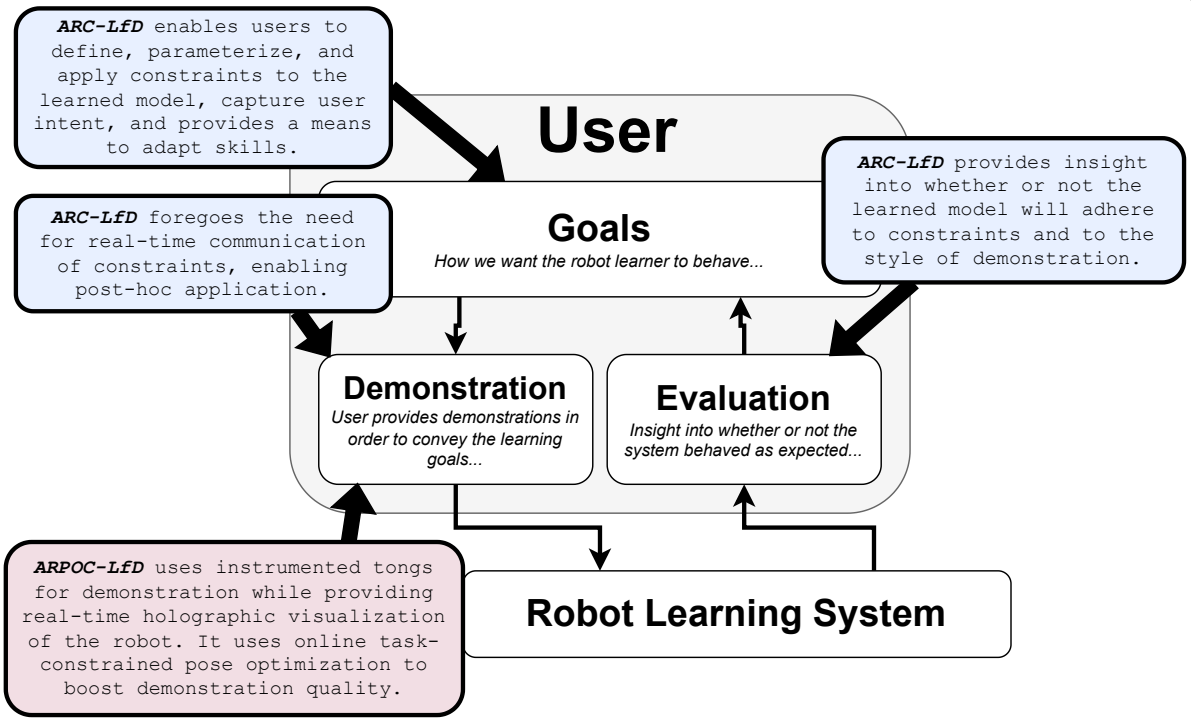


Figure 3.3: The Human-Action Cycle adapted for LfD from Szafir et al. [138], with contributions of ARC-LfD in blue and ARPOC-LfD in light red. ARC-LfD provides the means to capture user goals through constraint editing and application, enables users to evaluate learned models, and enables post-hoc application of constraints onto keyframes. ARPOC-LfD provides users real-time feedback during demonstration and boosts the quality of demonstrations in constrained settings through the aid of online-pose constrained pose optimization.

3.1.1 Utility of Augmented Reality for Learning from Demonstration

The above characterization extending [138] adds to the growing body of literature that bears out the utility of AR interfaces for robotics [52, 138, 14], enabling new methods of enhancing robotic control [154, 151, 148, 24], collaboration in human-robot teaming [121], allowing safe movement in shared spaces [147, 121], and communication of robot knowledge [79, 81, 42]. Augmented Reality also provides benefits as an interface mechanism for a variety of Learning from Demonstration methods as evidenced through a growing body of research [132, 47, 46, 112].

When using LfD methods for robot instruction, safe and successful deployment necessitates that a learned skill meets the requirements of the human teacher (i.e. *goals* in the Human-Action Cycle). This deployment also necessitates the system has learned a proper model of the skill after

the demonstration (i.e. *evaluation* in the Human-Action Cycle). While verification can be done in simulation, this requires a high-fidelity model of the environment in order for the visualization of the learned skill to be shown in the proper context (and obtaining such a model may be a technical endeavor) [150]. However, small changes in the robot's environment or the desired skill may require an entirely new set of demonstrations to fix it. This requirement for rigidity of environment and task can make long-term deployment and maintenance of the skill difficult in practice.

One approach to handling this rigidity is the creation of end-to-end policy learning systems that aim to model skills more generally [120, 10]. However, such systems may demand a prohibitive number of demonstrations or require unavailable simulation environments to capture user intent, and aren't designed to accommodate user selection of task constraints. The CC-LfD algorithm presented in Chapter 2 shows that without constraints, the naive Keyframe LfD model requires numerous demonstrations to reshape a skill. As such, ARC-LfD's central focus is on the application, editing, and integration of constraints in order to reduce the dependency on demonstration (i.e. the *demonstration* component of the Human-Action Cycle).

ARC-LfD safely demonstrates to users what skill has been learned and how executing that skill will cause the robot to move through the environment through in-situ holographic visualizations. The AR interface also facilitates the visualization and editing of constraints, enabling users to see how these constraints interact with objects or points of interest in the environment. Furthermore, constraint editing through AR allows the entire training process to take place in situ without requiring context-switching between the real environment and a 2D display [61].

The ARC-LfD systems enable users to examine a sample trajectory from a learned skill visualized in AR through an overlay in the workspace environment. Such skill visualization is intended to improve safety as the operator can "preview" robot behavior without the need for actual skill execution [82]. Prior work has established this potential through user studies: Walker et al. [147] conducted a user study that found that showing flying robot paths in AR made users more efficient and comfortable when sharing an environment with these robots. Similarly, Rosen et al. [121] found that AR visualization of possible robotic arm trajectories improved participants'

accuracy and quickness in identifying collisions with objects in the environment. These studies substantiate the notion that AR visualizations of robot trajectories may improve user understanding with respect to the path a robot will take and how that trajectory will interact with the environment.

In addition to visualizing the robot's possible future movement, ARC-LfD supplies visual cues that describe the robot's ability to adhere to user-supplied behavioral constraints on a learned skill. This is akin to helping users understand the internal state of the robot, another functionality that has been explored within the space of AR for human-robot interaction. Through AR, information such as the robot's battery life [81] or sensor readings [79] can be communicated to users through a heads-up display. This is particularly useful when performing complex tasks such as controlling a robot as it prevents disruptive context-switching when averting attention away from the environment towards a 2D display [61]. Using AR to visualize a robot's knowledge in the form of a learned skill or action can also provide a realistic demonstration of this knowledge without requiring extensive modeling of the environment to use in simulation [42].

The final type of interaction supported by AR in ARC-LfD is the ability to create and manipulate constraints on a learned skill. Visualizing constraints in the physical environment allows users to see the exact effect of applying these constraints [133]. Yamamoto et al. [154] illustrated that applying virtual constraints was an effective tool for robot-assisted surgery, allowing surgeons to specify thresholds that the robot should not cross. In our case, the constraints are both shown and edited in the environment in which the skill will be executed, allowing users to move constraints around physical objects to ensure the skill can be performed safely. Fang et al. introduce an approach that enables users to create a sequence of control points interactively on a parameterized curve model [46, 47]. Users then define the orientation of the end-effector associated with each control point. These points are used to generate a ruled surface representing the path to be planned. While this is similar to the ARC-LfD approach, ARC-LfD does not rely on the use of control points. The underlying CC-LfD model generates the trajectories and candidate keyframe points. ARC-LfD serves to shape the skill through a library of available constraints useful for the specific task, not by providing specific keyframe points.

3.1.2 Revisiting Interaction Modes for LfD

The predominant focus of LfD research to date has been on the initial learning process itself, often relying on the standard forms of demonstration introduced in Section 2.1.2. Kinesthetic demonstration remains one of the most widely used methods of human-robot interaction for Learning from Demonstration systems [99, 126, 26, 110, 4, 3, 7, 11]. A kinesthetic demonstration is exacting in that when done correctly it provides a very precise and accurate instantiation of the skill to be learned [3, 4, 2]. However, kinesthetic demonstration for naive users can be awkward without minimal training [110] which might result in sub-optimal demonstration [107, 51]. Furthermore, kinesthetic demonstration may not always be feasible for a given environment or robotic agent. For example, certain domains might be dangerous to operate within for a human user. Similarly, the robotic agent itself might be dangerous or impossible to physically manipulate.

In contrast to kinesthetic demonstration, teleoperation provides an alternative approach that forgoes the use of physical manipulation of the robotic agent, instead opting for a controlling device [13, 10, 120, 102, 129]. The development of teleoperation was predominantly pragmatic, where the control of the robot agent could not be left purely to automated motion planning methods if at all [129, 105, 62]. Given the need for tight user control, teleoperation has been used extensively in robotics and mechatronics, from surgical robotics [92, 34], mobile robotics [102], space robotics [123, 128], to collaborative/assistive robotics [9, 97]. The caveat with teleoperation is that teleoperative devices do not always provide an intuitive means to control the robotic agent. This is especially true for complicated high-degree-of-freedom robot agents. A difficult-to-use controller will result in poor agent behavior as the user cannot easily overcome the poor mapping of teleoperation inputs to control outputs. In the lens of Learning from Demonstration, this difficulty may result in poor demonstration [110, 120] or at the very least be cognitively burdensome to the operator during demonstration [56].

One approach to making teleoperation more effective is to utilize a proxy device that creates a more intuitive mapping between the controller and the agent. For example, Fang et al. uses

a data glove acts as a proxy for the end-effector, providing target points for the robot as a set of demonstrations [45]. This glove is intuitive from users' perceptive as they are using their own hand as the means to demonstrate, rather than physically manipulating the robotic arm as in kinesthetic demonstration or attempting to learn a difficult controller. Praveena et al. introduces an instrumented tong that serves as a proxy for the end effector [110]. Users only need to consider the close mapping between the behavior of the tong and the behavior of the manipulator's end-effector, which usually uses a dual-pronged gripper. Generally, by collapsing the input space and output space onto a common user-intuitive data space (i.e. task space), a demonstration is generally easy and more exact [110].

The challenge with these approaches is that while the device-to-agent mapping has been simplified from the usability perspective, control over the degrees of freedom of the robot is no longer supplied by the user. In this case, reliance on kinematics equations to produce agent configurations becomes paramount. Essentially the data glove and instrumented tong approach supplies pose targets rather than agent configurations. Many LfD methods rely on agent configuration/control trajectories as input (see Section 2.1.3). The motive is to keep model representation and the execution space (e.g. robot states or controls) as close as possible, if not equivalent. This reveals one of the major difficulties in using pose/task-space trajectories: there is a need to generate feasible agent configurations. Given the redundancy of high degree-of-freedom manipulator arms, the generated configuration from one pose to the next might produce an infeasible robot trajectory. This might result in a learned model that performs very poorly or may not achieve successful robot execution of the learned skill at all.

3.1.3 Task-space to Configuration Space Optimization

For some settings, demonstrations of pose/task-space data rather than agent configurations are preferable. During kinesthetic demonstration, a user might only care to focus on the end-effector rather than supporting the entirety of the arm. In the case of the CC-LfD algorithm, the less a user needs to focus on maintaining joint positions, the more they can focus on adhering to

task-space constraints and assigning constraints onto the skill. Similarly, teleoperation methods that produce pose/task-space data enable users to focus on the specifics of the skill and not on controlling multiple degrees of freedom that may play an insignificant role in skill success. The major challenge with utilizing this data when using trajectory-based learning methods, such as CC-LfD, is the generation of robot configurations. As mentioned above, a trajectory is only valid if the robot can physically move from configuration to configuration without any joint discontinuities [115, 116]. Given the kinematic redundancy of high degree-of-freedom agents, the inverse kinematics solution to these task-space trajectories often produces a configuration space trajectory that is kinematically/physically infeasible for the robot to execute.

Methods do exist for generating a feasible configuration trajectory from a task-space trajectory. [145] uses null-space impedance control to generate controls that produce configurations that track a task-space trajectory. However, redundancy resolution is dependent on problem domain-specific matrices during null-space projection. [155] uses ruled surface optimization for pose trajectory under kinematics constraints in order to generate a configuration space trajectory. Full trajectory optimization methods are generally not well-suited for real-time demonstration should a user desire real-time feedback. As such, optimization is also useful on a per-pose basis, as shown in [115, 116, 118] in which a non-linear multi-objective optimization program that utilizes a neural network-based approximation for fast self-collision look-ups enables online feasible pose-optimization.

The ARPOC-LfD method opts to use the online pose-optimization framework CollisionIK [118] to generate robot configurations in real time that respect collision objects and maintain feasibility with prior results from the optimization engine. As the multi-objective function utilizes gradient-estimation methods via a Gaussian-like wrapping function (called Groove Loss) per term, it enables the integration of task-space constraint terms. This approach allows users to generate pose targets during a demonstration against which the optimization engine generates pareto-optimal configurations that satisfy the pose target, avoid collisions, avoid joint limit constraints, and adhere to any number of constraint terms included in the objective. With the addition of a real-time

augmented reality interface employed by ARPOC-LfD, these configuration space solutions provide visualizations of the robot holographic that inform a user how well the robot is able to target the task-space instructions provided by that user. Furthermore, with the inclusion of task-space constraints, the optimization engine produces configurations that adhere more closely to task-specific constraints. This has the potential to increase the quality of demonstrations that ultimately produce a more useful model.

3.1.4 Feedback to Foster Self-Correction

The benefits of feedback as a mechanism for self-correction are well supported in the research literature. In the field of psychology, there is evidence that self-correction is an essential feature of the learning process for reading in young children [48]. Despite the common idea of error avoidance as an end goal of learning, Metcalfe et al. showed that errorful learning followed by corrective feedback is beneficial to learning both for students and teachers alike [95]. Freedberg et al. describe a study that indicates both positive feedback (indication performance is correct) and negative feedback (indication performance is incorrect) both help the learning process, but negative feedback might be more useful in this regard [49]. Within the domain of human-robot interaction, visualization as a feedback mechanism has been shown to benefit collaboration in literature as well [138, 147, 148, 24, 119]. To this end, ARPOC-LfD deploys two feedback mechanisms to facilitate self-corrective behavior in human demonstrators. As mentioned above, the primary mechanism of feedback is the holographic visuals of a robot agent tracking pose targets provided by an instrumented tong device. The second feedback mechanism provides an indication of the divergence between the virtual holographic end-effector and the physical target of the tong. The bigger the tracking error, the larger the indication that the user is providing an unobtainable pose target.

3.2 System I: ARC-LfD

This section describes the first AR system, *Augmented Reality for Constrained Learning from Demonstration or ARC-LfD* . ARC-LfD is proposed as a step toward producing practical, real-

world-ready LfD systems that allow non-roboticists to conduct training and evaluation of robotic systems. The use of AR for in-situ visualizations relaxes the requirement of a model of the environment to use in simulation for verification of learned skills. Through visualizing a sample trajectory directly in the environment, users can preview the robot's skill execution contextualized by the actual environment itself. The control flow of ARC-LfD provides an improvement over CC-LfD, allowing users to separate demonstration from constraint application. Rather than requiring users to specify constraints during live demonstrations, users are able to visualize and edit constraints at the verification step. This allows focusing on ensuring constraints are both appropriately specified and applied to the correct keyframes before application.

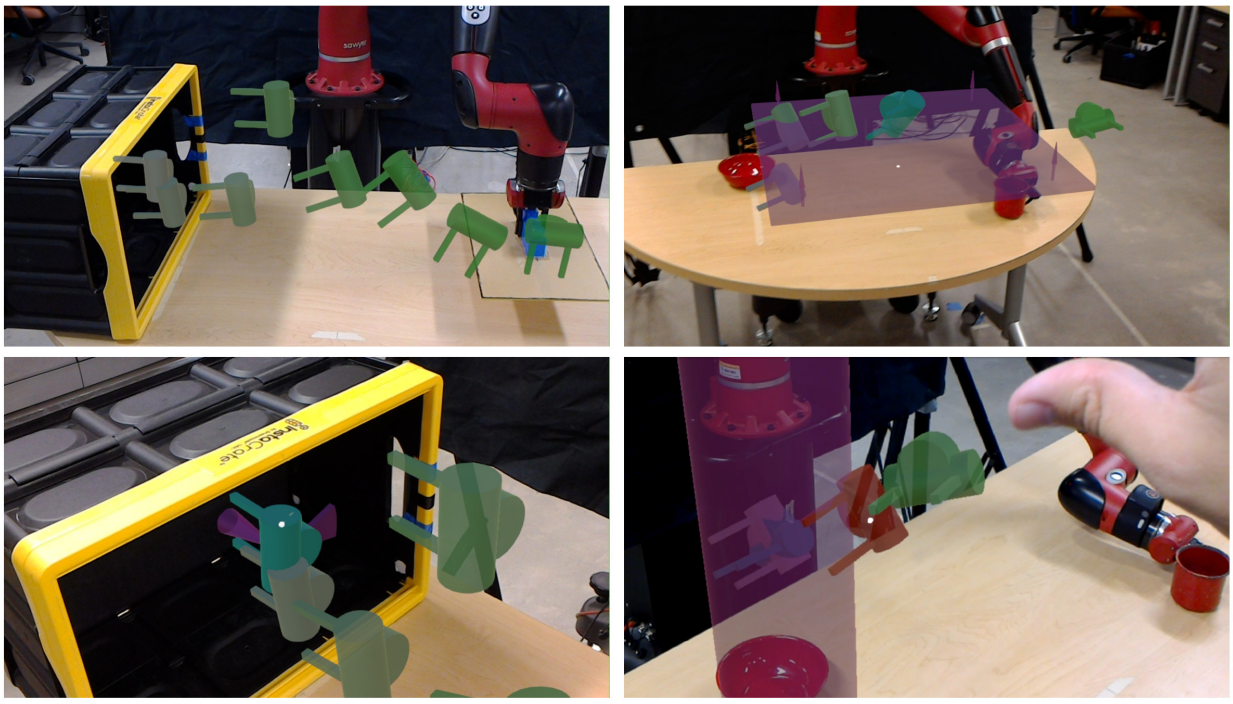


Figure 3.4: ARC-LfD allows the user to visualize trajectories as a series of keyframes (top left). Selecting a keyframe will show holograms representing any constraints active at that keyframe, such as the height constraint (top right) indicating the end-effector must stay above the plane, the orientation constraint (bottom left) overlaid on the selected end-effector to show its proper rotation, and the over-under constraint (bottom right) indicating the end-effector must stay within the cylinder. Note that in the bottom right image, one keyframe has the over-under constraint applied. However, this sample is not located inside the cylinder and is in violation of the constraint. This is indicated by coloring the hologram red to alert the user.

Finally, the proposed constraint editing interface relaxes the static environment assumption

often levied for successful LfD skill deployment. ARC-LfD enables direct skill repair and editing, creating constraints contextualized in the environment and applying them to keyframes of an existing skill. Thus, ARC-LfD fills a critical technical gap in LfD systems, enabling long-term skill assessment and validation as the environment or task requirements change over time, especially tasks that incorporate the concepts constraints of the CC-LfD system and corresponding algorithm.

3.2.1 System Design

ARC-LfD consists of two components communicating via the Robot Operating System (ROS): a Concept Constrained Learning from Demonstration subsystem (CC-LfD), which serves as a back-end for skill learning, and an AR subsystem for visualization and user interactions with a learned skill (see Figure 3.5). The first subsystem is a container for the CC-LfD algorithm to build learned models based on trajectory demonstration data and assigned constraints. This system rebuilds a model based on users indications. It sends a representation of the keyframes to the AR subsystem by sampling a candidate end-effector pose from each keyframe. If directed by the user, this subsystem generates a sequential motion plan (see Chapter 4) for the robot to execute.

The second subsystem of ARC-LfD (see Fig. 3.5) is an AR interface deployed on a HoloLens, a mixed-reality headset developed by Microsoft. A headset was chosen over alternative tablet-based pass-through AR solutions due to its hands-free nature, freeing users' hands for interaction with the robot, and its ability to show different imagery to different eyes, enabling superior depth perception [93]. Users wearing the HoloLens are able to see holographic visualizations of relevant keyframes and constraints projected onto the robot's workspace. User interaction is achieved through performing pinching gestures known as *air taps* on these visualizations and on menu buttons pinned above the robot (see Fig. 3.1).

3.2.2 Interaction Flow

ARC-LfD introduces an advancement over CC-LfD by enabling post-hoc application of constraints as opposed to requiring constraint application during demonstration. This new approach

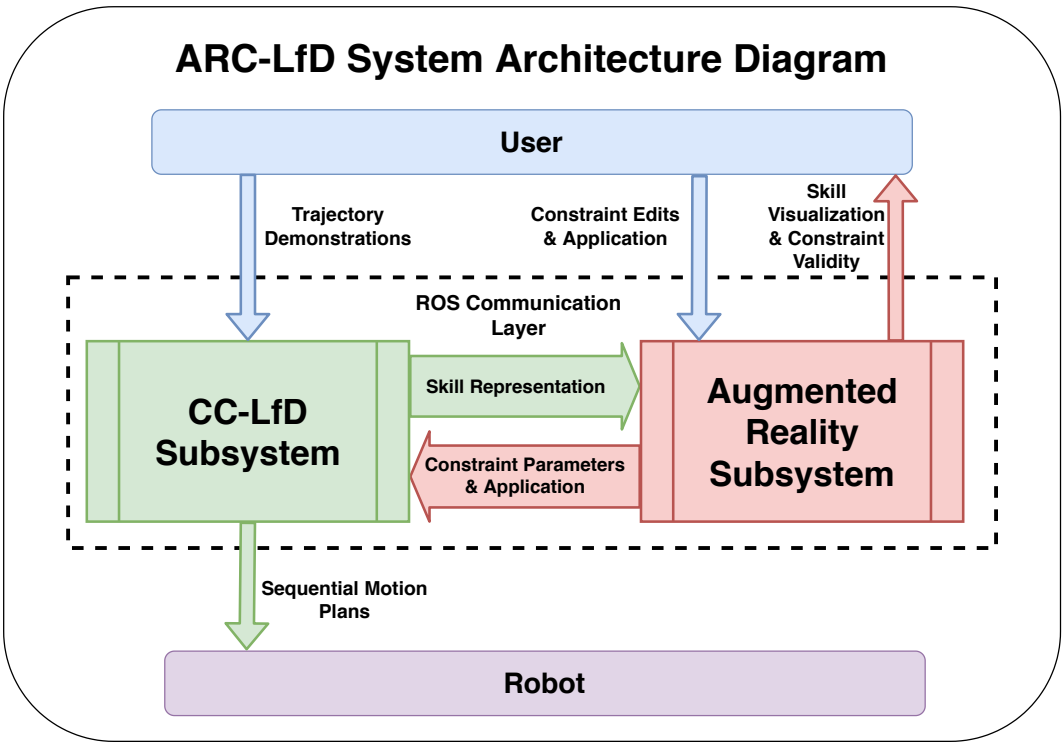


Figure 3.5: A diagram of the ARC-LfD system architecture. The user (blue) supplies the initial demonstrations to the CC-LfD subsystem (green). During the editing phase, the user also supplies constraint edits and their keyframe application to the AR subsystem (red). In return, the AR subsystem supplies skill and keyframe constraint validity visualizations to the user. Through a Robot Operating System (ROS) communication layer, the CC-LfD and AR subsystems exchange skill representation, constraint parameterization, and constraint application information. Finally, the CC-LfD subsystem provides sequential motion plans for the robot (purple) to execute.

facilitates an iterative update process that alters keyframe constraints and the corresponding distributions, providing the basis for ARC-LfD to achieve skill adaptation. ARC-LfD first generates an initial keyframe model of the skill (Fig. 3.6, Step 1), which is visualized as an instantiation of the keyframe waypoints that the robot will execute (Fig. 3.4). This visualization includes the validity of each waypoint relative to the keyframe's applied constraints (Fig. 3.6, Step 2). Using the AR interface, the user generates new constraints, or edits existing constraints (Fig. 3.6, Step 3), and assigns them to a chosen keyframe. This initiates a model rebuilding phase where keyframe distributions are relearned using the same rejection sampling and distribution fitting steps as CC-LfD (Fig. 3.6, Step 4). If the user is satisfied with the visualized robot behavior, skill execution

can proceed as carried out by the CC-LfD algorithm (Fig. 3.6, Step 5).

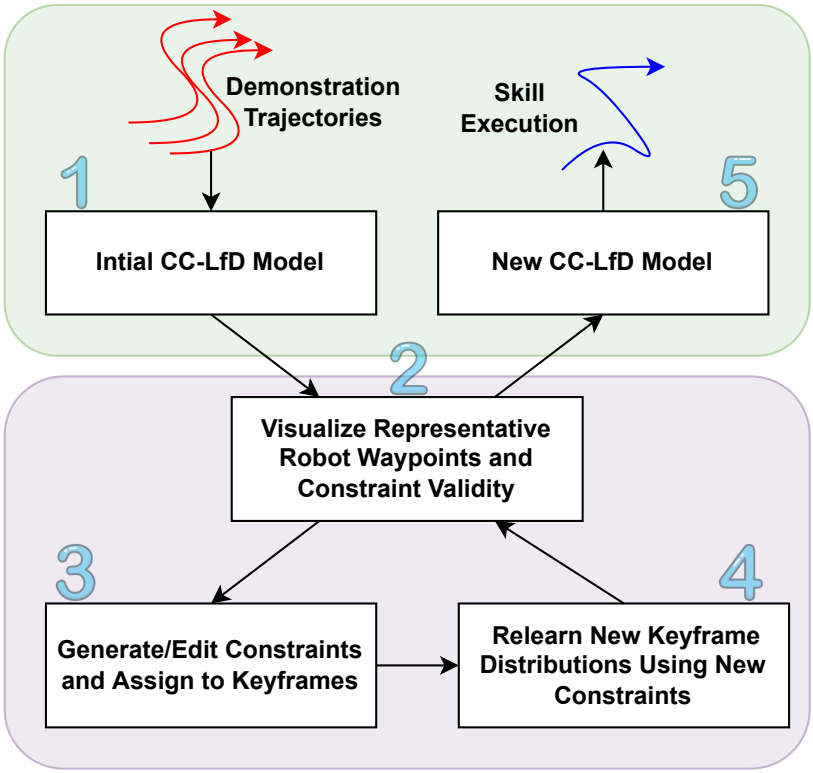


Figure 3.6: Flowchart indicating how ARC-LfD integrates into CC-LfD. Steps 2, 3, and 4 repeats until the user is satisfied. The pink region (bottom) indicates AR-based steps whereas the green region (top) indicates that AR is not strictly required.

3.2.3 Skill & Constraint Representation

For a given skill, each keyframe generated by CC-LfD is sent to the AR interface and visualized as a hologram of the robot's end-effector, whose position and rotation are representative of a randomly sampled valid waypoint within that keyframe. The combination of these keyframe visualizations traces out a trajectory that the robot would follow to execute the skill. To aid the user in evaluating a candidate trajectory at a glance, the end-effector holograms are colored in a gradient from green to gray to indicate the ordering of the keyframes, and any end-effector holograms in violation of an applied constraint are colored bright red (see Fig. 3.4).

Our test implementation incorporates three constraint types, representing a subset of possible

Implemented Constraints for ARC-LfD System Validation			
Constraint Type	**AR Visualization**	**Parameters**	**Example**
Height Above/Below	Plane w/ Arrows	Reference Height, Direction	Fig. 3.4, top-right
Orientation	Orientation Validity Cone and Fan	Orientation, Affordance Angle	Fig. 3.4, bottom-left
Over-Under	Cylinder	Position, Validity Radius	Fig. 3.4, bottom-right

Table 3.1: Description of constraint types used in the system validation testing.

parametric, predicate-based constraint templates for ARC-LfD, selected to provide coverage over a number of common robotic manipulation task setups. These are height constraints (the robot's end-effector must stay above or below a given height), orientation constraints (the robot's end-effector must maintain a given rotation, within a given affordance), and over-under constraints (the robot's end-effector must stay above a given location, within a given radius). Each constraint type has its own associated visualization: a plane with arrows indicating the valid direction for height constraints, a cone and fan overlaid onto an end-effector showing the affordance for each axis for orientation constraints, and a cylinder representing the radius around a target for over-under constraints. When the user selects a keyframe with a constraint applied, that constraint hologram appears, is positioned, rotated, and scaled according to its parameters, and is colored a translucent purple to maximize the visibility of the trajectory and environment. For a summary of these constraints, their AR visualizations, their editable parameters, and references to examples, see Table 3.1.

3.2.4 Constraint Editing & Application

ARC-LfD lets users edit existing constraints and create new ones from a template via the AR interface (see Fig. 3.7). The user accesses the constraint editing interface by selecting a constraint type and slot with the menu buttons above the robot. The user will then have the trajectory visualization cleared from their view and a lone constraint visualization will be rendered. The user can edit the parameters of their chosen constraint type (see Fig. 3.7), seeing the visualization

update in real time, which allows them to match constraints to environmental features (e.g., placing an over-under constraint on top of a target object for a pick-and-place task).

Once a user is satisfied with their new constraint, they press a confirmation button, which synchronizes the representation across the AR and CC-LfD subsystems of ARC-LfD. They are then able to apply that constraint to a keyframe or range of keyframes through the constraint application menu until they have added the constraint to the desired areas of the skill trajectory. Once this process is complete, and the trajectory has been satisfactorily inspected, the user selects the "Send to Robot" button to send the newly applied constraint to the CC-LfD subsystem, which initiates a rebuilding and resampling of the skill. After the CC-LfD subsystem has relearned a set of new keyframe distributions, it sends them back to the AR subsystem. It updates the trajectory visualization to inform the user if the system adequately captured their intent and whether the skill is likely to be executed successfully. This process of trajectory evaluation, constraint editing, and constraint application can be repeated until the user is satisfied.

3.2.5 System Validation

In order to validate the ARC-LfD system, we examine its operation within three test cases representative of potential task scenarios asked of robot manipulators. These case studies exemplify how ARC-LfD allows a user to demonstrate a skill, visualize the learned skill, then adapt the learned skill to two different environment setups (an "initial setup" and "secondary setup") using edited constraints. One of our research team members acted as a user to demonstrate the system's functionality. Eight kinesthetic demonstrations were provided as the basis for each skill using the Rethink Robotics Sawyer platform. Once the ARC-LfD system had generated a skill model learned from these demonstrations, the user was shown a sample trajectory of this skill. The user then edited and applied constraints with consideration given to the specific environment setup. ARC-LfD used the applied constraint to adapt the initial learned skill and sent a representation of the updated skill back to the user for visual inspection. Finally, the skill was executed on the robot.

These case studies demonstrate situations in which ARC-LfD allows a user to assess and edit

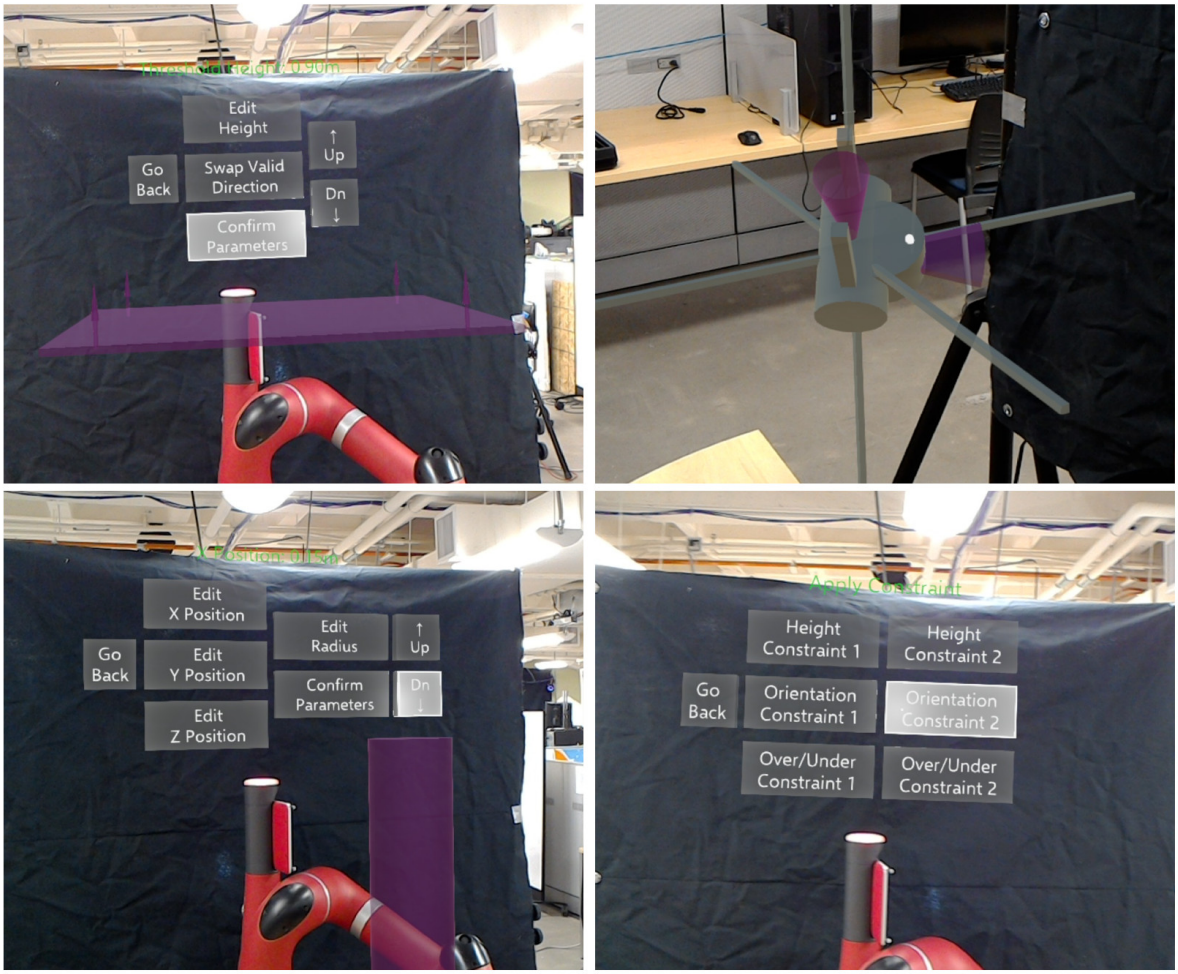


Figure 3.7: Users can customize constraints from templates via the AR interface. After selecting a height (top left), orientation (top right), or over-under (bottom left) constraint, they edit its parameters and see the corresponding visualization update in real-time. Once satisfied, they can apply the newly edited constraint to the model by selecting it from the application menu (bottom right), and by selecting which keyframes the constraint should apply to. After this process, they will send a request to the robot to rebuild and re-visualize the model using any new constraints, and evaluate whether the robot has correctly learned the skill.

a skill in response to changes in the environment or task setup. This illustrates a novel capability over CC-LfD as a user can craft and visualize constraint annotations to ensure successful model adaptation to differing task setups sans additional demonstrations. In these example applications, the entire process (skill visualization, creation and application of a constraint, skill updating within the CC-LfD subsystem, visualization of the updated skill, and approval of execution) took an average of 120 seconds per skill.

3.2.6 Case Study I (Precise Placement): A Placement Task with Orientation Change at Goal Pose

The first case study emulates situations in which the goals of the task are modified after initial demonstrations are given. In this task, the robot's objective was to place a rectangular object into an upright crate, with minimal clearance. If the object was placed using the wrong orientation, a collision with the crate would occur. The user first provided demonstrations with varied orientations of the object. We evaluate the task for two different orientations of the crate, horizontal and vertical, with no additional demonstrations provided between conditions. In both cases, the user applied an orientation constraint to the task's last few keyframes specifying the desired orientation. With the added constraints, the ARC-LfD system enabled the robot to successfully place the object without collision. The setup of this case study is shown in Figure 3.8.

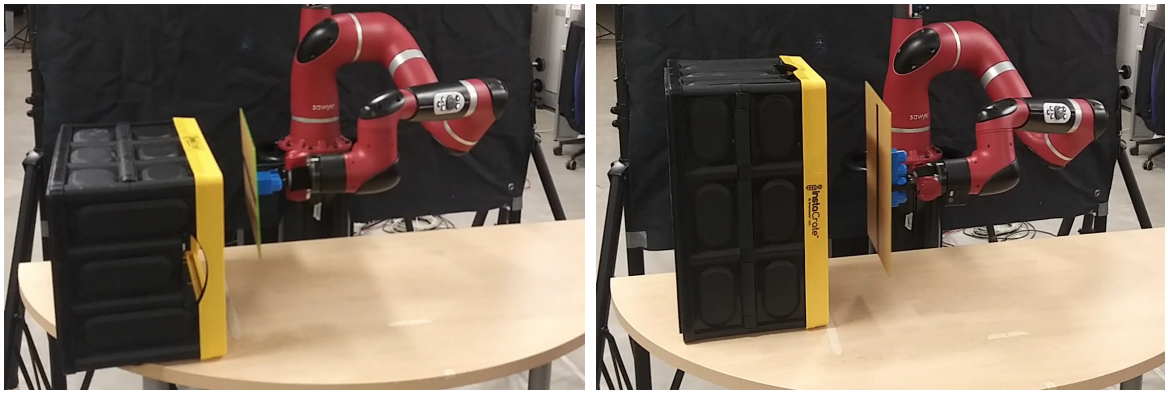


Figure 3.8: For Case Study I, the robot inserts a rectangular object into a similarly-sized rectangular crate. In this case study, the user applies orientation constraints to the final keyframes in the trajectory in order to match the initial setup (left) with a horizontal crate or the secondary setup (right) with a vertical crate.

3.2.7 Case Study II (Changing Environment): Introducing New Obstacles in a Pick-and-Place Task

In the second case study, the robot's task involved moving an object from one side of a table to another. This task is representative of pick-and-place kitting tasks with known start/goal locations but with configurations of obstacles that may change over time. For this case study, the

user provided 8 demonstrations of moving the robot's arm across the table from right to left. The initial environment setup had no obstacles in the way. In the test condition, we placed stacked foam obstacles halfway across the table. By applying a height constraint, the user can edit the skill so that the robot can still complete the task without colliding with the new obstacles and without requiring additional demonstrations. This case study exemplifies how a generic constraint can be used in lieu of a simulated collision obstacle required by motion planning. Images from this case study are given in Figure 3.9.

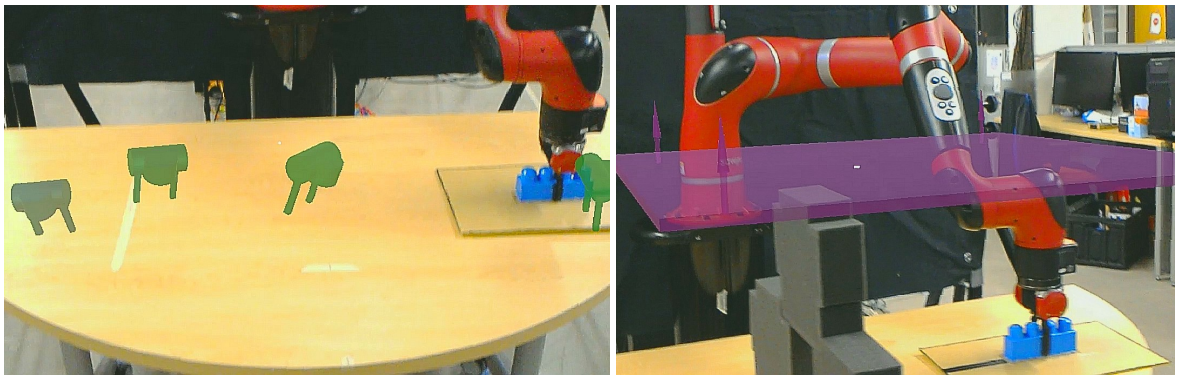


Figure 3.9: In Case Study II, the robot completes a pick-and-place task either with or without an obstacle present. The initial environment (left) has no obstacles on the table, allowing the robot to freely move the object from right to left across the table. The test condition setup (right) introduces an obstacle halfway across the table, requiring the user to apply a height constraint that ensures the robot lifts its payload over the obstacle to complete the task.

3.2.8 Case Study III (Changing Goal): Moving the Receptacle for a Pouring Task

The third and final case study we conducted involved a task in which the robot poured a cup of material into a receptacle. The modification for this case study consisted of moving the receptacle to a different position. Using ARC-LfD's over-under constraint, the user was able to specify where on the table the pouring part of the task should begin. This allowed the robot to execute the cup pouring task successfully with two different end goal positions without any new demonstrations. Figure 3.10 illustrates the environment setup and constraint applications for this case study.

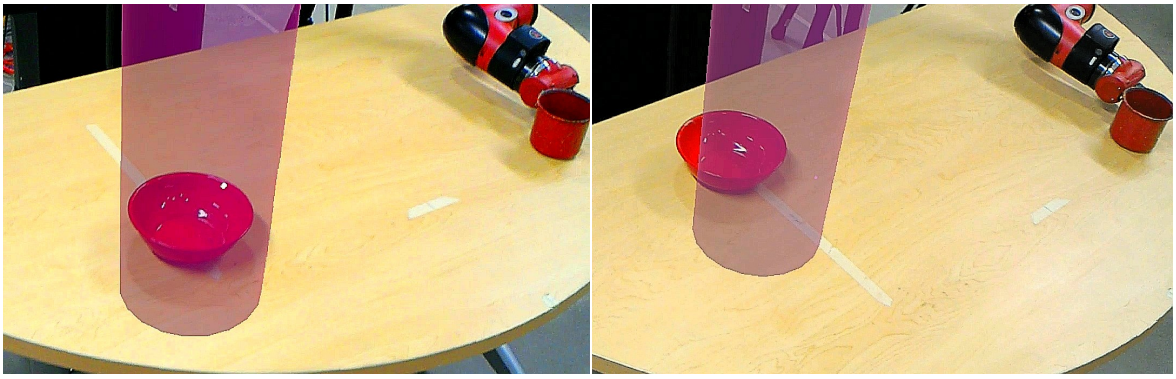


Figure 3.10: Case Study III involves the robot pouring a cup into a bowl positioned at different points on the table. In the initial setup (left), the bowl is placed toward the front of the table, while in the test condition (right), the bowl is placed further back. In both cases, the user applies an over-under constraint to the trajectory representation in order to ensure the pouring motion takes place at the correct position.

3.2.9 Benefits

These three case studies exhibit the functionality of ARC-LfD and its ability to make LfD systems more robust. Case Studies I and III illustrate that ARC-LfD can make a set of demonstrations robust to changes in the task, provided sufficient variance of demonstrations in the set: through the application of constraints to an existing skill, the robot can execute an altered version of a task. Case Study II shows how ARC-LfD can make learned skills robust to changes in the environment by using constraints that alter the skill trajectory to fit a new execution context. Furthermore, the interface of ARC-LfD enables users to conduct these alterations after demonstrations have been given, allowing for any-time editing of a skill. In addition to its functionality for verifying and previewing skills directly in the environment, ARC-LfD introduces a method for maintaining robotic skills even if the particulars of the task and environment shift over time. We posit that ARC-LfD presents a safer-by-construction alternative to general end-to-end policy learning systems, trading generally unneeded levels of model expressivity for system transparency, enabling successful safer skill execution across a broad range of robotics tasks.

The contributions of the ARC-LfD system are as follows:

(1) AR visualizations of learned skills, in-situ robot behavior, and constraints without needing

a model of the entire environment.

(2) An iterative process to verify, repair, and edit existing skills through AR using visualized constraints employed by the underlying LfD algorithm.

(3) Three case studies illustrate how the system enables skill adaptation with no further demonstration.

3.3 System II: ARPOC-LfD

This section outlines the second augmented reality system, *Augmented Reality for Teleoperation-based Constrained Learning from Demonstration (ARPOC-LfD)*, and experiment and evaluation protocols that will ultimately demonstrate the system's utility. The system is a close cousin of the ARC-LfD system in that it provides an in-situ interaction environment for LfD applications using augmented reality. It also incorporates constraints into the demonstration process, but in this scenario, they are used by the system to aid in constraint compliance of generated robot configurations produced by an online optimization process to adhere to user-provided pose targets. However, the primary benefit is that the real-time holographic visuals of the robotic manipulator provide online information for a user to adjust the use of their teleoperative device (in this case an instrumented tong) to accommodate the capability of the robotic system. Whether through the use of constraints to assist in constraint-compliant demonstration or to aid in the self-correction of teleoperation, this system assists users in providing higher quality demonstrations that will ultimately improve the resulting LfD model that uses those demonstrations.

3.3.1 System Design

Much like ARC-LfD, ARPOC-LfD is composed of two subsystems (see Figure 3.12) communicating via the Robot Operating System (ROS): 1) an online feasible pose-optimization subsystem, which generates feasible robot configurations, and 2), an AR subsystem for real-time visualization of a robot holographic as it tracks the instrumented tongs (see Figure 3.11). At its core, the

Figure 3.11: The instrumented tongs used in the ARPOC-LfD system for evaluation. The tongs are tracked using the OptiTrack motion capture system via infrared markers. The two separate groups of markers enable the system to detect a closed gripper based on the center-point distance between each group.

pose optimization subsystem employs a multi-objective non-linear constrained optimization program to provide real-time production of pareto-optimal configurations. This optimization engine is an extension of the Rust-based software, CollisionIK, developed for [118]. It incorporates task constraints as terms in the multiobjective function. In this manner, the system supports the integration of multiple constraints along with the collision avoidance, self-collision avoidance, and joint limit constraints implemented into the CollisionIK software.

The second subsystem of ARPOC-LfD (see Figure 3.12) uses the augmented-reality headset developed by Microsoft. Users wearing the HoloLens are able to see holographic visualizations of a rendered robot agent (e.g. a ReThink Robotics Sawyer). These visualizations align the holographic agent's end-effector with the instrumented tongs end-point. The tongs are tracked via the OptiTrack Motion Capture system, which produces the pose targets supplied to the optimizer given the tong's position and orientation in the environment. The system is able to also provide an indication as to whether or not the tongs are open or closed based on the distance between the track position

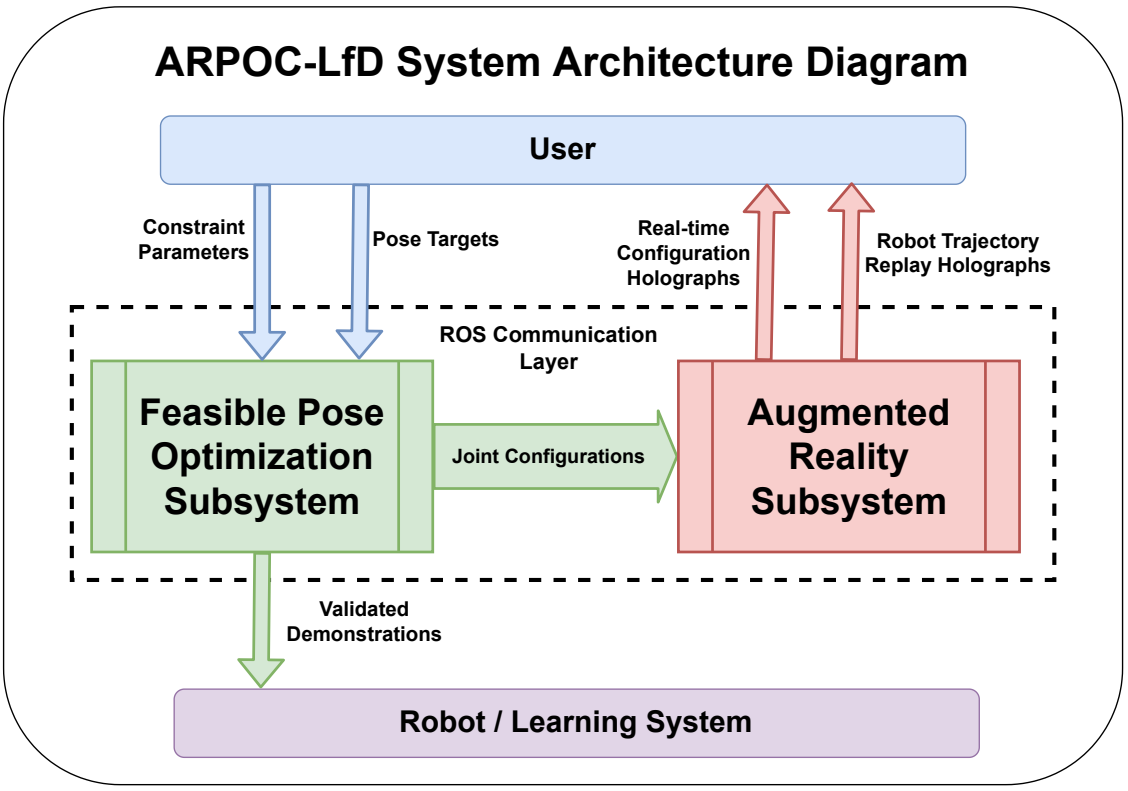


Figure 3.12: The ARPOC-LfD System Architecture diagram. The pink region (bottom) indicates the AR and LfD learning components whereas the blue region (top) indicates how the user interacts with the system.

of each side of the tong. The system provides visual feedback in the form of a changing robot color to indicate the tracking error based on a threshold distance between the position of the tong and the resulting end-effector position based on the forward kinematics of the generated robot configuration.

3.3.2 Interaction Flow

The interaction flow of ARPOC-LfD is outlined in Figure 3.13, the end goal of which is for users to create demonstration trajectories of robot configurations generated from the online pose optimization subsystem. Users first supply pose targets using instrumented tongs tracked by the OptiTrack motion capture system, (Step 1, Figure 3.13). These pose targets are provided to the feasible pose-optimization engine, which produces a pareto-optimal robot configuration that best

targets the provided pose, but also optimizes for static collision avoidance, self-collision avoidance, dynamic obstacle avoidance, and for task space constraints (Step 2, Figure 3.13). If the user is amidst a demonstration, configurations are given to the AR subsystem, which renders holographs of the robot agent into the visual field of the user's Microsoft Hololens (Step 3, Figure 3.13). If a user has reached the end of a potentially desirable demonstration, ARPOC-LfD supports the replay of the entire configuration space trajectory (Step 4, 3.13) on demand.

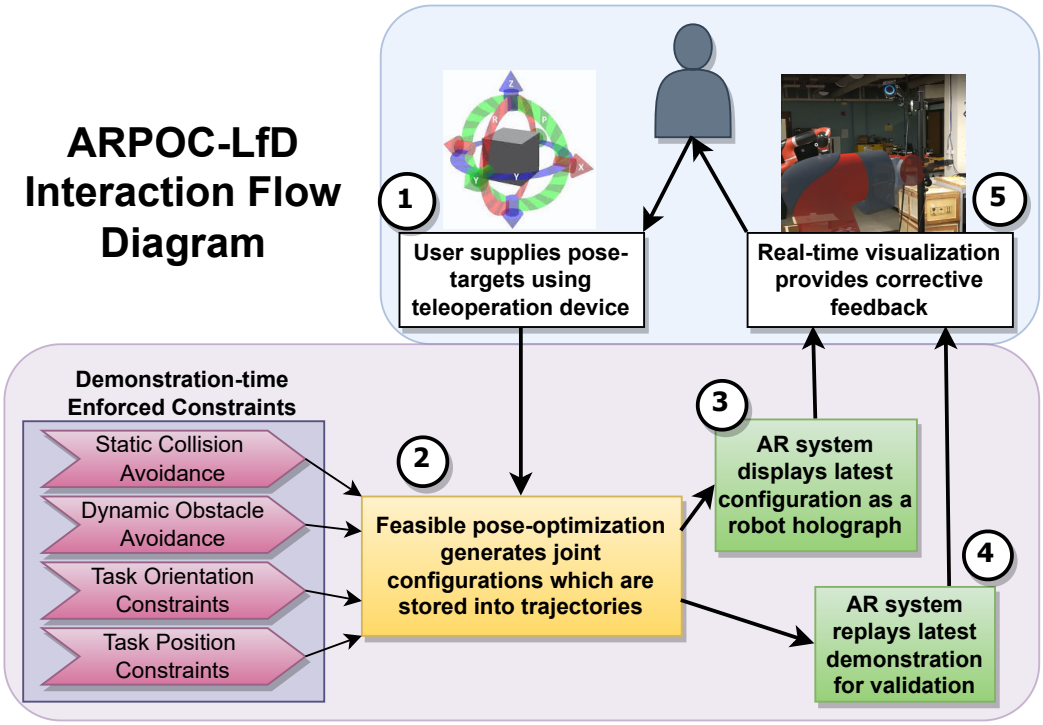


Figure 3.13: The ARPOC-LfD interaction flow diagram. The pink region (bottom) indicates the AR-based steps whereas the blue region (top) indicates how the user interacts with the system. The general interaction flow is ordered 1-5.

The final process in the interaction with ARPOC-LfD (already hinted at in Steps 3 and 4), is to utilize the visual feedback of the latest robot holograph or the entirety of the trajectory to ensure that what they've demonstrated matched their intent. One of the major challenges with using devices as proxies for teleoperation, as outlined in section 3.1.2, is the lack of control of the degrees of freedom of the robot. Furthermore, the instrumented tongs do not necessarily provide an immediate understanding of the robot agent's capabilities (e.g. reachability) or if the robot will

potentially execute the intended behavior improperly.

3.3.3 Hypotheses

Given this interaction flow, the hypotheses associated with this system are as follows:

(1) In situ real-time visual feedback in the form of augmented reality holographs acts as visual feedback for users to self-correct instrumented tong-based teleoperation to produce feasible demonstration.

(2) Instrumented tongs with visual feedback will produce the highest quality demonstrations relative to non-AR-assisted tong-based teleoperation and relative to kinesthetic demonstration.

(3) Users will score the instrumented tong-based teleoperation form of demonstration the highest according to subjective measures.

3.3.4 Experiment Protocol

This section outlines an experimental protocol that can be used to test these hypotheses. In order to compare how ARPOC-LfD compares with other forms of Learning from Demonstration modes of interaction, this protocol will compare: 1) kinesthetic demonstration, 2) instrumented tong demonstration (without AR visualization), and 3) instrumented tong demonstration with AR visualization. This protocol uses a between-subjects design with subjects randomly assigned to one of three conditions corresponding to the three interfaces/interaction modes. Within each of these conditions, users would demonstrate at least three demonstrations for three tasks. The tasks consist of a mailbox opening task, a glue tracing task, and a stacking task. These tasks were chosen to proxy real-world useful problems. After completing demonstrations for each task, participants will complete a final, post-experimental survey to test users' perception of the interface's usability and confidence in the system, as well as to obtain open-ended feedback and gather basic demographic data. Demonstration trajectories will undergo a post-experiment analysis using qualitative

measures of demonstration quality.

3.3.4.1 Experiment Conditions

Condition 1 - Kinesthetic Demonstration: Kinesthetic demonstration involves a user physically moving a robotic manufacturing arm through the intended skill, tracing out the trajectory the user is attempting to teach the robot. During any single demonstration, the robot will be in a free-movement mode called "zero-g." This is a mode in which the robot attempts to counteract the force of gravity to maintain its current position, but can easily be moved in any direction and speed by the user. The robot is not capable of any movement in this mode that is not physically initiated by the user.

Users will be physically interacting with a Rethink Robotics Sawyer collaborative manufacturing robot arm. Sawyer is a 7-DOF (degree of freedom) robot arm with 1260mm of reach, specifically designed to work alongside human collaborators. As such, it is equipped with built-in safety features. Its weak joint torques combined with soft padded joints and torque-limit sensing protect against the application of any dangerous forces to human collaborators. It is also equipped with a kill switch that will immediately shut down the arm if triggered.

Condition 2 - Instrumented Tong Demonstration, no AR: The instrumented tong input device resembles an ordinary pair of kitchen tongs, with minimal sensors/motion tracking apparatus attached to it. To provide demonstration trajectories, participants will use the device to mimic the robot's end effector. They will perform the task as if they were the robot, using the tongs to manipulate objects for the task. The position and rotation of this device will be continuously tracked throughout the demonstration to provide trajectory data using OptiTrack. The physical Sawyer robot will not be present in this condition - participants will only interact with the instrumented tong device.

Condition 3 - Instrumented Tong Demonstration with AR Visualization: This condition involves the same input device as Condition 2, but with additional visualization provided by the Microsoft HoloLens 2 augmented reality head-mounted display. Participants will be wearing

a HoloLens 2 headset while demonstrating tasks using the instrumented tong. Through their headset, participants will be able to see a virtual 3D model of Sawyer following the instrumented tong with its arm, providing real-time visualization of how Sawyer would perform the task. After providing a demonstration, participants will have the ability to rewatch the demonstration by issuing a replay command with their headsets.

Constraint-compliance Assistance: For the condition that utilizes AR-based visualizations, users will have the option to toggle between free pose-tracking teleoperation and constraint-compliance-assisted teleoperation. This assistance will be task specific to certain specific subtasks of the experimental tasks outlined below.

3.3.4.2 Tasks

Participants provided demonstrations for three tasks, designed as simplified proxies of robotic collaboration tasks, such as block stacking, opening a cabinet, tracing specified patterns, etc. All of these tasks are designed to provide challenging precision that can make one form of demonstration easier than the other. While certain tasks might favor the precision of kinesthetic demonstration, others might favor the convenience and ease of the data tongs.

Task I - Mailbox Opening In this task, users are required to demonstrate opening a mailbox (see Figure 3.14). The following subtasks are required for a successful demonstration:

(1) The first step is to open the mailbox placed on the side of the environment.

(2) Then a small block must be picked up.

(3) The block will then be placed into the mailbox.

(4) Finally, the mailbox must be closed.

This serves as a potential real-world task in which things must be opened and closed and items must be placed within. The opening of the mailbox is a particularly challenging movement to demonstrate, especially kinesthetically. The end-effector must be directed to the right distance

away from the handle for the gripper to grasp the handle. And the motion of opening and releasing the door requires careful precision. This difficulty makes the instrumented tongs a much easier form of demonstration. The constraint-compliance assistance for this task will be to maintain a horizontal orientation of the end-effector when placing the object into the mailbox.

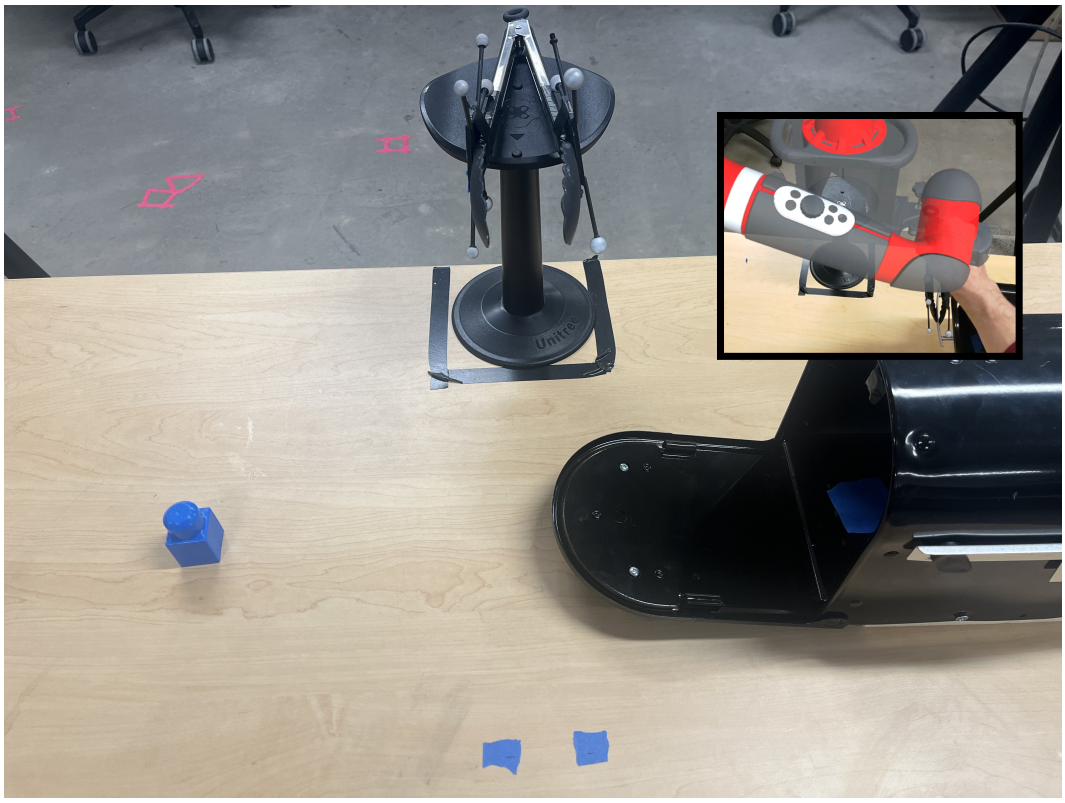


Figure 3.14: The mailbox opening task where users must open the mailbox, place a block within, and close the door. *Inset*: Example visualization during the mailbox opening portion of the skill.

Task II - Glue Tracing In this task, users must demonstrate tracing a fake glue stick around the perimeter of an object (see Figure 3.15). The following subtasks are required for a successful demonstration:

(1) They must first direct the robot agent to pick up the glue stick from a jar.

(2) The must then avoid an unregistered collision object (it will not be included in any collision avoidance mechanism).

(3) Then then must trace around an object on the workbench of the environment.

(4) Finally, they must return the glue stick back to its holder.

This task is also challenging to demonstrate, however, it is likely to be more advantageous for kinesthetic demonstration as the instrumented tongs approach might not have the fidelity needed to successfully provide a demonstration. The constraint-compliance assistance for this task will be to help users maintain a consistent vertical end-effector orientation during the tracing portion of the task.

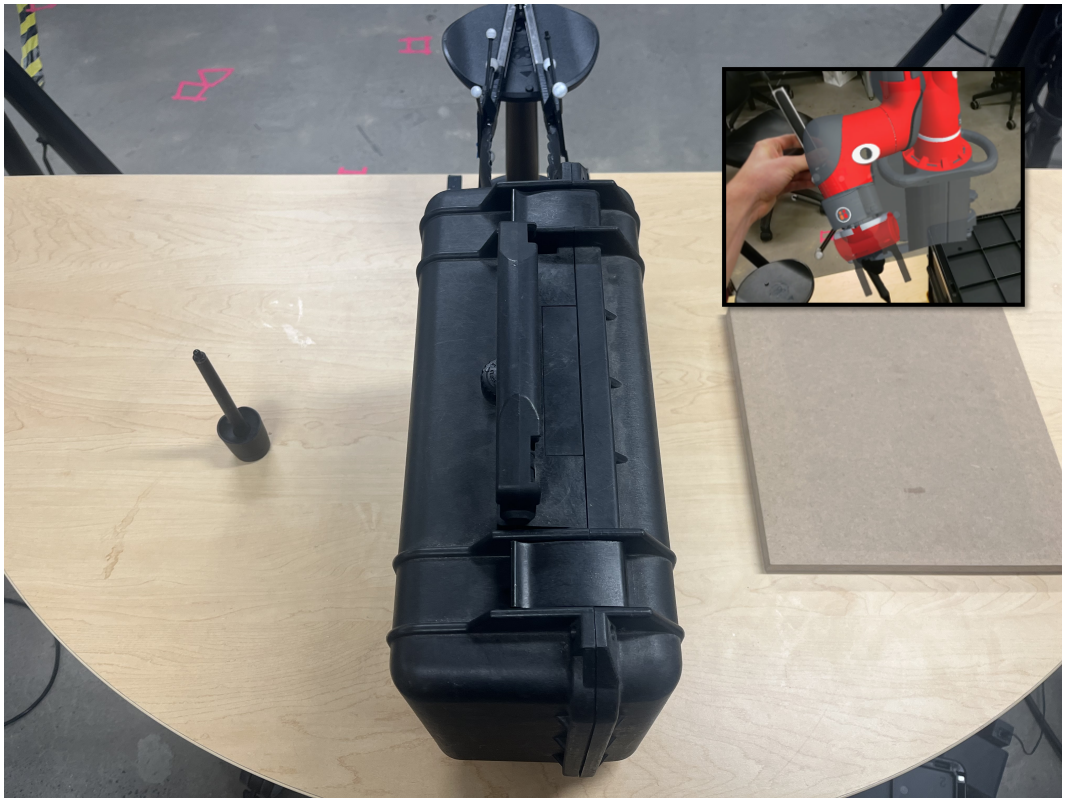


Figure 3.15: The glue tracing task where users must pick up the gluestick and trace the object on the workbench, then return the stick to its holder. *Inset*: Example visualization where the tracing object is held by the tong.

Task III - Stacking In this task, users demonstrate stacking multiple objects in-order in the center of the workspace environment (see Figure 3.16). The following subtasks are required for a successful demonstration:

(1) The user will pick up each block in order.

(2) They must then stack the object one atop the other so that they do not fall.

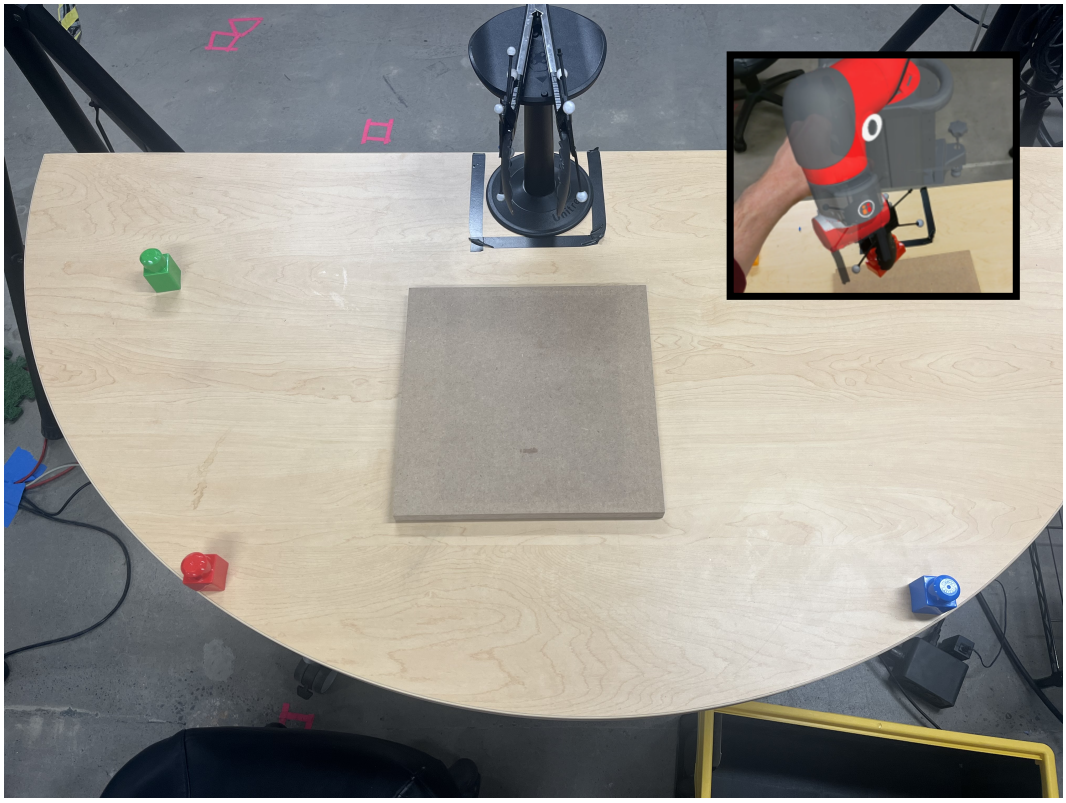


Figure 3.16: Stacking task in which the user must demonstrate pickup up four objects and stacking them in order in the center of the workbench. *Inset*: Example visualization where the user is stacking the red block.

Each object is placed on the corners of the workspace near the limits of the robot's reachable operating space. The purpose of this task is to stress the importance of demonstrating in a manner in which the robot can feasibly reach the block. This will be most important for the instrumented tongs as users might not be aware that the general sequence of poses demonstrated to the system results in robot joint configurations that do not reach the block. The constraint-compliance assistance for this task will be an end-effector centering assist in the X and Y dimensions co-planar with the workbench. Orientation and vertical height will be up to the user.

3.3.4.3 Experiment Procedure

For each task, participants will provide at least three demonstrations (though they are free to discard demonstrations and retry of their own volition, meaning they can perform more than three demonstrations until they are satisfied). Trajectory data from the demonstrations will be recorded for post-experimental analysis to evaluate the quality of learned skills as performed by Sawyer. This post-experiment evaluation will not involve participants. No execution of demonstrated skills on the Sawyer robot will be conducted in the presence of participants. If given consent, recordings will be taken during the task rounds for data coding and post-hoc analysis. After participants provide their demonstrations for each task, they will complete a short survey on their confidence in the robot's learned skill, as well as assess the difficulty of the task. Participants will engage in a single visit consisting of around 60 minutes.

After arriving for their allotted experiment time, subjects will read and sign a consent form and then review a study information sheet. Participants will be randomly assigned to one of the three experimental conditions (see above). The proctor will explain the task of human-to-robot skill demonstration. Each participant will be granted a 5-minute practice session in order to familiarize themselves with:

- **Condition 1** - Operating the robot in 'zero-g' mode of Sawyer as well as giving sample demonstrations.
- **Condition 2** - Using instrumented tongs to provide sample demonstrations.
- **Condition 3** - Using instrumented tongs to provide sample demonstration. They will also have a brief calibration step with the HoloLens 2, which makes the visualizations appear more clearly by adjusting for the inter-pupillary distance of the wearer, and ensuring that the visor is positioned properly in front of the wearer's eyes.

Following these onboarding steps, participants will perform the three task demonstrations outlined above. For each task, participants will perform skill demonstrations using the respective

interface based on their assigned experimental condition. The PI and/or IRB-approved assistants will protocol the experiment from start to finish. Once participants are satisfied with the given number of demonstrations for a task, they will complete a post-task survey. Upon completion of all tasks satisfactorily, users will complete a post-experiment survey.

3.3.5 Evaluation Protocol

This section will outline an evaluation protocol used to test the hypotheses presented above. This evaluation will include both subjective and objective measures that quantify user experience and the characteristics of the provided demonstrations.

3.3.5.1 Objective Demonstration Evaluation

In order to differentiate between the three interaction mode conditions, the following objective demonstration metrics will be used to determine demonstration quality:

Demonstration Trajectory Mean Warping Distance: Each user's set of demonstrations will be used to calculate a mean warping distance. The set of demonstration trajectories will undergo a Gaussian Mean Regression (GMR) methodology to generate a representative trajectory of the set [40, 29, 3]. This trajectory will serve as the candidate trajectory for the Dynamic Time Warping distance measure, which will be used to calculate the mean and variance of the set of trajectories warped against the GMR-produced candidate.

Physical Robot-Feasibility Percentage: Each user's set of demonstrations will be used to test for robot execution feasibility. This will be a percentage. Kinesthetic demonstrations will likely have a 100% success rate given that demonstration trajectories are intrinsically feasible in the modality.

Task Execution Feasibility Percentage: The last objective measure will test for a demonstration trajectories ability to successfully complete the task when executed on a physical agent.

3.3.5.2 Subjective Evaluation

In addition to the above objective measures, an array of subjective measures will be used to ensure that the demonstration trajectories are of appropriate quality according to robotics experts and to assess the usability of, trust/confidence of, and preference for the varying demonstration modalities.

Expert Demonstration Analysis: Each user's set of demonstrations will be analyzed by 3 non-study affiliated robotics experts and ranked on a scale of 1 to 10 on their belief that the trajectory is a proper demonstration for the task.

User Questionnaires: As mentioned in the experiment procedure, each user will conduct a post-task and post-experiment survey. The post-experiment survey will conduct a System Usability Score (SUS) assessment [23] and a trust/confidence assessment [88], and an explainability assessment using Likert scale questions to assess the value of visualizations. The post-task survey will use the NASA Task Load Index assessment [54] and a reduced trust/confidence using Likert scales [88].

Chapter 4

Combining Constrained Motion Planning and Learning from Demonstration

Chapter's 2 and 3 outline novel methods and systems for generating context-rich Learning from Demonstration models through the use of task-space constraints. For end-to-end success, these models must rely on automated motion planning algorithms for execution. Given that CC-LfD, ARC-LfD, and ARPOC-LfD all utilize constraints in some form, the underlying motion planning algorithms must produce executable trajectories (either of states or controls) that respect these constraints. As such, constrained motion planning methods generate solutions to planning problems that demand robotic agents adhere to specified behavioral restrictions. These methods rely on the ability to produce constrained points for use in traditional sampling-based motion planning algorithms, but this process is generally more computationally costly than unconstrained approaches.

Traditionally, constrained motion planning problems require adherence to a single set of constraints from start to finish. However, in the case of CC-LfD models in which users can define varying constraints over a skill, a more complicated planning problem arises because the solution trajectory must comply with multiple changing sets of constraints. As constraints define implicit manifolds within the planning space of the agent, solutions to these problems must traverse sequentially intersecting manifolds. This is known as a *Sequential Manifold Planning Problem* (SMPP). The choice of which manifold intersection points to traverse within a solution plays a critical role in solving SMPPs, as a particular intersection point may not connect to an intersection at a subsequent constraint transition, preventing motion planners from finding solutions in a reasonable or even finite time. This chapter outlines how the CC-LfD model intrinsically defines an SMPP

and contributes an algorithm for *Intersection Point Dependence Relaxation (IPD-Relaxation)* that utilizes distributions extracted from this model that denote changes in constraints. These distributions supply candidate points for an optimization process to produce correct intersection points, solving SMPPs with much greater efficiency than uninformed approaches. See Figure 4.1 for an overview how where IPD-Relaxation fits into the general LfD pipeline.

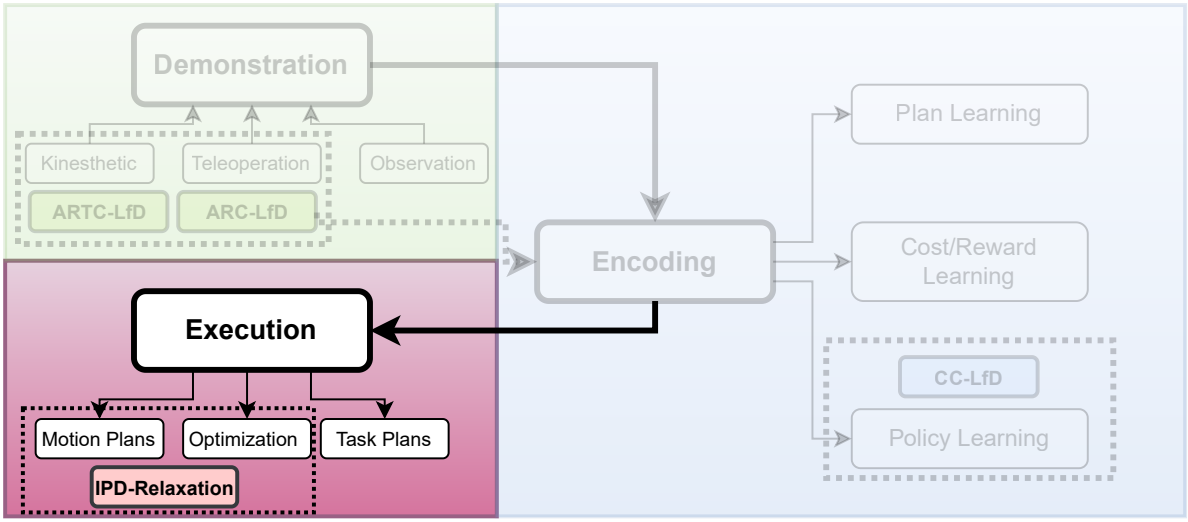


Figure 4.1: Referring back to Figure 1.1, the IPD-Relaxation algorithm combines motion planning methods with point-based optimization to generate more efficient solutions for Sequential Manifold Planning problems in order to generate constrained *execution* for LfD models.

4.1 Constrained Motion Planning Preliminaries

This section provides the necessary background for constrained motion planning, sequential manifold planning problems, and how CC-LfD models both define and help solve such problems.

4.1.1 Sampling-based Motion Planning

Motion planning algorithms solve the fundamental problem of producing fluid and feasible movement of a physical agent between arrangements. For robotic manipulators, motion planning methods supply states (or controls) that serves as a path to move the manipulator from one configuration to another. Often the representation of robot configuration is through the lens of topology,

where a robot's configuration is represented by a point on a manifold. For example, a two-degree-of-freedom robot manipulator arm's configuration (without joint limits) can be represented as a point on the surface of a torus (see Figure 4.2).

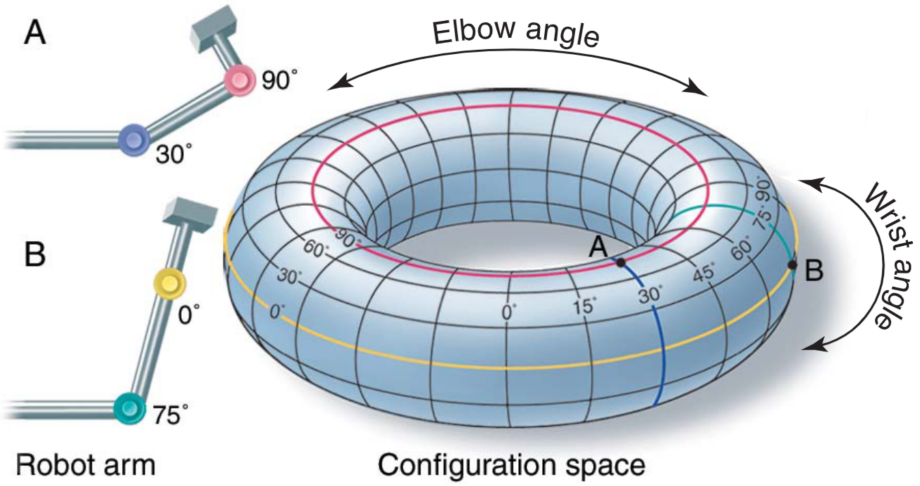


Figure 4.2: The joint configuration space of a 2-DOF arm is represented by a torus, which is the resulting manifold surface of the product topology of two circles. Each circle represents the angle of each of the robot's two joints, respectively, as seen by the degree markers in the figure. A point on the surface concurrently represents both joint angles. The figure is taken from [91].

Paths provided by motion planning algorithms are essentially paths along the surface of the manifold representative of the robot's configuration space [84]. When the robot is free to move without any restrictions (self-collision or otherwise) linear interpolation often serves as a convenient motion planning method. However, the introduction of self-collision, collision objects, joint limits, and other motion restrictions reduce the configuration space of a robot in a way that challenges simplistic motion planning algorithms. The representation of these considerations on the manifold surface is rarely explicitly defined, and thus finding feasible motion plans becomes challenging. Configuration space graph discretization and potential field methods [84] accommodate such restrictions to a point, but struggle in higher-dimensional configuration spaces. Stochastic sampling-based motion planning algorithms like Rapidly-Exploring Random Trees (RRT) [85] and Probabilistic Roadmaps (PRMs) [68] serve as basis methods for an abundance of successful mo-

tion planning algorithms that accommodate these considerations in high-dimensional configuration spaces. These methods achieve an approximate coverage of the manifold surface through random sampling of configuration points. Paths between sampled points are tested for feasibility (usually through local linear interpolation and collision/state validity checking) and then stitched together to generate a feasible solution path. Oftentimes this stitching is accomplished through splining methods for creating smooth and low-jerk trajectories [83].

4.1.2 Constrained Motion Planning

One challenging twist to these algorithms is when paths are confined by task-specific behavioral restrictions or constraints. As an example, a robot manipulator might be required to maintain a certain orientation during its motion. Such motion planning problems restrict the configuration space X to the set points where each point q in the set is compliant with a set of constraints C:

$$X_c = \{q \in X : C(q) = 0\} \tag{4.1}$$

It would appear that such constraints can be sampled directly within configuration space, but as such constraints are often defined and evaluated for validity in task-space, the constrained configuration space X_c may be explicitly undefined. Thus X_c is only defined implicitly by the task-space constraint definition (orientation, position, etc,.). For example, a constraint might require a robot to maintain contact to a task-space object. This could potentially restrict one or more of the dimensions of a robot's configuration to a restricted or singular set of values, effectively reducing the dimensionality of the planning space relative to X. If these values are not explicitly known a priori, the probability of sampling a constraint feasible configuration space point is effectively zero. To overcome this challenge, there are a number of approaches that draw from notions of gradient-descent optimization and notions of topological definitions of manifold surfaces.

4.1.2.1 Constraint Definitions

It should be noted that throughout this document the constraints employed are *scleronomic* (i.e. time-invariant) holonomic constraints, that is, for the state of the robot $q = \{q_1, q_2, \ldots, q_i\}$, one can write the constraints of the form:

$$c(q_1, q_2, \ldots q_i, t) = 0, \tag{4.2}$$

that is, the constraints are a function of the state variables and may be expressed as an equality. Equation (4.2) implies that a sufficient condition for holonomicity is that the equation has an exact differential form. Constraints that cannot be expressed as above are known as non-holonomic constraints, often referred to as non-integrable constraints. For efficiency, this dissertation affords the constraint equality a tolerance which relaxes the equation to an inequality constraint. This creates what are known as volume-swept constraints [75], but, given a small enough tolerance, is a close approximation for the holonomic constraints defined above [135, 17].

4.1.2.2 Orthogonal Sampling

An important point about constrained motion planning methods is that the generation or sampling of constrained points is orthogonal to traditional sampling-based algorithms. If constrained points for a given problem are producible, then all traditional sampling-based motion planning algorithms become feasible [76]. In other words, the sampling of constraint-compliant points becomes independent of the planning algorithm. This enables the use of RRT-like and PRM-like motion planning algorithms without significant alterations to their standard implementation.

4.1.2.3 Point-projection Methods

To retain the benefit of sampling-based methods, the majority of constrained motion planning methods rely on the ability to produce constraint-compliant configuration points. As mentioned

above, it is not usually possible to directly sample such points in configuration space. However, constraints can usually be explicitly represented in the task space of the robot (also called end-effector space) [17]. While constraint-compliant points can be generated by sampling in this space and performing inverse kinematics, one cannot provably show such an approach sufficiently covers the feasible planning space to guarantee probabilistic completeness for sampling-based planners [17]. Through an iterative process as shown in Figure 4.3 we generate an error signal in this space that we map into configuration space. This mapped signal is used to push a configuration point toward constraint compliance. In other words, we can project a randomly sampled configuration q onto the implicit constraint manifold embedded in the configuration space.

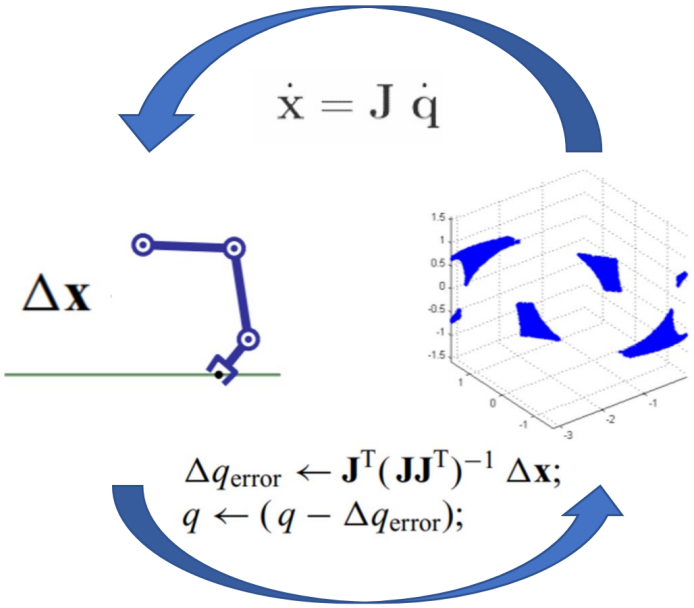


Figure 4.3: The iterative approach to projecting a configuration point q onto an implicitly defined constraint manifold embedded in configuration space. Portions of this figure were taken from [18]

The process is as follows according to Figure 4.3. A randomly sampled configuration space point q is mapped to task space using the current Jacobian of that point. A 'distance-to-constraint', Δx, is produced in task space. This is the error signal mentioned above. For an orientation constraint, this distance could be the axis-angle differentials. For a positional constraint, this could be the Euclidean distance of the end-effector to the ideal constrained position. This distance

is mapped back to configuration space using the pseudoinverse of the Jacobian, which creates a configuration space error Δq_{error}. This error signal is used to update the current configuration q towards the constraint manifold. This approach is a type of approximated gradient-descent, which is also known as a First Order Retraction [135] in the spirit of manifold retraction methods.

4.1.2.4 Manifold Theory Methods

Nearly all constrained motion planning techniques that rely upon sampling-based methods utilize the above projection method in one way or another to produce constraint-compliant points. Even more advanced methods that employ neural network architectures still rely on this method of projection to produce training data or provide corrective adjustment to points generated from such architectures [114]. As such, there are some methods that attempt to take advantage of the definition of a manifold to help improve the efficiency of producing constraint-compliant points for planning.

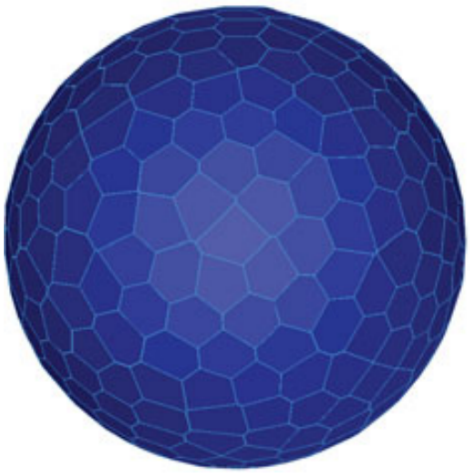


Figure 4.4: A collection of charts that approximate a sphere. As the number of charts increases, the approximation error decreases. Image taken from [64]

Some constrained motion planning methods leverage concepts from the atlas-based description of a manifold. Using the idea of navigational atlases and charts, we can use the surface of the

earth as an analogous example. If the surface of the earth is our manifold surface, we can think of defining the surface as a collection of charts that form an atlas. Informally, a manifold is a topological space that is locally Euclidean. In navigational terms, each chart represents a patch of the surface of the earth that is represented as locally Euclidean. In other words, we have mapped a curved surface section of the earth to a Euclidean representation that is a navigational chart. See Figure 4.4 for a visualization of this representation on a sphere. The collection of all charts (individual patches in Figure 4.4) constitutes the atlas of the earth/sphere. It is this atlas that defines the 'earth' manifold. The key addition to this analogy is that we must have the means to transition from one chart to another. In more formal terms, this means we need to have a smooth transition function from one overlapping region of a chart to its neighbor.

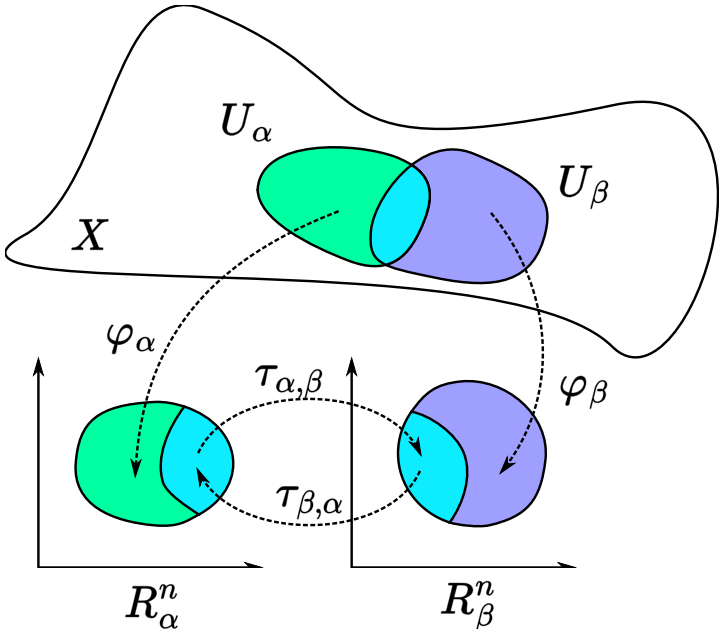


Figure 4.5: A simplified visual to aide with the concept of the Atlas-Chart definition of a manifold.

Figure 4.5 provides a visualization of this concept. Given some manifold X, we choose two approximately close points and a local neighborhood around each point U_α and U_β. Each of these neighborhoods is homeomorphic to Euclidean space via the mappings $\varphi_\alpha : U_\alpha \to R^n_\alpha$ and $\varphi_\beta : U_\beta \to R^n_\beta$. The transition functions $\tau_{\alpha,\beta}$ and $\tau_{\beta,\alpha}$ map the coordinate system of one

chart to another for the corresponding overlapping regions. The nature of these mapping functions (differentiability, etc,.) bring added structure to the definition of a manifold. An atlas is therefore a collection of these homeomorphic mappings and transition functions according to an indexing family I defined as $\{(U_\alpha, \varphi_\alpha) : \alpha \in I\}$. And it is the union of these charts that defines a manifold $X = \bigcup_{\alpha \in I} U_\alpha$.

From the perspective of constrained motion planning, the notion of a manifold being locally homeomorphic to a Euclidean space can be leveraged to avoid constant projection sampling as outlined in the previous section. Ultimately methods like AtlasRRT [64], Tangent-bundle RRT [71], tangent-space RRT [136], and others assume that the tangent-space of a point known to be on the manifold can serve as an approximation of the locally Euclidean space that the neighborhood around that point maps according to some implicit chart. These methods sample within tangent space up to some distance away from the central point, avoiding the costly projection step. The assumption is that any point close to this central point within the tangent space will be nearly on-manifold and can be used to adhere to the constraints represented by the manifold.

4.1.2.5 Task Space Regions: A Framework for Constraint Representation

In order to facilitate the convenient implementation of sampling constrained points using the above projection method, this dissertation employs the framework of Berenson et al. called Task Space Regions [17], a representative framework that extends the work of [135]. This framework represents constraints as a sequence of virtual transformations from the end-effector to an object or for a target position, orientation, or pose for the end-effector. Figure 4.6 provides an overview of the sequence of transformations used to represent a basic task-space constraint that desires to keep an object (i.e. the red soda can) in a specific orientation. To provide a mechanism for generating the Δx, this framework utilizes a bounds matrix B that provides threshold 'distances' or error along each dimension (e.g. X, Y, Z, R, P, Y) of the current end-effector to the desired pose constraint. These errors are used to then create a vector of errors, the norm of which becomes the norm distance or Δx for use in the Jacobian projection method.

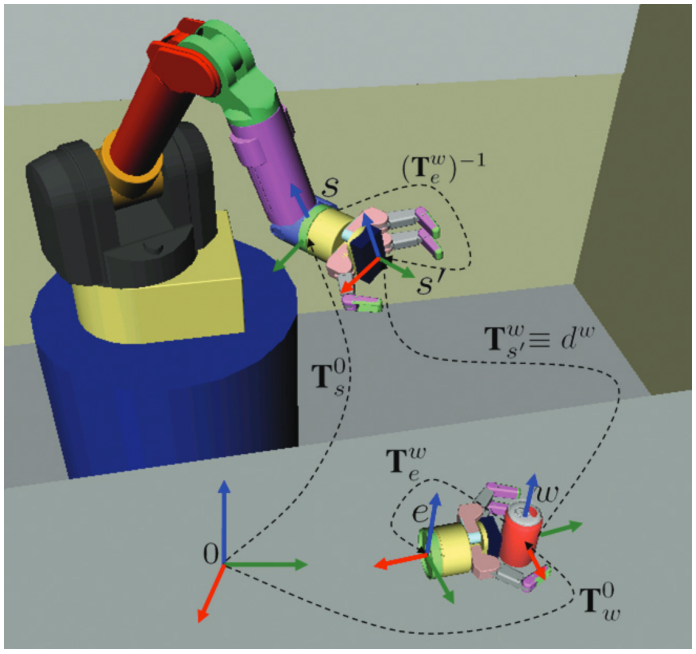


Figure 4.6: An example visualization taken from the Task Space Region framework of [17]. It outlines a series of transformations that collectively represent a task-space constraint. The first transformation (T_s^0) moves from the global origin 0 to the current end-effector pose s. The second ($(T_w^e)^{-1}$) moves the the end-effector e to the virtual grasp pose w. The last transform ($T_{s'}^w$) moves from the current pose to the object $s^{'}$ to the pose-target of interest w. This last transformation provides an easy mechanism to calculate a Δx distance for use in the Jacobian projection method (e.g. d^w).

4.1.3 Biased Sampling in Motion Planning

One of the conditions for sampling-based motion planning algorithms to achieve probabilistic completeness is their stochastic sampling characteristics, usually through uniform random sampling of the planning space [137, 78, 86, 67, 18, 17, 63, 84]. However, a drawback of uniform sampling is the wasted effort of sampling points that do not contribute to a solution. And in the case of constrained motion planning, this adds additional unnecessary computation during the projection process mentioned previously. An approach to overcome this issue is to bias sampling such that it produces a population of points whose distribution more closely covers an expected solution space (with a fraction uniform to retain probabilistic completeness) [63, 19, 149, 141, 70, 5, 30, 21]. For example, narrow corridor problems that usually challenge uniform sampling approaches might see

a benefit if the sampling bias is within the corridor.

4.1.4 Sequential Manifold Planning Problems

All of the constrained motion planning techniques introduced above traditionally focus on a single set of constraints during planning. A more challenging problem arises when a solution trajectory must adhere to changing constraint requirements. Constrained motion planning requires a trajectory to traverse along the surface of an unknown manifold in the agent's configuration space that represents constraint-valid configurations. With changing constraint requirements, the trajectory must traverse a sequence of these intersecting constraint manifolds (see Figure 4.7), avoiding pitfalls such as manifold intersections that result in infeasible planning. This is called a *Sequential Manifold Planning Problem (SMPP).*

While constrained motion planning algorithms produce feasible trajectories over a single set of constraints, many real-world problems require planning over multiple changing constraint sets or conditions. *Task and motion planning (TAMP)* methods treat such planning problems as solving for connected sequences of subtasks each differentiated by varying modes defined by environmental conditions [50, 55, 144, 44, 66, 39, 73, 38]. This is also referred to as *multimodal planning.* The combined solutions to the sequence of subtasks create a solution to the global task or motion planning problem.

In most TAMP/multimodal methods, switching between subtasks/modes occurs under specific conditions, such as contact events or precise change points in the task. In other words, points chosen as the boundary/transition between modes are usually single points associated with conditions accounting for a change in constraints, agent behavior, or in the environment. For example, in Englert et al. [44] intersection points are chosen as discrete contact events such as picking up an object. Work by Kingston et al. presupposes that many planning problems are inherently discrete in nature [77, 74]. However, these approaches do not adequately handle conditions where constraints change independently (perhaps at a user's discretion) of environment events or agent behavior.

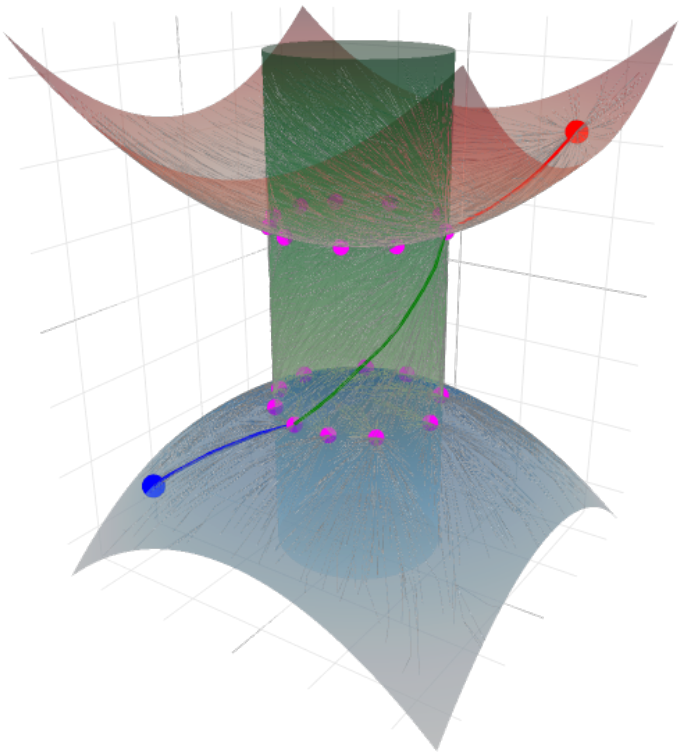


Figure 4.7: A simplified visual to aid with the concept of Sequential Manifold Planning. A solution path is visible as a line that traces the surface of three 'manifolds'. In this case, the manifolds are simple mathematical curves (parabolic and cylindrical). Pink points represent manifold intersections and they act as anchoring points for motion planners to plan along one manifold, then the next, and so forth. Generally, constraint manifolds in configuration space cannot be explicitly defined as such. Produced by software associated with [44].

4.1.4.1 How CC-LfD Models Define SMPPs

The CC-LfD algorithm introduced in Chapter 2 enables users to define a multimodal planning problem at their discretion. A solution trajectory must traverse a sequence of intersecting constraint manifolds (as specified by the user), each representing a unique set of constraints. In order to produce this trajectory, points on the intersections of adjacent overlapping manifolds must be chosen such that the next intersection point is reachable. This problem is a generalized version of the Sequential Manifold Planning problem (SMPP) introduces above. A CC-LfD-defined SMPP is uniquely challenging because planning for discrete changes in constraints no longer relies on obvious change point events as expected in other approaches [44]. Herein lies a main contribution

of this work: namely, that CC-LfD defines the SMPP and provides the means to generate solutions efficiently through a more general intersection point choice selection approach.

Borrowing the definition from Englert et al. sans notions of path optimality [44], an SMPP is formulated as:

$$
\begin{aligned}
& \boldsymbol{\tau} = (\tau_1, \ldots, \tau_n) \\
& s.t. \\
& \tau_1(0) = q_{\text{start}} \\
& \tau_i(1) = \tau_{i+1}(0) && \forall_{i=1,\ldots,n-1} \\
& C_{\text{free},i+1} = \Upsilon(C_{\text{free},i}, \tau_i(1)) && \forall_{i=1,\ldots,n} \\
& \tau_i(s) \in C_{M_i} \cap C_{\text{free},i} && \forall_{i=1,\ldots,n} \quad \forall_{s\in[0,1]},
\end{aligned}
\tag{4.3}
$$

where $\boldsymbol{\tau}$ is a sequence of paths that traverses through free configuration space and along the geodesic paths of sequential intersecting manifold surfaces. The second constraint ensures that the ending point of one path is the start point of the subsequent path. These points are referred to as *steering points* or *intersection points*. The third constraint implies that each segment is associated with its own collision-free space. The fourth ensures the path is both collision-free and on the manifold surface. In other words, it is constraint-complaint for manifold C_{M_i}.

4.1.4.2 Intersection Point Dependence/Independence

The use of these intersection points is key to a global solution to an SMPP. SMPPs are solved by performing sequential constrained motion planning steps for each manifold. The intersection points serve as anchors within overlapping constraint manifolds. These points act as the start and end points for constrained motion planning on each manifold stringing together individual solution trajectories into a global solution trajectory. The major challenge presented by this solution characterization is in choosing the correct intersection points that enable feasible planning.

Due to configuration space discontinuities/singularities the constraint manifold that must be planned on may be either disjoint or foliated [44, 73, 96, 72]. A foliation of a manifold M is

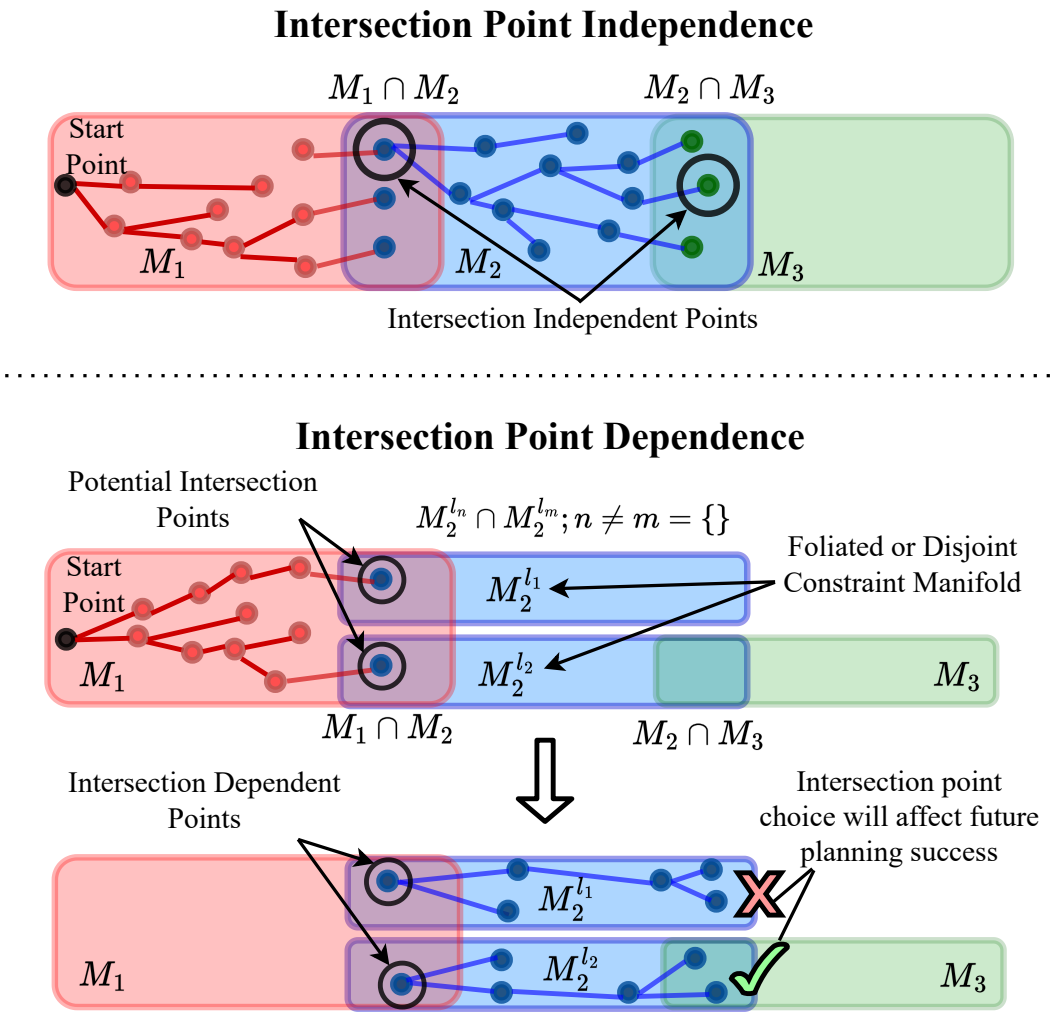


Figure 4.8: *Top*: simplified sequential manifold planning problem (SMPP) with intersection point independence. *Bottom*: outline of intersection point dependent SMPP.

the decomposition of that manifold into submanifolds or leaves L_i that do not intersect. In other words, $M = \bigcap_{i=1}^{N} L_i$ where $L_n \cap L_m = \{\}; m \neq n$. Such manifolds might arise due to collision object occlusion of the manifold or particular grasp poses needed to achieve an orientation-constrained behavior. A prior or subsequent manifold might only intersect with $N-1$ of these leaves or disjoint sets, with the correct intersection choice unknown.

Solving an SMPP depends on reaching an intersection of the current and subsequent manifold for each constraint manifold in the sequence. However, there is no guarantee that a path within a chosen leaf can feasibly reach the intersection of the subsequent manifold (if it intersects at all).

This introduces the concept of *Intersection Point Dependence* (IPD) as outlined in [44]. As seen in the bottom half of Figure 4.8, the choice of the intersection $p \in M_1 \cap M_2$ will affect the success of the planner reaching the set of points $p \in M_2 \cap M_3$. Only the bottom intersection region provides intersection points in $M_1 \cap M_2$ that successfully enable planning to the set of points in $M_2 \cap M_3$. This is in contrast to an *Intersection Point Independent* (IPI) SMPP exemplified in the top half of Figure 4.8 for which the choice of intersection point has no bearing on the success of the problem so long as they meet the constraint criteria of the manifold overlap.

We propose that learned models from human demonstrations of approximate solutions to the planning problem provide enough information to relax an intersection point dependent (IPD) SMPP to an intersection point independent (IPI) SMPP. This information directs the appropriate choice of intersection points through an optimization process called Omega Optimization and also utilizes sampling biasing such that the planner efficiently produces a feasible solution. We refer to this process as *Intersection Point Dependence Relaxation* or *IPD-Relaxation*.

4.2 Intersection Point Dependence Relaxation

This section describes the last major contribution of this dissertation by outlining how Concept Constrained Learning from Demonstration (LfD) defines an SMPP. This section also describes how the resulting learned models can be used to obtain a set of distributions to efficiently solve an SMPP by guiding the selection of intermediate constraint manifold intersection points. Our contributions are as follows:

- A method for combining learned models from demonstration data with constrained motion planning methods to both define and efficiently solve SMPPs through a novel process called *Intersection Point Dependence Relaxation*.
- A quantitative evaluation showing how learned distributions result in a substantial increase in sampling efficiency for constrained motion planners that utilize Jacobian projection-based methods for producing on-manifold samples, supporting the above contribution by

increasing planning efficiency in scenarios that utilize human demonstration data.

4.2.1 Information Needed to Solve SMPPs

4.2.2 ρ-usefulness and the Ω-set

In order for LfD models to facilitate IPD-Relaxation, the model needs to provide information as to which intersection points will ultimately result in a feasible solution. To characterize this information, we borrow from the definition of δ-usefulness in Rakita et al. [117] to define *ρ-useful points* as the following: $Q_\rho = \{q \in C_{M_i} \cap C_{\text{free},i};\ d(q, \tau_i) \leq \rho\}$. The set of ρ-useful points Q_p are distance at most ρ from a solution path, τ_i, for a planning problem given a planning space $C_{M_i} \cap C_{free,i}$ for the ith constraint set/planning segment. This precise distance is generally unknown, but it captures an approximate region of the planning space within which feasible solutions lie. In the case of SMPPs, ρ-useful points will be those that are a distance at most ρ away from an IPD-Relaxed solution path. Our approach uses a distribution D learned from demonstration data to bias the sampling such that the $P(q \in Q_\rho; x \sim D) >> P(x \in Q_\rho; x \sim Q_{free})$. In other words, the probability that a point drawn from the learned distribution D is in the Q_ρ set is much greater than if the point were drawn uniformly from free configuration space Q_{free}.

User demonstration data also needs to supply the planner with a means to select the correct intersection point in a foliation leaf or disjoint constraint manifold intersection such that continued planning will feasibly reach the next manifold intersection. We define *Ω-points* as the set of points on intersections of sequentially adjacent manifolds that enable a feasible planning problem solution. The presumption is that demonstration data will either directly provide such points, or that demonstration data will inform the selection of such points. In a similar vein, demonstration data can also inform where a planner should move from free configuration space onto an implicit constraint manifold, as the choice without any guidance can otherwise be arbitrary.

To generate Ω-points, we introduce *Omega Optimization*: a novel intersection point generation technique that utilizes a *problem domain-specific* multiobjective optimization program to

generate intersection points that are optimized for constraint manifold intersection compliance and that minimizes the distance away from the corresponding constraint transition keyframe distribution. This process increases the likelihood that the generated intersection points are within the ρ-useful set and Omega set, conditions necessary for IPD-Relaxation. To empirically assess the performance improvements of this approach, we compare intersection point generation mechanisms (line 10 in Algorithm 1) using domain-specific omega optimization (OO), keyframe-naive constraint-only optimization (CO), and direct keyframe distribution sampling (KF). See section 4.3.1 for more details about these conditions.

4.2.3 IPD-Relaxation Formulation

Extending the SMPP formulation of [44], this paper defines an IPD-Relaxed Sequential Manifold Planning Problem in Equation 4.4, adopting C_* as a subset of the configuration space defined by some criteria $*$. $\boldsymbol{\tau}$ is a sequence of paths that traverses through free configuration space and along geodesic paths over sequential intersecting manifold surfaces. Constraint 1 ensures that a point S_i^{i+1} that is an ending point of one path is the start point of the subsequent path. In other words, S_i^{i+1} is an intersection point. Constraint 2 dictates that this point is within the Ω_i^{i+i} set of $M_i \cap M_{i+1}$. The Ω_i^{i+i} set is defined as the set of all configuration points lying on the intersection of the current manifold M_i intersecting with the subsequent manifold, specifically within the foliation/subset $M_{i+1}^{l^*}$ (with l^* unknown) that enables planning feasibility. The Ω_i^{i+i} set will inherently be a local subset of the ρ-useful set as indicated by Constraint 3. Constraint 4 implies that each segment is associated with its own collision-free space. Constraint 5 ensures the path is both collision-free and on the manifold surface. In other words, it is constraint complaint for manifold M_i.

$$
\begin{aligned}
&\boldsymbol{\tau} = (\tau_1, \ldots, \tau_n) \\
&s.t. \\
&1) \quad S_i^{i+1} = \tau_i(1) = \tau_{i+1}(0) && \forall_{i=1,\ldots,n-1} \\
&2) \quad S_i^{i+1} \in \Omega_i^{i+i} = \{q;\ C_{M_i} \cap C_{M_{i+1}^{l^*}}\} && \forall_{i=1,\ldots,n-1} \\
&3) \quad \Omega_i^{i+i} \subset \{q \in C_{M_i} \cap C_{\text{free},i};\ d(q, \tau_i) \leq \rho\} && \forall_{i=1,\ldots,n-1} \\
&4) \quad C_{\text{free},i+1} = \Upsilon(C_{\text{free},i}, S_i^{i+1}) && \forall_{i=1,\ldots,n-1} \\
&5) \quad \tau_i(s) \in C_{M_i} \cap C_{\text{free},i} && \forall_{i=1,\ldots,n} \\
& && \forall_{s\in[0,1]}
\end{aligned}
\tag{4.4}
$$

4.2.4 Constrained-LfD Keyframe Distribution Taxonomy

Constrained-LfD model distributions can be characterized as one of two types [98]:

(1) *Constraint Transition Keyframe Distributions*: Learned distributions from trajectory data at constraint set transitions, used to find productive constraint manifold intersections.

(2) *Intermediate Trajectory Distributions*: Learned distributions from trajectory data between constraint transition keyframes, used to bias constrained sampling-based motion planners to adhere to demonstration style.

An intermediate trajectory distribution exists for each segment of the SMPP between constraint-set manifold transitions. This distribution acts as a biasing distribution for the sampling of points that are then projected onto the segment's constraint manifold. The constraint transition keyframe distributions also provide candidate points that are used in a multi-objective optimization process that generates constraint-compliant intersection points as visualized in Figure 4.9. As such we hypothesize the following:

(1) Candidate points sampled from constraint transition keyframes will optimize consistently into Ω-points using Omega Optimization, achieving IPD-Relaxation. (Figure 4.9, top).

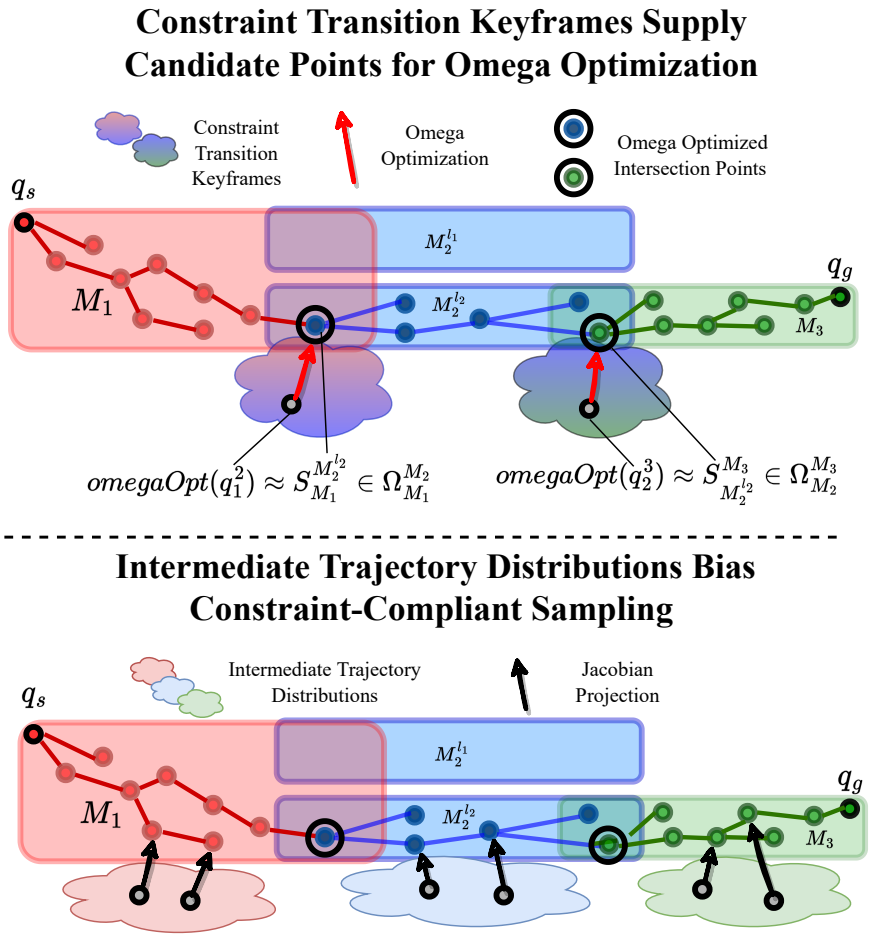


Figure 4.9: *Top:* Constraint transition keyframes (multicolored) supply candidate points that are made constraint-compliant through Omega Optimization. *Bottom:* Intermediate trajectory keyframes (uniform color) and their distributions help to bias sampling during planning.

(2) Sampling biasing using intermediate trajectory distributions will boost planning efficiency by sampling points likelier to be from the ρ-useful see (Figure 4.9, bottom).

4.2.5 The IPD-Relaxation Algorithm

Algorithm 4 outlines the process by which a planning model consisting of a series of constraint-annotated keyframes (e.g., [98]) can achieve IPD-Relaxation. A constrained keyframe model K is passed as an input. It initializes a planning graph, G_p, that serves as the basis for generating a sequence of connected plans that solve the IPD-Relaxed SMPP. The keyframe model K is traversed sequentially in reverse order over the constraint transition keyframes (lines 2-3), ignoring the inter-

Algorithm 4: IPD-Relaxation

```
   Input: K ;                                          // Planning Keyframe Sequence
   Output: G_p ;                                                  // IPD Relaxed SMPP
 1 upcomingC = K[−1].C;
 2 for k in K.reversed() do
 3     if isConstrained(k) then
 4         while ! valid do
 5             s ← sampleKeyframePoint(k.D);
 6             combinedC = set(upcomingC + k.C);
 7             if constraintValid(s, combinedC) then
 8                 break;
 9             end
10             o, valid ← omegaOptimize(s, combinedC, k.D);
11         end
12         upcomingC = K.C;
13         G_p.addNode(o, k.C)
14     end
15 end
16 return G_p
```

mediate keyframes used for constraint continuation and consistency when generating intermediate waypoints.

The motive behind reverse traversal is twofold: 1) To allow the combination of current keyframe constraints with the upcoming constraints (line 6) to ensure generated intersection points are valid for constraint overlap, and 2) To enable information from future portions of the LfD model solution to guide the choice of intersection points such as leaves of foliation classifiers [72].

From each constraint transition keyframe $k \in K$ a candidate intersection point s is drawn from the distribution $k.D$ (line 5), which is passed to the *omegaOptimize* function (line 10) that performs Omega Optimization. The formulation of the optimization program can be problem-specific so long as it integrates these candidate points and biasing terms in the objective to steer the process to converging onto the Ω-set. If the optimization call converges successfully, the resulting point is then passed into internal state and constraint validity functions within *omegaOptimize*. Some evaluation conditions either directly use this sampled point without optimization, or the optimized candidate point is instead drawn uniformly from the agent's configuration space. This optimized

Algorithm 5: omegaOptimize

Input: s, C
Output: o $omegaValid$
1 $o \leftarrow$ domainSepcificOptimizzation(w, c);
2 $omegaValid \leftarrow$ omegaTest(o, C);
3 **return o, omegaValid**

intersection point o and the associated constraints $k.C$ are added as a new node to the planning graph G_p (line 13). Once all intersection points are collected, a constrained motion planner (e.g., CBiRRT2 [17]) plans trajectory segments using G_p to create a global solution trajectory given start and end points added to G_p externally.

4.3 Evaluation

4.3.1 Evaluation Domains

We chose three representative domains to characterize and demonstrate the benefits afforded by our proposed methods to improve sampling efficiency and to achieve planning feasibility through IPD-Relaxation for SMPP-constrained motion planning tasks through the use of keyframe LfD models.

- *Domain I - Constraint Demonstration for Biasing*: Domain I shows how distributions learned from user-supplied examples of constrained robot configurations, independent of a particular task, increase the sampling efficiency of constraint-compliant points. This is to support the justification for using intermediate trajectory distributions (see Figure 4.9) to speed up planning for SMPP problems. In this domain, learned distributions from user demonstration data of a single constraint are used to bias projection sampling.

- *Domain II - 2D Navigation with Explicit Constraints*: Domain II provides a minimal environment with an explicitly designed and easily visualized Intersection Point dependency. This domain involves planning a path over the 2D plane for a 3-DOF (XY-position and orientation) holonomic agent with line-tracking constraints (see Figure 4.10).

- *Domain III - Simulated Manipulator with Implicit Manifold Constraints*: Domain III provides a scenario with an induced Intersection Point dependency in order to show the benefits of IPD-Relaxation in a realistic planning environment. This domain uses a 7-DOF manipulator arm with constraints that define implicit manifolds (see Figure 4.11).

4.3.2 Metrics

Table 4.1 outlines the metrics used for each domain. For evaluation Domain I, the metric of sampling time shows how biasing affects planning efficiency, as sampling points on constraint manifolds is the dominant source of computational overhead in constrained motion planning. For Domains II and III, we follow prior work by [44] in using metrics for Path Length, Success %, and planning time. We developed the final two metrics, *adherence to function and adherence to style (A2F/A2S)*, to assess effectiveness in capturing or preserving aspects of the human-provided signal while still realizing benefits of flexibility and capability provided by probabilistically complete motion planning techniques, as this paper represents a novel fusion of LfD with Sequential Manifold Planning.

Table 4.1: Description of Metrics for each Evaluation Domain

Metric	Abbreviation	Description	Domain
Sampling Time	n/a	Time (seconds) to sample 1000 constrained points	1
Success %	Success %	% successful planning trials	2, 3
Path Length	PL	Euclidean path length (Domain 2: pixels, Domain 3: meters)	2, 3
Adherence to Style	A2S	Dynamic Time Warping (DTW) distance in task space to 'gold' demonstration	2, 3
Adherence to Function	A2F	% of planned trajectory constraint-compliant (no tolerance allowed)	2, 3
Planning Time	PT	Time (seconds) needed to for successful plan	2, 3

4.3.3 Experimental Conditions to Evaluate IPD-Relaxation

The following conditions are utilized to validate the IPD-Relaxation algorithm:

(1) *Biased Sampling vs. Uniform Sampling:* Biased sampling from intermediate trajectory distributions vs. uniform sampling from configuration space for the CBiRRT2 planner.

(Domains II and III)

(2) *Collision Object:* Introduces a collision object that occludes the most common region demonstrated by users, creating a 'narrow corridor' condition as in Figure 4.11. (Domain III)

(3) *Intersection Point Generation Mechanism* - Intersection points for use in planning will be generated by different means (i.e. the term 'o' in line 10, Algorithm 4). (Domains II and III)

4.3.4 Intersection Point Generation Mechanism Details

The intersection point generation mechanism condition provides the most significant differentiation in validating Omega Optimization and the IPD-Relax algorithm. As such, the three-generation methods are outlined below:

(1) *Constraint-only Optimization (CO)*: The intersection point will be generated by optimizing for constraint intersection compliance, ignoring any guidance from the keyframe model. This is akin to a naïve search mechanism and uses a uniformly random sampled seed point as a start for the optimization.

(2) *Keyframe-Only (KF)*: Utilize points directly sampled from the constraint transition keyframe distributions as the intersection points.

(3) *Omega Optimization (OO)*: Combining both CO and KF conditions, the OO generation mechanism utilizes candidate points derived from constraint transition keyframe distributions for a multiobjective optimization program that maximizes constraint intersection compliance and minimizes the distance from the keyframe distribution. This condition does not guarantee that the point produced by optimization convergence is in the Omega set. However, results presented in section 6 empirically bear out the utility of the IPD-Relaxation algorithm, indicating that it efficiently chooses Omega set intersection points

even in the absence of a theoretical completeness guarantee.

Depending on the chosen sampling mechanism, intersection points generated during IPD-Relaxation could be off-manifold and beyond the constraint-compliance tolerance allowed by the CBiRRT2 planner. To level the playing field and provide a maximally fair basis for comparison, these points are injected into the tree with a relaxed ϵ tolerance to avoid simply failing to find a path entirely. This will be reflected in the Adherence to Function metric (see Table 4.1) as portions of the plan that reach these off-manifold points will not adhere to the constraints for portions of the planned segment.

4.3.4.1 Implementation Details

All evaluations were executed on an AMD Ryzen 9 5950X 16-Core Processor with 32 GB of RAM. All implementations use the Python programming language except where noted below.

- Domain II uses a simplified constrained RRT planner with the following parameters: ϵ-tolerance: 50; extension distance: 10; max planning time; 60 seconds.
- Domain III uses the CBiRRT2 planner [17] with the following parameters: ϵ-tolerance: 0.15; q-step: 0.35; smoothing time: 5 seconds. For global planning: Omega Optimization tolerance: 0.075; max SMPP segment planning time: 30 seconds. Omega Optimization tolerance is the allowed off-manifold error of the produced intersection point.
- In both domains, a bandwidth of 0.15 for Kernel Density Estimation (KDE) was used to fit keyframes and intermediate trajectory data. These parameters balance performance against time efficiency and were consistent across all conditions.

4.4 Evaluation Experiments

4.4.1 Domain I - Constraint Demonstration for Biasing

In this evaluation domain, three robotics experts demonstrate two different constraints on a Rethink Robotics Sawyer 7-DOF arm. These demonstrations are independent of any particular task and instead, users are asked to kinesthetically demonstrate constrained configurations (see inset images of Figure 4.12). The first constraint is an orientation constraint (e.g. holding a cup in an upright position). The second constraint is a line tracing constraint (e.g. glue application) restricting both orientation and position against a surface. The data from these demonstrations serve to fit KDE distributions. These distributions in turn produce candidate samples which are input into the projection operator outlined in Section 4.1.2.3. In order to evaluate efficiency gains in sampling, this evaluation utilizes varying ratios of unbiased (i.e. uniform) to biased (i.e. demonstration distribution biased) seed samples that are then fed into the Jacobian projection operator. Results for this domain are presented in Figure 4.12.

4.4.2 Domain II - 2D Navigation with Explicit Constraints

For this domain, five users each provided at least three demonstrations on a holonomic 2D agent navigating from a start point to a goal point, which were used to train a CC-LfD model. The constraints are *explicit constraints* on the path and/or angle the agent must traverse. Two of the constraints restrict the XY position of the agent (blue and red in Figure 4.10). The third constraint (in green in Figure 4.10) restricts the XY position as well as the angle of the agent as it must target the center of a black 'X' in the middle of the planning space. The agent must adhere to the red, green, then blue constraints on its way to the goal. The task is an IPD SMPP because the agent must choose the upper intersection between red and green constraints (Ω-set in Figure 4.10). The lower red-green constraint intersection (lower red-green intersection in Figure 4.10) results in planning failure.

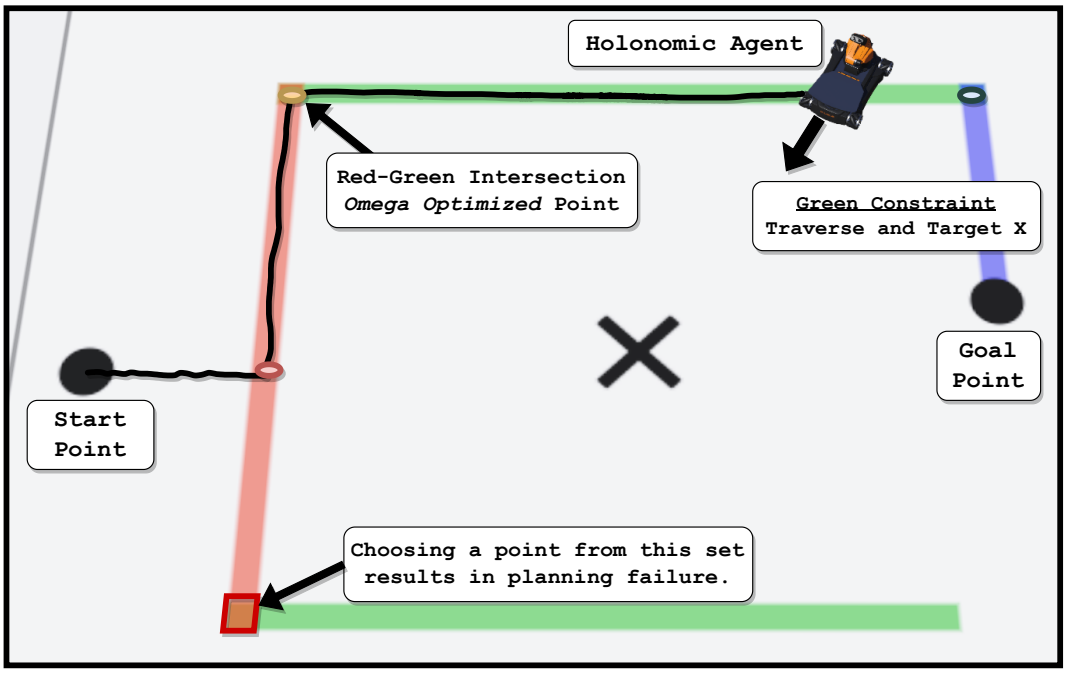


Figure 4.10: Experiment environment for Domain II. A holonomic agent (Kuka YouBot using WeBots simulation environment) must move from the start point, onto the red line-tracking constraint, onto the green line-tracking-and-targeting constraint, and then onto the blue line-tracking constraint. The red-green intersection Ω-set is the upper left red-green constraint intersection as those points are the only red-green constraint intersection points that enable the agent to successfully reach the blue constraint and ultimately the goal point.

4.4.2.1 Omega Optimization: Non-linear Mixed Integer Multiobjective Program

In order to select the correct Ω-set point, the Omega Optimization used in the IPD-Relaxation algorithm for this domain is a mixed-integer non-linear multiobjective program solved using the GEKKO solver [16]. While the constraints in this particular domain (including those with disjoint sets) can be explicitly solved for, choosing the optimal disjoint set depends on the set's proximity to a candidate point and we use a general optimization approach to maintain consistentcy with Ω-point selection for more complicated implicit constraint manifold domains. The objective function for this domain is outlined in Equation 4.5:

$$
\begin{aligned}
\min_{q=(q_x,q_y,q_\theta)} \quad & f(q) = w_1 * A * distIntersectionOne(q) \\
& + w_2 * (1 - A) * distIntersectionTwo(q) \\
& + w_3 * distToKeyframePoint(q, q_{kf}) \\
s.t. \quad & \\
& 1) \quad withinLimits(q) \\
& 2) \quad A * distIntersectionOne(q) \leq \epsilon \\
& 3) \quad (1 - A) * distIntersectionTwo(q) \leq \epsilon \\
& 4) \quad A \in \{0, 1\} \\
& 5) \quad q_\theta = 360 - arctan(\frac{t_y - q_y}{t_x - q_x}) * 180/\pi
\end{aligned}
\tag{4.5}
$$

The binary integer component A (constraint 4) forces the optimizer to determine the intersection choice that best minimizes the cost function. The point q is initialized as q_{kf}, the candidate value that is generated according to the different conditions outlined in section 4.3.3. In conditions in which keyframe points are not used, w_3 is set to zero to effectively remove the keyframe-associated terms from the cost function. Collision objects are not used in this setting as any collision object that occludes any of the line constraints renders the planning problem infeasible. Constraints 2 and 3 dictate that the value chosen for the intersection point is within some ϵ tolerance. Constraint 5 ensures that the chosen value for the angle of the agent is the smallest value.

4.4.3 Evaluation Domain 3: Simulated Manipulator Arm with Implicit Manifold Constraints

In this evaluation domain, five human users provided at least three kinesthetic demonstrations of a pouring task (see Figure 4.11) on a ReThink Robotics Sawyer 7-DOF manipulator in order to generate a CC-LfD model for each. The task utilized three constraints: 1) an orientation constraint that requires the manipulator arm to hold a liquid vessel in its upright position to avoid spilling; 2) a height-restricting constraint where the end-effector must maintain a certain distance

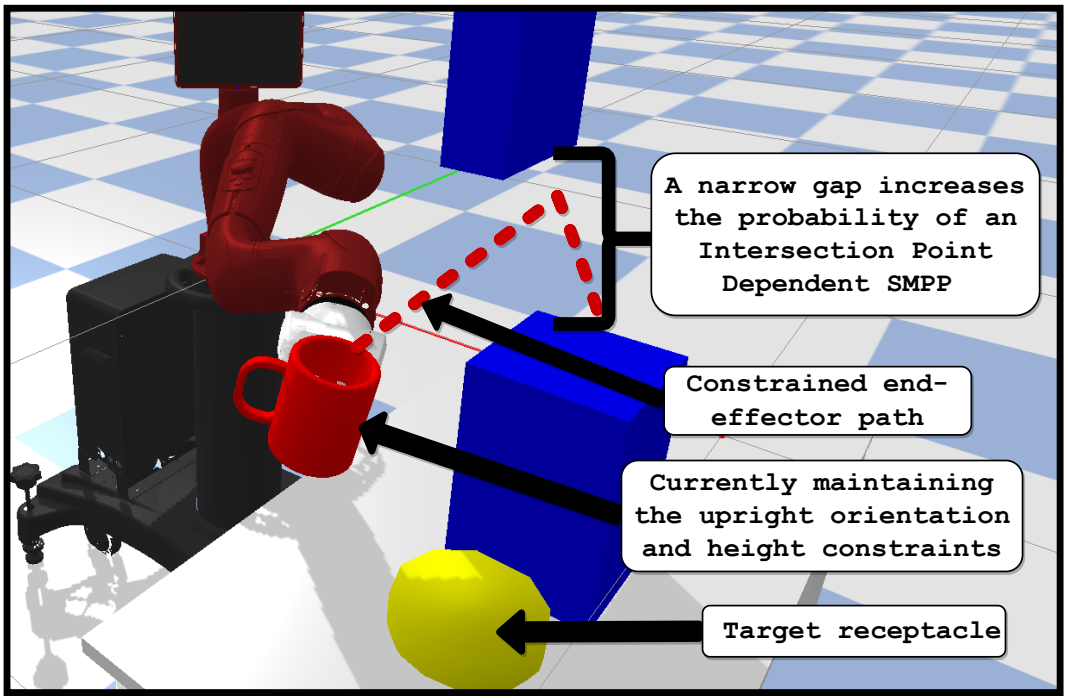


Figure 4.11: Evaluation Domain III: Simulated robot manipulator executing a pouring task in which three constraints must be adhered to in a sequential overlapping manner indicating a Sequential Manifold Planning Problem (SMPP). The manipulator must observe an upright orientation until pouring, maintain a height above the lower collision object, and remain centered over the target once pouring.

above the workbench; 3) a positional constraint requiring the end-effector to remain centered over a receptacle. While it's not possible to explicitly verify whether this task is an SMPP with intersection point dependency, the design has a permanent collision object close to the receptacle that generally configures the end effector wrist at a joint limit, creating (with high probability) a disjoint manifold intersection between the first and third constraint. It also utilizes an additional collision object (upper blue collision object in Figure 4.11) to induce a narrow gap that the arm must pass through to complete the task.

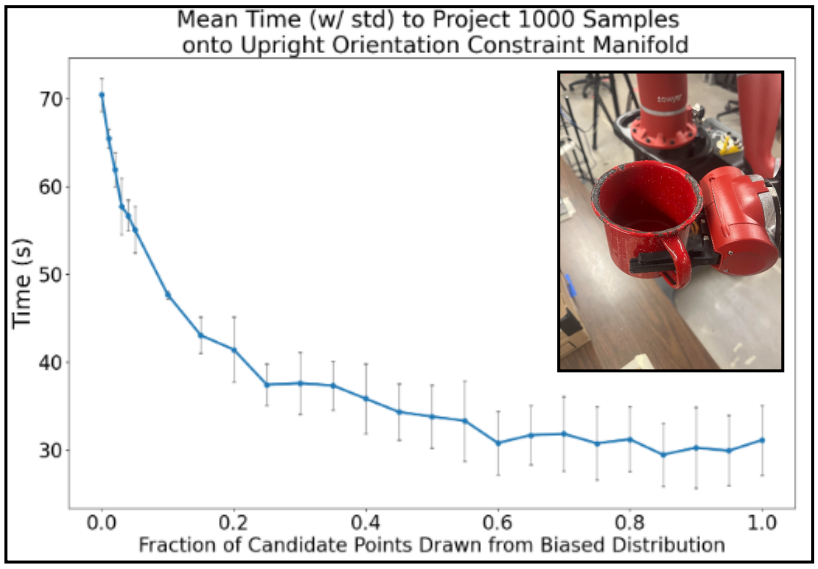
4.4.3.1 Omega Optimization: Non-linear Multiobjective Program

As this domain utilizes implicit manifold constraints, a more sophisticated optimization approach is used for Omega Optimization. This approach borrows from [118], utilizing the PANOC optimization library for the Rust programming language [134]. In Equation 4.6 the cost function

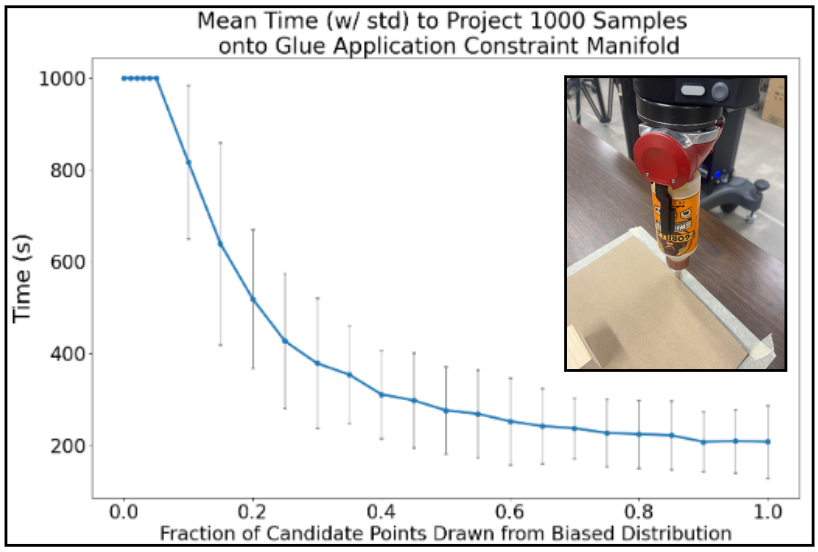
terms are each contained within a 'groove loss' function ('GL()') that combines linear and Gaussian terms with the effect of generating much lower costs near the optimal value for the wrapped function (see [118] for more details about this function and the standard parameters adopted by this paper). This choice enables the easy integration of multiple terms into a multi-objective cost function and enables the use of finite-differencing methods for generating approximate gradients within the optimizer.

$$\begin{aligned} \min_{q} \quad f(q) &= w_1 * GL(DistTSRPosition(q)) \\ &+ w_2 * GL(DistTSRQuat(q)) \\ &+ w_3 * GL(DistKFPosition(q, q_{kf})) \\ &+ w_4 * GL(DistKFQuat(q, q_{kf})) \end{aligned} \tag{4.6}$$

The $DistTSRPosition(q)$ and $DistTSRQuat(q)$ terms utilize the distance conventions of the Task Space Regions (TSRs) constraint framework introduced by [17] (see Figure 4.6). This convention is used to easily define XYZ-position and Roll-Pitch-Yaw orientation constraints i.e. task space constraints. We implement the distance for position and orientation separately to weigh the terms individually in the cost function. $DistKFPosition(q, q_{kf})$ and $DistKFQuat(q, q_{kf})$ restrain the optimizer so that the converged value does not stray too far from the candidate sample provided by the constraint transition keyframe distribution. These terms drive value in using keyframe distributions as a heuristic for generating Ω-set compliant points. The assumption is that finding an intersection point much closer to the distribution makes it much likelier to be in the Ω-set. For experiment conditions that perform optimization but do not utilize keyframe data, weights w_3 and w_4 are set to zero.



(a) Domain I - Orientation (Upright) Constraint: Users provide task-independent demonstrations of the cup in an upright orientation (see inset image). Even a small amount of bias (20%) nearly halves required sampling and projection time.



(b) Domain I - Glue Application Constraint: Users provide task-independent demonstrations of the robot agent holding the glue bottle in the correct orientation and height (see inset image). As this constraint is more restrictive, sampling times are longer (capped at 1000 seconds).

Figure 4.12: Results for Domain I show the mean time (s) with σ (gray) against fractional-biased sampling for sampling 1000 constraint-compliant points. Candidate samples are drawn either uniformly or from a biasing distribution to seed the Jacobian projection operator in order to produce constraint-compliant points. The greater the fraction of points that come from learned distributions, the better the time efficiency of the sampling of constraint-compliant points.

4.5 Results and Discussion

4.5.1 Domain I Results - Biased Sampling

As seen in Figures 4.12a and 4.12b, the utilization of distributions learned from user demonstrations that adhere (even if approximately) to the desired constraints can *substantially decrease* the sampling time needed to produce 1000 constraint-compliant points. This substantial drop in sampling time exists even when 20% of the candidate points are drawn from a biased distribution. As the distributions produce candidate points that are close to the implicit constraint manifold, less computational effort is needed to push the candidate point onto the manifold surface iteratively. Many constrained motion planning algorithms [17, 135, 113, 114, 64, 136] that use the Jacobian projection to generate constrained points could benefit from using human-provided demonstrations to bias sampling, realizing benefits that may persist even across dissimilar tasks or environments. Likewise, biased sampling may be useful for planning problems requiring many constraint-compliant points for training [113, 114] or for roadmap-based techniques that need broad coverage of the planning space [76].

4.5.2 Domain II Results - 2D Navigation with Explicit Constraints

For Domain II, Table 4.2 indicates that the Omega Optimization (OO) intersection point generation mechanism results in sequential manifold planning that performs best in most metrics given the explicit constraints and intersection point dependency. However, there are some metrics that show little difference given the simplicity of the domain's environment.

Sampling Bias Condition: For this domain, sampling bias shows that utilizing intermediate keyframe distributions decreases planning time for successful planning events.

Planning Success Metric: Planning Success in Domain 2 relies on keyframe candidate points (KF & OO conditions). The CO conditions often choose the wrong intersection point, resulting in planning failure ($\leq$ 30% for CO conditions). Given the relaxation of off-manifold points (see section 4.3.4), KF conditions see an inflated percentage given the simplicity of the domain and the

Table 4.2: Metrics across 50 trials per 5 subjects for Domain 2.
Bold: best within intersection point gen. group (gray/white)
(KF=Keyframe-only, CO=Constraint-only Optimization, OO=Omega Optimization)
**Only includes successful trials.

Conditions		Metrics				
Sampling	Intersection Pt. Gen.	Success %	PL (pixels)**	A2S (DTW)**	A2F (%)**	PT (s)**
Biased	KF	**100**	13758.11 ± 1602.04	21957.74 ± 5643.12	91.0 ± 0.01	**0.58 ± 1.81**
	CO	18.4	16223.32 ± 3928.009	50390.76 ± 34197.85	**95.0 ± 0.02**	2.05 ± 2.48
	OO	**100**	**13400.16 ± 1629.37**	**18242.04 ± 3855.13**	93.3 ± 0.01	0.68 ± 0.09
Uniform	KF	**100**	14387.65 ± 1481.11	21938.00 ± 5254.53	94.2 ± 0.942	**0.59 ± 1.40**
	CO	30	16682.55 ± 4107.92	82589.71 ± 51390.81	96.0 ± 0.01	5.67 ± 8.40
	OO	**100**	**13693.93 ± 1472.88**	**19794.15 ± 4003.43**	**96.2 ± 0.00**	0.92 ± 1.84

fact that demonstrators often provided fairly optimal demonstrations. Keyframe distributions produced intersection points in the KF conditions that were fairly close to being manifold-intersection constraint-compliant.

Path Length Metric: Across all conditions, Path Length values are highest (worst) in the CO condition because this generation mechanism produces constraint-compliant points but does not follow the style of the demonstrated skill, often resulting in long segments between intersection points. The OO condition produced the lowest (best) path length across all conditions.

Adherence to Style Metric: Adherence to Style (A2S) for CO results in very high (poor) values as intersection points are not coupled to demonstration data. For the OO condition, A2S is the lowest followed closely by the KF condition. Given the relaxed off-manifold-intersection tolerance, KF condition utilizes intersection points that have greater variance than the OO condition, and thus the KF condition sees slightly worse A2S performance.

Adherence to Function Metric: For successful events, during biased sampling the CO intersection point generation mechanism produced a slightly higher adherence to function (A2F) percentage. When successful planning under the CO condition occurred (which was rare), the biasing enables generated points during planning to produce on-manifold points. However, given the simplicity of this evaluation domain, the A2F values were all quite high ($\geq 91\%$). Likewise, given the simplicity of the domain, interpolation between points can produce segments of the final solution trajectory during constrained planning that are slightly off-manifold, hence why none of

the conditions achieved exactly 100% A2F.

Planning Time Metric: Planning times for the KF and OO conditions are fairly equivalent given how simple the environment is for producing points. However, the CO method produced significantly longer planning times. As the CO method does not rely on the keyframe distributions for seed points, the optimization often took much longer to converge during successful planning events.

4.5.3 Domain III Results - Simulated Manipulator Arm with Implicit Manifold Constraints

Table 4.3 indicates that the Optimization (OO) intersection point generation mechanism results in sequential manifold planning that is empirically more performant for Domain III, as evidenced by the generally best-performing metrics across all conditions, especially planning success percentage. Planning failures occur much more often when induced by environmental conditions (Domain III) when not using the proposed OO intersection point generation mechanism.

Table 4.3: Metrics across 25 trials per 5 participants for Domain III.
Bold: best within intersection point gen. group (gray/white)
(KF=Keyframe-only, CO=Constraint-only Optimization, OO=Omega Optimization)
**Only includes successful trials.

Conditions			Metrics				
Sampling	Collision Obj.	Intersection Pt. Gen.	Success %	PL (m)**	A2S (DTW)**	A2F (%)**	PT (s)**
Biased	Yes	KF	2.4	12.51 ± 1.22	1488.36 ± 644.43	69.65 ± 14.01	**43.49 ± 3.18**
		CO	20.8	26.38 ± 5.87	8961.26 ± 3689.74	99.76 ± .58	49.93 ± 7.08
		OO	**96.0**	**10.06 ± 2.26**	**1218.68 ± 395.34**	**99.99 ± .10**	44.18 ± 8.07
	No	KF	11.2	11.57 ± 1.57	1285.38 ± 442.55	75.09 ± 12.90	**41.46 ± 1.28**
		CO	27.2	24.22 ± 4.54	7434.85 ± 1890.90	99.89 ± 0.38	48.51 ± 7.52
		OO	**100**	**9.99 ± 2.09**	**1188.09 ± 362.50**	**99.94 ± .30**	42.79 ± 6.22
Uniform	Yes	KF	.8	21.15 ± .00	2686.66 ± 0.00	82.28 ± 0.00	94.57 ± 0.00
		CO	4.0	63.10 ± 7.28	21921.38 ± 3935.41	99.77 ± .36	86.69 ± 11.87
		OO	**88.0**	**12.23 ± 3.65**	**1474.80 ± 471.43**	**99.98 ± .12**	**50.50 ± 8.20**
	No	KF	.8	18.01 ± 4.65	2798.93 ± 1457.33	63.27 ± 14.08	75.75 ± 18.2
		CO	4.8	58.84 ± 11.86	18547.05 ± 8668.48	99.77 ± .34	92.30 ± 25.62
		OO	**99.2**	**10.50 ± 2.62**	**1247.61 ± 433.55**	**99.98 ± .14**	**45.40 ± 8.48**

Sampling Bias Condition: As indicated in Table 4.3 sampling bias for Domain III again shows that intermediate trajectory distribution utilization decreases planning time for successful planning events. In Domain III, biased sampling has a normalizing effect on planning time for successful

planning events across all other conditions. This shows the demonstration data itself is useful for more efficient planning, especially for constrained motion planning, assuming approximately compliant demonstrations.

Planning Success Metric: Planning success percentages in Domain III display the biggest differential amongst intersection point generation conditions, with OO generally resulting in $\geq$ 88% planning success whereas all other intersection point generation methods never surpass 27.2%. The inclusion of a collision object that creates a narrow feasible gap reduces the planning success across all conditions but reveals the resiliency of OO in maintaining high planning success rates.

Path Length Metric: Path Length values are highest (poor) in the CO condition. The CO condition produced constraint-compliant points but does not follow the style of the demonstrated skill, often resulting in long segments between intersection points. Conditions utilizing keyframe-derived candidate points significantly reduced path lengths, whereas the OO condition slightly reduced path length relative to the KF conditions. Given that OO produces the optimal choice for intersection points, the resultant path lengths are slightly shorter than the KF condition which might produce slightly off-manifold-intersection points.

Adherence to Style Metric: Adherence to Style (A2S) for CO results in very high (poor) values as intersection points are not coupled to demonstration data. In Domain II, the inclusion of a collision object that creates a narrow gap the arm must traverse through results in the biggest differentials in A2S. As keyframe points are derived from demonstration data, it follows that A2S for CO conditions is much worse as the intersection points no longer are coupled to the stylistic intent of human demonstrators, often resulting in unusual solution trajectories despite maintaining constraint compliance.

Adherence to Function Metric: In Domain II, Adherence to Function (A2F) is $\geq$99% across all conditions for CO and OO intersection point generation methods those intersection point generation methods likely produced Ω-points for *successful* planning events. However, the success percentage for CO condition is substantially lower, with 27.2% as the best success percentage and the lowest 4.0% despite 99.89 and 99.94% adherence to function percentages, respectively. As such,

all conditions cannot produce constraint-compliant points perfectly, hence all conditions see <100% adherence to function values.

Planning Time Metric: As mentioned in section 4.5.1, planning time is generally much lower when using sampling bias. And across all conditions, OO resulted in substantially lower planning times, whereas in cases where KF out-competed other approaches, the success percentages are quite low (2.4% and 11.2%).

Chapter 5

Conclusion

This dissertation presents novel contributions that touch on all three major components of robot learning from demonstration (see Figure 5.1). The first is Concept Constrained Learning from Demonstration (CC-LfD), an algorithm that serves as the nucleus of the contributions herein. It extends an established Learning from Demonstration technique by incorporating additional context in the form of task space constraints. This algorithm motivated the development of an augmented reality ecosystem for user-friendly teleoperation-based demonstration, skill editing, and evaluation. This ecosystem is realized by two major contributions, a novel interactive augmented reality system called Augmented Reality for Constrained Learning from Demonstration (ARC-LfD), and an in-situ visualization engine for rendering robot holographic to aid in the self-correction and quality of demonstration using an instrumented tong device called Augmented Reality-based Pose Optimization for Constrained Learning from Demonstration (ARPOC-LfD). Lastly, this dissertation describes an algorithm to more efficiently generate execution trajectories to sequential manifold planning problems that are defined by the CC-LfD model. This algorithm is called Intersection Point Dependency Relaxation (IPD-Relaxation). This planning algorithm is the key essential piece that unlocks Constraint-based Learning from Demonstration by providing a mechanism for solving the challenging Sequential Manifold Planning problems that such models can produce. This opens the door for robot agents to execute constraint-varying behavior as defined by users in a human-robot collaborative environment.

5.1 Summary of Contributions

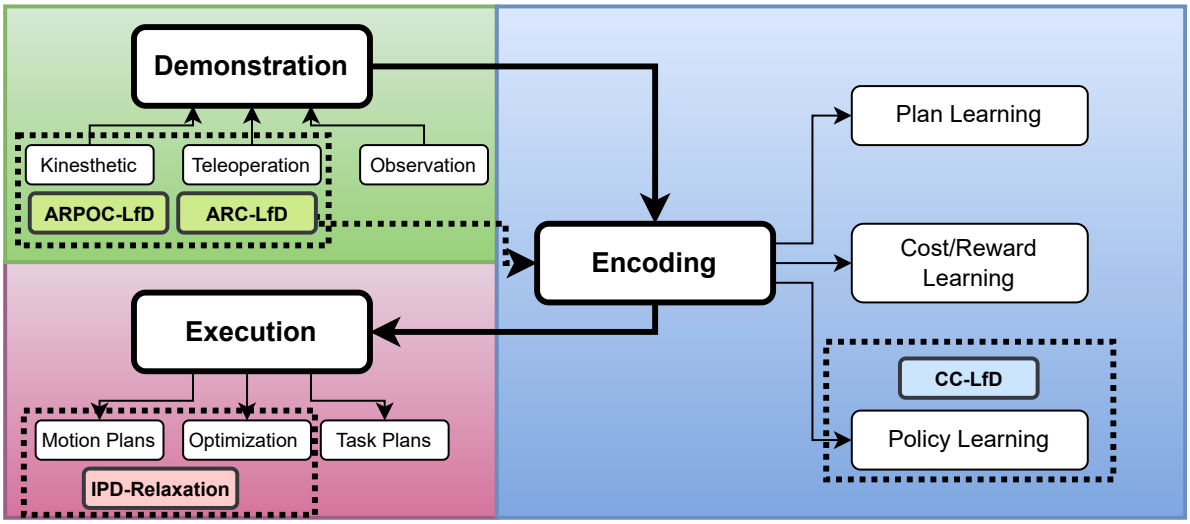


Figure 5.1: Revisiting the contributions of this dissertation within the framework of the Learning From Demonstration pipeline.

5.1.1 Concept Constrained Learning from Demonstration (CC-LfD)

Chapter 2 describes the details of how CC-LfD enables learning of policies demonstrated through demonstration trajectories while allowing skill repair through minimal additional demonstrations. Results show that a single well-performed constrained demonstration dramatically repairs poor skill performance. This improvement is remarkable when compared with traditional approaches to skill repair whereby perfect demonstrations are introduced in an attempt to correct existing data or models that are compromised. CC-LfD does not require the removal of existing corrupted demonstrations for successful skill repair. This is significant because corrupted demonstrations have viable skill information. Introducing conceptual constraints decouples the corrupting features from this viable information. Without these constraints, a single poor demonstration can prevent successful skill reconstruction. The unconstrained statistical keyframe distributions expose the possibility of sampling a corrupted point whereas CC-LfD parses the distribution in such a way that all sampled points satisfy the concept constraints. By relearning from this parsed data,

CC-LfD effectively shifts the representation of keyframes towards constraint compliance to produce a better representation overall.

The CC-LfD model also provides empirical evidence for skill transfer by CC-LfD, further confirmed by the ARC-LfD interface. The redefining of objective behavior with a conceptual constraint shows that corrected behavior is achievable after the introduction of one well-performed constraint. The new model shows an ability to overcome the bias of prior demonstrations beholden to the old constraint definition. This preliminary evidence suggests that concept constraints play a role in skill transfer learning, but more research is desired.

5.1.2 Augmented Reality Systems for Constrained Robot Learning from Demonstration

Chapter 3 describes two systems that expand upon the interaction capabilities of human-robot collaboration involving constrained learning from demonstration. ARC-LfD is proposed as a step toward producing practical, real-world-ready LfD systems that allow non-roboticists to conduct training and evaluation of robotic systems. The use of AR for in-situ visualizations relaxes the requirement of a model of the environment to use in simulation for verification of learned skills. Through visualizing a sample trajectory directly in the environment, users can preview the robot's skill execution contextualized by the actual environment itself.

The control flow of ARC-LfD provides an improvement over CC-LfD, allowing users to separate demonstration from constraint application. Rather than requiring users to specify constraints during live demonstrations, users are able to visualize and edit constraints at the verification step. This allows focusing on ensuring constraints are both appropriately specified and applied to the correct keyframes before application. Finally, the proposed constraint editing interface relaxes the static environment assumption often levied for successful LfD skill deployment. ARC-LfD enables direct skill repair and editing, creating constraints contextualized in the environment and applying them to keyframes of an existing skill. Thus, ARC-LfD fills a critical technical gap in LfD systems, enabling long-term skill assessment and validation as the environment or task requirements change

over time.

The other AR system, ARPOC-LfD, combines elements from online pose optimization and constrained demonstration to facilitate easier and more user-friendly forms of demonstration. The ability to provide accurate demonstrations of both the stylistic intent of the skill as well as valid examples of constrained motion both assist in the generation/encoding of the CC-LfD model. IPD-Relaxation also depends on a quality keyframe model whose distributions aid in the discovery of the correct constraint manifold overlap. By enabling users more easily provide demonstration data through an easy-to-operate device, supported by systems that help users demonstrate potentially challenging difficult constrained behavior, this helps facilitate better-constrained model representations and aides the IPD-Relaxation algorithm in choosing the correct constraint manifold overlap regions for successful planning in a Sequential Manifold Planning problem.

5.1.3 Intersection Point Dependency Relaxation

Chapter 4 demonstrates how the CC-LfD model defines a Sequential Manifold Planning problem (SMPP), and then outlines how the distributions representing constraint transition boundaries aid in relaxing an Intersection Point Dependent SMPP into an Intersection Point Independent SMPP. This is facilitated by an optimization process called Intersection Point Dependency Relaxation or IPD-Relaxation. It utilizes a problem-specific Omega Optimization that greatly improves the efficiency in choosing planning-success enabling intersection points for SMPP problems. Results presented in sections 4.5.2 and 4.5.3 show that this Omega Optimization process more often produces the correct choice of intersection points when seeded with constraint transition keyframe points. This demonstrates the utility of using points derived from the CC-LfD model in order to solve Intersection Point Dependent SMPPs efficiently. These important results show how constrained motion planning can be used by constraint Learning from Demonstration models that otherwise struggled to efficiently produce solution trajectories.

5.2 Implications for Future Work

In its entirety, this dissertation presents a suite of algorithms and systems that enable the integration of task-pertinent context into the Robot Learning from Demonstration pipeline. However, there are many avenues where further research is warranted.

CC-LfD introduces a novel mechanism for constraint integration into the encoding portion of the LfD pipeline (see Chapter 2 and Figure 5.1). There are unexplored problem-areas that warrant investigation as it relates to the integration of constraint-based context. For example, the automatic keyframing mechanism of trajectory data may not correctly capture the intent of the user, which could place too much emphasis on regions of the trajectory that have little importance on the intended skill. Similarly, the culling process could remove keyframes with high information about the behavioral intent of the demonstration. This may result in a poorly informed model that is prone to error should users care heavily about the stylistic performance of the robot agent.

As it stands, CC-LfD models use single-chain (i.e. non-branching) directed graph representations of skills. If user demonstrations are highly variant, perhaps because a user is demonstrating multiple alternative ways to complete a skill, the CC-LfD model tries to encompass all these demonstrations into a single chain of sequential pose distributions. A model that relies more on a skill tree representation with dedicated skill-variant paths through this tree-like representation would provide a much broader representation of the task. Constraint transition keyframes would still exist in these skill-variant paths. However, the alignment of skill-variants and skill-variant path jumping are open problems. How to represent such models in AR and for use in IPD-Relaxation is also an open problem.

This dissertation did not focus on the role that the statistical model choice plays in skill repair. Perhaps models that reshape their distribution to bias new unconstrained demonstrations will result in more effective skill repair than that of the CC-LfD evaluations. Furthermore, constrained demonstration might not always be feasible or desirable depending on the setting. Perhaps a user does not quite know what constraints are important relative to a task.

While ARC-LfD and ARPOC-LfD provide alternative mechanisms for applying constraints and providing demonstration, users might not always have a prior understanding of what constraints are needed for a specific skill. A system that attempts to infer constraints and skill style from previously learned skills could help users better understand how to shape a skill they are teaching a robot. The AR systems presented in this paper could provide recommendation interfaces that help explain what prior knowledge the robot system possesses and how this prior knowledge might be helpful in constraining or demonstrating a new skill the user would like the robot to learn. This form of lifelong experiential learning and continuous human-in-the-loop model update could result in a much more sophisticated learning system. As it stands now, CC-LfD, ARC-LfD, and ARPOC-LfD expect skill expertise from human collaborators, a requirement that might limit their application feasibility.

While the IPD-Relaxation opens up constraint Learning from Demonstration to constrained motion planning techniques, there could exist certain planning problems that undermine its ability to perform Intersection Point Dependence Relaxation. For example, experiment conditions that shift the ρ-useful set away from the learned distributions of the CC-LfD model might negatively impact planning performance, as biased sampling would more often produce off-manifold points. Future work that could better account for such shifts would make IPD-Relaxation generalize to noisier LfD models. In a similar vein, adversarial demonstrations that result in multimodal keyframe distributions could be detrimental to this algorithm as adjacent intersection points in the planning graph might jump across different modes that result in infeasible planning. Future work might consider how to generate robot LfD models that can self-differentiate between internal variance in demonstrations.

A major area of future research lies within constraint type and representation. The task space constraints in this dissertation are represented either as Boolean classifiers for rejection sampling schemes or as Task Space Regions [17] (see Section 4.1.2.5). The systems and algorithms in this dissertation rely on constraints that ignore time-dependent higher-order constraints like velocity, acceleration, jerk, and torques. Expanding upon the diversity of usable constraints for the com-

ponents of this dissertation would drastically expand the applicability of the contributions of this dissertation. However, from a topological perspective, such constraints complicate the state representation even further given that the integration of time indicates a potentially changing surface topology of constraint manifolds with respect to time. Further fusing areas of geometric control with constrained motion planning might unlock IPD-Relaxation and constrained LfD models for more complicated constraint types.

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning, page 1, 2004.

[2] Baris Akguen, Kaushik Subramanian, and Andrea Lockerd Thomaz. Novel interaction strategies for learning from teleoperation. In AAAI Fall Symposium: Robots Learning Interactively from Human Teachers, volume 12, page 07, 2012.

[3] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L Thomaz. Keyframe-based learning from demonstration. International Journal of Social Robotics, 4(4):343–355, 2012.

[4] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, pages 391–398, 2012.

[5] Baris Akgun and Mike Stilman. Sampling heuristics for optimal motion planning in high dimensions. In 2011 IEEE/RSJ international conference on intelligent robots and systems, pages 2640–2645. IEEE, 2011.

[6] Baris Akgun, Kaushik Subramanian, and Andrea Lockerd Thomaz. Novel interaction strategies for learning from teleoperation. 2012 AAAI Fall Symposium: Robots Learning Interactively from Human Teachers, 12, 2012.

[7] Baris Akgun and Andrea Thomaz. Simultaneously learning actions and goals from demonstration. Autonomous Robots, 40(2):211–227, 2016.

[8] Heni Ben Amor, Gerhard Neumann, Sanket Kamthe, Oliver Kroemer, and Jan Peters. Interaction primitives for human-robot cooperation tasks. In 2014 IEEE international conference on robotics and automation (ICRA), pages 2831–2837. IEEE, 2014.

[9] Stephanie Arevalo Arboleda, Franziska Rücker, Tim Dierks, and Jens Gerken. Assisting manipulation and grasping in robot teleoperation with augmented reality visual cues. In Proceedings of the 2021 CHI conference on human factors in computing systems, pages 1–14, 2021.

[10] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. Robotics and Autonomous Systems, 57(5):469–483, 2009.

[11] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In ICML, volume 97, pages 12–20. Citeseer, 1997.

[12] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. Learning robot objectives from physical human interaction. In Conference on Robot Learning, pages 217–226. PMLR, 2017.

[13] Paul Bakker, Yasuo Kuniyoshi, et al. Robot see, robot do: An overview of robot imitation. In AISB96 Workshop on Learning in Robots and Animals, volume 5. Citeseer, 1996.

[14] Zahraa Bassyouni and Imad H Elhajj. Augmented reality meets artificial intelligence in robotics: A systematic review. Frontiers in Robotics and AI, page 296, 2021.

[15] Chandrayee Basu, Mukesh Singhal, and Anca D Dragan. Learning from richer human guidance: Augmenting comparison-based learning with feature queries. 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2018.

[16] Logan Beal, Daniel Hill, R Martin, and John Hedengren. Gekko optimization suite. Processes, 6(8):106, 2018.

[17] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. The International Journal of Robotics Research, 30(12):1435–1460, 2011.

[18] Dmitry Berenson and Siddhartha S Srinivasaz. Probabilistically complete planning with end-effector pose constraints. In 2010 IEEE International Conference on Robotics and Automation, pages 2724–2730. IEEE, 2010.

[19] Joshua Bialkowski, Michael Otte, and Emilio Frazzoli. Free-configuration biased sampling for motion planning. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1272–1279. IEEE, 2013.

[20] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. Technical report, Springrer, 2008.

[21] Valérie Boor, Mark H Overmars, and A Frank Van Der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), volume 2, pages 1018–1023. IEEE, 1999.

[22] Cynthia Breazeal and Brian Scassellati. Robots that imitate humans. Trends in Cognitive Sciences, 6(11):481–487, 2002.

[23] John Brooke et al. Sus-a quick and dirty usability scale. Usability evaluation in industry, 189(194):4–7, 1996.

[24] Connor Brooks and Daniel Szafir. Visualization of intended assistance for acceptance of shared control. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[25] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[26] Riccardo Caccavale, Matteo Saveriano, Alberto Finzi, and Dongheui Lee. Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction. Autonomous Robots, 43(6):1291–1307, 2019.

[27] Maya Cakmak and Andrea L Thomaz. Designing robot learners that ask good questions. In Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, pages 17–24. ACM, 2012.

[28] Sylvain Calinon, Paul Evrard, Elena Gribovskaya, Aude Billard, and Abderrahmane Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In 2009 International Conference on Advanced Robotics, pages 1–6. IEEE, 2009.

[29] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 37(2):286–298, 2007.

[30] Constantinos Chamzas, Anshumali Shrivastava, and Lydia E Kavraki. Using local experiences for global motion planning. In 2019 International Conference on Robotics and Automation (ICRA), pages 8606–8612. IEEE, 2019.

[31] Crystal Chao, Maya Cakmak, and Andrea L Thomaz. Towards grounding concepts for transfer in goal learning from demonstration. In Development and Learning (ICDL), 2011 IEEE International Conference on, volume 2, pages 1–6. IEEE, 2011.

[32] Sonia Chernova and Andrea L Thomaz. Robot learning from human teachers. Synthesis Lectures on Artificial Intelligence and Machine Learning, 8(3):1–121, 2014.

[33] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. IEEE Robotics & Automation Magazine, 19(1):18–19, 2012.

[34] Paul J Choi, Rod J Oskouian, and R Shane Tubbs. Telesurgery: past, present, and future. Cureus, 10(5), 2018.

[35] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022.

[36] Adam Coates, Pieter Abbeel, and Andrew Y Ng. Learning for control from multiple demonstrations. In Proceedings of the 25th international conference on Machine learning, pages 144–151. ACM, 2008.

[37] Ashwin Dani et al. Learning position and orientation dynamics from demonstrations via contraction analysis. Autonomous Robots, 43(4):897–912, 2019.

[38] Neil T Dantam, Swarat Chaudhuri, and Lydia E Kavraki. The task-motion kit: An open source, general-purpose task and motion-planning framework. IEEE Robotics & Automation Magazine, 25(3):61–70, 2018.

[39] Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. Incremental task and motion planning: A constraint-based approach. In Robotics: Science and systems, volume 12, page 00052. Ann Arbor, MI, USA, 2016.

[40] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Methodological), 39(1):1–22, 1977.

[41] Miha Deniša, Andrej Gams, Aleš Ude, and Tadej Petrič. Learning compliant movement primitives through demonstration and statistical generalization. IEEE/ASME transactions on mechatronics, 21(5):2581–2594, 2015.

[42] Maximilian Diehl, Alexander Plopski, Hirokazu Kato, and Karinne Ramirez-Amaro. Augmented reality interface to verify robot learning. In 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 378–383. IEEE, 2020.

[43] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: a task-level planning approach. International Journal of Advanced Robotic Systems, 5(3):33, 2008.

[44] Peter Englert, Isabel M Rayas Fernández, Ragesh K Ramachandran, and Gaurav S Sukhatme. Sampling-based motion planning on sequenced manifolds. arXiv preprint arXiv:2006.02027, 2020.

[45] Bin Fang, Di Guo, Fuchun Sun, Huaping Liu, and Yupei Wu. A robotic hand-arm teleoperation system using human arm/hand with a novel data glove. In 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 2483–2488. IEEE, 2015.

[46] HC Fang, SK Ong, and AYC Nee. Orientation planning of robot end-effector using augmented reality. The International Journal of Advanced Manufacturing Technology, 67(9):2033–2049, 2013.

[47] HC Fang, Soh-Khim Ong, and Andrew YC Nee. Novel ar-based interface for human-robot interaction and visualization. Advances in Manufacturing, 2(4):275–288, 2014.

[48] Salli Forbes, Mary Ann Poparad, and Maryann McBride. To err is human; to self-correct is to learn. The reading teacher, 57(6):566–572, 2004.

[49] Michael Freedberg, Brian Glass, J Vincent Filoteo, Eliot Hazeltine, and W Todd Maddox. Comparing the effects of positive and negative feedback in information-integration category learning. Memory & cognition, 45(1):12–25, 2017.

[50] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. Annual review of control, robotics, and autonomous systems, 4:265–293, 2021.

[51] Gal Gorjup, George P Kontoudis, Anany Dwivedi, Geng Gao, Saori Matsunaga, Toshisada Mariyama, Bruce MacDonald, and Minas Liarokapis. Combining programming by demonstration with path optimization and local replanning to facilitate the execution of assembly

tasks. In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 1885–1892. IEEE, 2020.

[52] Scott A Green, Mark Billinghurst, XiaoQi Chen, and J Geoffrey Chase. Human-robot collaboration: A literature review and augmented reality approach in design. International Journal of Advanced Robotic Systems, 5(1):1, 2008.

[53] Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In 2011 IEEE international conference on robotics and automation, pages 3804–3809. IEEE, 2011.

[54] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Advances in psychology, volume 52, pages 139–183. Elsevier, 1988.

[55] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. The International Journal of Robotics Research, 30(6):678–698, 2011.

[56] Ioannis Havoutis and Sylvain Calinon. Learning from demonstration for semi-autonomous teleoperation. Autonomous Robots, 43(3):713–726, 2019.

[57] Bradley Hayes and Brian Scassellati. Discovering task constraints through observation and active learning. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 4442–4449. IEEE, 2014.

[58] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 5469–5476. IEEE, 2016.

[59] Bradley Hayes and Julie A Shah. Interpretable models for fast activity recognition and anomaly explanation during collaborative robotics tasks. In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 6586–6593. IEEE, 2017.

[60] Gillian M Hayes and John Demiris. A robot controller using learning by imitation. University of Edinburgh, Department of Artificial Intelligence, 1994.

[61] Hooman Hedayati, Michael Walker, and Daniel Szafir. Improving collocated robot teleoperation with augmented reality. In 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 78–86, 2018.

[62] Peter F Hokayem and Mark W Spong. Bilateral teleoperation: An historical survey. Automatica, 42(12):2035–2057, 2006.

[63] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 7087–7094. IEEE, 2018.

[64] Léonard Jaillet and Josep M Porta. Path planning with loop closure constraints using an atlas-based rrt. In Robotics Research, pages 345–362. Springer, 2017.

[65] Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. Learning trajectory preferences for manipulators via iterative improvement. In Advances in neural information processing systems, pages 575–583, 2013.

[66] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. The International Journal of Robotics Research, 32(9-10):1194–1227, 2013.

[67] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. IEEE Transactions on Robotics and automation, 14(1):166–171, 1998.

[68] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation, 12(4):566–580, 1996.

[69] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. Knowledge and information systems, 7(3):358–386, 2005.

[70] Scott Kiesel, Ethan Burns, and Wheeler Ruml. Abstraction-guided sampling for motion planning. In SoCS, 2012.

[71] Beobkyoon Kim, Terry Taewoong Um, Chansu Suh, and Frank C Park. Tangent bundle rrt: A randomized algorithm for constrained motion planning. Robotica, 34(1):202–225, 2016.

[72] Jinkyu Kim, Inyoung Ko, and Frank C Park. Randomized path planning for tasks requiring the release and regrasp of objects. Advanced Robotics, 30(4):270–283, 2016.

[73] Zachary Kingston, Constantinos Chamzas, and Lydia E. Kavraki. Using experience to improve constrained planning on foliations for multi-modal problems. In IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2021.

[74] Zachary Kingston and Lydia E. Kavraki. Scaling multimodal planning: Using experience and informing discrete search. IEEE Transactions on Robotics, pages 1–19, 2022.

[75] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Exploring implicit spaces for constrained sampling-based planning. The International Journal of Robotics Research, 38(10-11):1151–1178, 2019.

[76] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Decoupling constraints from sampling-based planners. In Robotics Research, pages 913–928. Springer, 2020.

[77] Zachary Kingston, Andrew M Wells, Mark Moll, and Lydia E Kavraki. Informing multi-modal planning with synergistic discrete leads. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 3199–3205. IEEE, 2020.

[78] Michal Kleinbort, Kiril Solovey, Zakary Littlefield, Kostas E Bekris, and Dan Halperin. Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation. IEEE Robotics and Automation Letters, 4(2):x–xvi, 2018.

[79] Kazuhiko Kobayashi, Koichi Nishiwaki, Shinji Uchiyama, Hiroyuki Yamamoto, Satoshi Kagami, and Takeo Kanade. Overlay what humanoid robot perceives and thinks to the real-world by mixed reality system. In 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 275–276. IEEE, 2007.

[80] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. The International Journal of Robotics Research, 31(3):360–375, 2012.

[81] Tomáš Kot and Petr Novák. Utilization of the oculus rift hmd in mobile robot teleoperation. In Applied Mechanics and Materials, volume 555, pages 199–208. Trans Tech Publ, 2014.

[82] Dennis Krupke, Frank Steinicke, Paul Lubos, Yannick Jonetzko, Michael Görner, and Jianwei Zhang. Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–9. IEEE, 2018.

[83] Boris Lau, Christoph Sprunk, and Wolfram Burgard. Kinodynamic motion planning for mobile robots using splines. In 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2427–2433. IEEE, 2009.

[84] Steven M LaValle. Planning algorithms. Cambridge university press, 2006.

[85] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[86] Steven M LaValle, James J Kuffner, BR Donald, et al. Rapidly-exploring random trees: Progress and prospects. Algorithmic and computational robotics: new directions, 5:293–308, 2001.

[87] J.A.N. Lee and J.A.N. Lee. International Biographical Dictionary of Computer Pioneers. Fitzroy Dearborn, 1995.

[88] Michael Lewis, Katia Sycara, and Phillip Walker. The role of trust in human-robot interaction. In Foundations of trusted autonomy, pages 135–159. Springer, Cham, 2018.

[89] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33:9459–9474, 2020.

[90] Matthew B Luebbers, Connor Brooks, Carl L Mueller, Daniel Szafir, and Bradley Hayes. Arc-lfd: Using augmented reality for interactive long-term robot skill maintenance via constrained learning from demonstration. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 3794–3800. IEEE, 2021.

[91] Dana Mackenzie. Topologists and roboticists explore an'inchoate world', 2003.

[92] Murilo M Marinho, Bruno V Adorno, Kanako Harada, Kyoichi Deie, Anton Deguet, Peter Kazanzides, Russell H Taylor, and Mamoru Mitsuishi. A unified framework for the teleoperation of surgical robots in constrained workspaces. In 2019 international conference on robotics and automation (ICRA), pages 2721–2727. IEEE, 2019.

[93] John P McIntire, Paul R Havig, and Eric E Geiselman. Stereoscopic 3d displays and human performance: A comprehensive review. Displays, 35(1):18–26, 2014.

[94] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. ACM Computing Surveys (CSUR), 54(6):1–35, 2021.

[95] Janet Metcalfe. Learning from errors. Grantee Submission, 68:465–489, 2017.

[96] Joseph Mirabel and Florent Lamiraux. Manipulation planning: addressing the crossed foliation issue. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 4032–4037. IEEE, 2017.

[97] Alexandre Monferrer and David Bonyuet. Cooperative robot teleoperation through virtual reality interfaces. In Proceedings Sixth International Conference on Information Visualisation, pages 243–248. IEEE, 2002.

[98] Carl Mueller, Jeff Venicx, and Bradley Hayes. Robust robot learning from demonstration and skill repair using conceptual constraints. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6029–6036. IEEE, 2018.

[99] Daniel Müller, Carina Veil, Marc Seidel, and Oliver Sawodny. One-shot kinesthetic programming by demonstration for soft collaborative robots. Mechatronics, 70:102418, 2020.

[100] Chrystopher L Nehaniv, Kerstin Dautenhahn, et al. The correspondence problem. Imitation in animals and artifacts, 41, 2002.

[101] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. Learning and generalization of complex tasks from unstructured demonstrations. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5239–5246. IEEE, 2012.

[102] Samwel Opiyo, Jun Zhou, Emmy Mwangi, Wang Kai, and Idris Sunusi. A review on teleoperation of mobile ground robots: Architecture and situation awareness. International Journal of Control, Automation and Systems, 19(3):1384–1407, 2021.

[103] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. Advances in neural information processing systems, 26, 2013.

[104] Emanuel Parzen. On estimation of a probability density function and mode. The annals of mathematical statistics, 33(3):1065–1076, 1962.

[105] Carolina Passenberg, Angelika Peer, and Martin Buss. A survey of environment-, operator-, and task-adapted controllers for teleoperation systems. Mechatronics, 20(7):787–801, 2010.

[106] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In 2009 IEEE International Conference on Robotics and Automation, pages 763–768. IEEE, 2009.

[107] Peixi Peng, Junliang Xing, and Lili Cao. Hybrid learning for multi-agent cooperation with sub-optimal demonstrations. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pages 3037–3043, 2021.

[108] Claudia Pérez-D'Arpino and Julie A Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 4058–4065. IEEE, 2017.

[109] Emmanuel Pignat and Sylvain Calinon. Learning adaptive dressing assistance from human demonstration. Robotics and Autonomous Systems, 93:61–75, 2017.

[110] Pragathi Praveena, Guru Subramani, Bilge Mutlu, and Michael Gleicher. Characterizing input methods for human-to-robot demonstrations. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 344–353. IEEE, 2019.

[111] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan, 2009.

[112] Camilo Perez Quintero, Sarah Li, Matthew KXJ Pan, Wesley P Chan, HF Machiel Van der Loos, and Elizabeth Croft. Robot programming through augmented trajectories in augmented reality. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1838–1844. IEEE, 2018.

[113] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip. Neural manipulation planning on constraint manifolds. IEEE Robotics and Automation Letters, 5(4):6089–6096, 2020.

[114] Ahmed H Qureshi, Jiangeng Dong, Asfiya Baig, and Michael C Yip. Constrained motion planning networks x. IEEE Transactions on Robotics, 2021.

[115] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. A motion retargeting method for effective mimicry-based teleoperation of robot arms. In Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, pages 361–370, 2017.

[116] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. Relaxedik: Real-time synthesis of accurate and feasible robot arm motion. In Robotics: Science and Systems, pages 26–30. Pittsburgh, PA, 2018.

[117] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. Single-query path planning using sample-efficient probability informed trees. IEEE Robotics and Automation Letters, 6(3):4624–4631, 2021.

[118] Daniel Rakita, Haochen Shi, Bilge Mutlu, and Michael Gleicher. Collisionik: A per-instant pose optimization method for generating robot motions with environment collision avoidance. arXiv preprint arXiv:2102.13187, 2021.

[119] Preeti Ramaraj. Robots that help humans build better mental models of robots. In Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction, pages 595–597, 2021.

[120] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. Annual review of control, robotics, and autonomous systems, 3:297–330, 2020.

[121] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. Communicating robot arm motion intent through mixed reality head-mounted displays. In Robotics Research, pages 301–316. Springer, 2020.

[122] Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. The Annals of Mathematical Statistics, 27(3):832 – 837, 1956.

[123] Carl F Ruoff. Teleoperation and robotics in space, volume 161. Aiaa, 1994.

[124] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. Artificial intelligence: a modern approach, volume 2. Prentice hall Upper Saddle River, 2003.

[125] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. IEEE transactions on acoustics, speech, and signal processing, 26(1):43–49, 1978.

[126] Maram Sakr, Martin Freeman, HF Machiel Van der Loos, and Elizabeth Croft. Training human teacher to improve robot learning from demonstration: A pilot study on kinesthetic teaching. In 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 800–806. IEEE, 2020.

[127] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. Proceedings of the IEEE, 109(5):612–634, 2021.

[128] Thomas B Sheridan. Space teleoperation through time delay: Review and prognosis. IEEE Transactions on robotics and Automation, 9(5):592–606, 1993.

[129] Weiyong Si, Ning Wang, and Chenguang Yang. A review on manipulation skill acquisition through teleoperation-based learning from demonstration. Cognitive Computation and Systems, 3(1):1–16, 2021.

[130] David Silver, J Andrew Bagnell, and Anthony Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. The International Journal of Robotics Research, 29(12):1565–1592, 2010.

[131] Joao Silvério, Yanlong Huang, Leonel Rozo, Sylvain Calinon, and Darwin G Caldwell. Probabilistic learning of torque controllers from kinematic and force constraints. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–8. IEEE, 2018.

[132] Marco AC Simões, Robson Marinho da Silva, and Tatiane Nogueira. A dataset schema for cooperative learning from demonstration in multi-robot systems. Journal of Intelligent & Robotic Systems, 99(3):589–608, 2020.

[133] Dennis Sprute, Klaus Tönnies, and Matthias König. A study on different user interfaces for teaching virtual borders to mobile robots. International Journal of Social Robotics, 11(3):373–388, 2019.

[134] Lorenzo Stella, Andreas Themelis, Pantelis Sopasakis, and Panagiotis Patrinos. A simple and efficient algorithm for nonlinear model predictive control. 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pages 1939–1944, 2017.

[135] Mike Stilman. Task constrained motion planning in robot joint space. In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3074–3081. IEEE, 2007.

[136] Chansu Suh, Terry Taewoong Um, Beobkyoon Kim, Hakjong Noh, Munsang Kim, and Frank C Park. Tangent space rrt: A randomized planning algorithm on constraint manifolds. In 2011 IEEE International Conference on Robotics and Automation, pages 4968–4973. IEEE, 2011.

[137] Petr Svestka. On probabilistic completeness and expected complexity for probabilistic path planning, volume 1996. Utrecht University: Information and Computing Sciences, 1996.

[138] Daniel Szafir. Mediating human-robot interactions with virtual, augmented, and mixed reality. In 2019 International Conference on Human-Computer Interaction (HCI), pages 124–149. Springer, 2019.

[139] Aaquib Tabrez, Shivendra Agrawal, and Bradley Hayes. Explanation-based reward coaching to improve human performance via reinforcement learning. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 249–257. IEEE, 2019.

[140] Aaquib Tabrez, Matthew B Luebbers, and Bradley Hayes. A survey of mental modeling techniques in human–robot teaming. Current Robotics Reports, 1(4):259–267, 2020.

[141] Shawna Thomas, Marco Morales, Xinyu Tang, and Nancy M Amato. Biasing samplers to improve motion planning performance. In Proceedings 2007 IEEE international conference on robotics and automation, pages 1625–1630. IEEE, 2007.

[142] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. arXiv preprint arXiv:1905.13566, 2019.

[143] Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh Janabi-Sharifi. Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42(4):1039–1052, 2012.

[144] William Vega-Brown and Nicholas Roy. Asymptotically optimal planning under piecewise-analytic constraints. In Algorithmic Foundations of Robotics XII, pages 528–543. Springer, 2020.

[145] Luigi Villani, Hamid Sadeghian, and Bruno Siciliano. Null-space impedance control for physical human-robot interaction. In Romansy 19–Robot Design, Dynamics and Control: Proceedings of the 19th CISM-Iftomm Symposium, pages 193–200. Springer, 2013.

[146] Najdan Vuković, Marko Mitić, and Zoran Miljković. Trajectory learning and reproduction for differential drive mobile robots based on gmm/hmm and dynamic time warping using learning from demonstration framework. Engineering Applications of Artificial Intelligence, 45:388–404, 2015.

[147] Michael Walker, Hooman Hedayati, Jennifer Lee, and Daniel Szafir. Communicating robot motion intent with augmented reality. In 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 316–324, 2018.

[148] Michael E Walker, Hooman Hedayati, and Daniel Szafir. Robot teleoperation with augmented reality virtual surrogates. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 202–210. IEEE, 2019.

[149] Zhuping Wang, Yunsong Li, Hao Zhang, Chun Liu, and Qijun Chen. Sampling-based optimal motion planning with smart exploration and exploitation. IEEE/ASME Transactions on Mechatronics, 25(5):2376–2386, 2020.

[150] Matt Webster, David Western, Dejanira Araiza-Illan, Clare Dixon, Kerstin Eder, Michael Fisher, and Anthony G Pipe. A corroborative approach to verification and validation of human–robot teams. The International Journal of Robotics Research, 39(1):73–99, 2020.

[151] Jonathan Weisz, Peter K Allen, Alexander G Barszap, and Sanjay S Joshi. Assistive grasping with an augmented reality user interface. The International Journal of Robotics Research, 36(5-7):543–562, 2017.

[152] Sebastian Wrede, Christian Emmerich, Ricarda Grünberg, Arne Nordmann, Agnes Swadzba, and Jochen Steil. A user study on kinesthetic teaching of redundant robots in task and configuration space. Journal of Human-Robot Interaction, 2(1):56–81, 2013.

[153] Chris Xie, Sachin Patil, Teodor Moldovan, Sergey Levine, and Pieter Abbeel. Model-based reinforcement learning with parametrized physical models and optimism-driven exploration. In 2016 IEEE international conference on robotics and automation (ICRA), pages 504–511. IEEE, 2016.

[154] Tomonori Yamamoto, Niki Abolhassani, Sung Jung, Allison M Okamura, and Timothy N Judkins. Augmented reality and haptic interfaces for robot-assisted surgery. The International Journal of Medical Robotics and Computer Assisted Surgery, 8(1):45–56, 2012.

[155] Xuan F Zha. Optimal pose trajectory planning for robot manipulators. Mechanism and Machine Theory, 37(10):1063–1086, 2002.