Martin Tykal

Optimizing Programming by Demonstration for in-contact task models by Incremental Learning

School of Electrical Engineering

Department of Electrical Engineering and Automation

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Technology

Espoo, August 15, 2015

Instructor: Ph.D. Alberto Montebelli
Aalto University
School of Electrical Engineering

Supervisors: Professor Ville Kyrki
Aalto University
School of Electrical Engineering

Professor Thomas Gustafsson
Luleå University of Technology

Preface

First of all, I would like to thank my supervisor Ville Kyrki and my instructor Alberto Montebelli for giving me the chance to work on this project. I want to thank Ville especially for always being on hand with help and advice and giving me free choice of the topic and direction of the thesis. My deepest gratitude goes to Alberto for his guidance, motivation and confidence he placed in me all through the work and his assistance to improve the language and style of this thesis.

I also want to thank my SpaceMaster colleagues for two incredible years and many unforgettable memories. Special thanks to those who stood together with me through this thesis in Helsinki.

Finally I would like to express my gratitude to my dearest Sarah, Benni, Chris and my parents who were always there to support me throughout this time.

Espoo, August 15, 2015

Martin Tykal

Aalto University
School of Electrical Engineering Abstract of the Master's Thesis

Author:	Martin Tykal
Title of the thesis:	Optimizing Programming by Demonstration for in-contact task models by Incremental Learning
Date:	August 15, 2015 **Number of pages:** 12+72
Department:	Electrical Engineering and Automation
Programme:	Master's Degree Programme in Space Science and Technology
Professorship:	Automation Technology (AS-84)
Supervisors:	Professor Ville Kyrki (Aalto)
	Professor Thomas Gustafsson (LTU)
Instructor:	Ph.D. Alberto Montebelli

Despite the increasing usage of robots for industrial applications, many aspects prevent robots from being used in daily life. One of these aspects is that extensive knowledge in programming a robot is necessary to make the robot achieve a desired task. Conventional robot programming is complex, time consuming and expensive, as every aspect of a task has to be considered. Novel intuitive and easy to use methods to program robots are necessary to facilitate the usage in daily life.
This thesis proposes an approach that allows a novice user to program a robot by demonstration and provides assistance to incrementally refine the trained skill. The user utilizes kinesthetic teaching to provide an initial demonstration to the robot. Based on the information extracted from this demonstration the robot starts executing the demonstrated task. The assistance system allows the user to train the robot during the execution and thus refine the model of the task.
Experiments with a KUKA LWR4+ industrial robot evaluate the performance of the assistance system and advantages over unassisted approaches. Furthermore a user study is performed to evaluate the interaction between a novice user and robot.

Keywords: Programming by Demonstration, Incremental Learning, Human-Robot Interaction, Assistance

Contents

List of Tables

List of Figures

Abbreviations

BIC	Bayesian Information Criterion
DMP	Dynamic Movement Primitives
DOF	Degree of Freedom
DTW	Dynamic Time Warping
FRI	Fast Research Interface
GMM	Gaussian Mixture Model
HMM	Hidden Markov Models
HRI	Human Robot Interaction
IL	Incremental Learning
KCP	KUKA Control Panel
KRC	KUKA Robot Controller
KRL	KUKA Robot Language
LWL	Locally Weighted Learning
LWPR	*Locally Weighted Projection Regression*
LWR	*Locally Weighted Regression*
NAC	Natural Actor-Critc
OPS	Orocos Program Script
Orocos	Open Robot Control Software
PA	Proactive Assistance

PbD	Programming by Demonstration
PI^2-CMA	Policy Improvement with Path Integrals with Covariance Matrix Adaptation
PI^2	Policy Improvement with Path Integrals
PoWER	Policy Learning by Weighting Exploration with the Returns
RL	Reinforcement Learning
RMS	Root-Mean-Square
ROS	Robot Operating System
TDGMR	Time-Dependent Gaussian Mixture Regression
UDP	User Datagram Protocol

Chapter 1

Introduction

Robots are increasingly utilized in industrial and domestic applications [IFR, 2014]. However, several challenges avert them from being extensively used in daily life. For example, safety regulations well summarize current concerns regarding humans sharing a common workspace with robots. In human-robot interaction, conveying the intention of a user to the robot constitutes another open problem. Crucially, the high costs for robot programming are a further issue to be addressed, as applying conventional programming to state of the art applications is complex, time consuming and expensive. In fact, every aspect of a potentially complex task has to be explicitly considered, and such an assessment is often not simple or even feasible.

Programming by Demonstration (PbD) has developed to constitute a wide field within robotics [Billard et al., 2008]. PbD has introduced novel ways to address aforementioned challenges thus facilitating the use of robots in daily life. Once the general software infrastructure for PbD is available, robots can be trained on new tasks and become operative within minutes. Accordingly, the traditional role of the programmer as the person who implements code is overtaken by an instructor, who directly provides demonstrations of the required skill to the robot. The goal of PbD is to enable the robot to extract necessary and important information about the task from the demonstration and generalize over one or several demonstrations to novel situations. In contrast to conventional programming, where trajectories are programmed by an expert and then replayed, the robot can be directly instructed by a person without any prior knowledge in robotics. Therefore, new possibilities for Human Robot

Interaction (HRI) and the integration of robots in human workspaces to solve collaborative tasks arise. Calinon et al. [2014] described the current situation in robotics as *"reminiscent of the pre personal-computer age when people needed to be expert in computer programming to make the computer achieve a desired task. As with personal computers, the development of robots and the reduction of cost are now reaching a point where more natural and user-friendly interfaces are required to re-program the robot"*.

Tasks for industrial robots normally require a physical interaction between the robot and an object, and are thus called *in-contact tasks*. Often, such interactions require a precise control of the arising interaction forces, as too low or too high forces might result in a failure of the task or even damage the robot or its environment. The interaction forces that can cause undesired effects are not only the forces exerted by the robot, but include friction or external forces. Friction, for example, can cause the manipulator to jerk or bounce. These aspects increase the complexity of programming a robot in an industrial setting even further, as all possible interactions of the robot with its environment have to be considered and reasonable constraints have to be implemented to protect the robot and its environment while ensuring the fulfillment of the task. This complexity can be reduced by PbD, as the instructor can perceive the results of the exerted forces and act accordingly during the demonstration.

In a near future, small and medium size companies will have an increasing interest in industrial robots as Industry 4.0, the combination of factories with virtual reality, plays a growing role in manufacturing [IFR, 2014]. Energy-efficiency, growing consumer markets, decreasing products' life cycles and an increase in the variety of products are only a few reasons for increasing automation and demand for industrial robots and their quick and flexible programming. In fact, orders of goods produced in low quantity could be optimally processed by flexibly training a robot by demonstration without necessarily hiring a software engineer.

Current PbD approaches have proven successful in teaching robots relatively complex tasks which would require extensive and expensive software development by expert programmers. However, PbD also presents several open issues. A basic issue, called *correspondence problem*, occurs due to different embodiment between the instructor and robot [Calinon and Billard, 2007]. A further

problem relates to the quality of the demonstration. Through the demonstration, the instructor has to effectively transfer to the robot the dynamics required by the task. Several teaching methods will be presented in Section 2.1. All these methods force the instructor to face unfamiliar conditions during the demonstration of the task. For example, they might require additional sensors attached to his body or dragging a robotic manipulator attached to the familiar tool. Such requirements can negatively affect the quality of the demonstration. Poor demonstrations will result in poor skills acquired by the robot. Thus, it is necessary to support the human instructor with natural and intuitive means to effectively convey skills to the robot. In principle, a robotic system can improve the quality of the demonstration and rise above the level of skill that had initially been demonstrated. Such refinement techniques can contribute to such a natural and intuitive training interface and are the focus of this thesis.

1.1 Problem

Little research is currently available in the technical literature for PbD robot training for in-contact tasks, despite their significance for the industry. Also, the use of state of the art industrial robots with in-built force and joint torque sensors and variable compliance for in-contact tasks relies on relatively limited research. Traditional industrial robots have to work in controlled production lines, where the poses of tools and work pieces are strictly known and their environment is completely predictable. The compliance offered by state of the art robots ensures a good contact between tool and work piece, even if the poses of both slightly differ from the nominal poses. Furthermore compliant manipulators are safer, as uncontrolled high forces between tool and work piece are prevented.

Therefore, this new class of robots, like the KUKA LWR [Bischoff et al., 2010], are particularly well suited for in-contact tasks. The KUKA LWR has furthermore the advantage of torque sensors in every joint and a force torque sensor in its wrist. Thus no additional hardware is required to measure forces exerted by the robot or external forces that act upon the robot. Montebelli et al. [2015] developed a system that encoded simultaneously the trajectory of the tool and the force profile of a task. The system was tested by teaching a KUKA

LWR4+ robotic arm to plane a wooden plank. The same control system was further extended to handle disturbances in dynamic environments during the reproduction of a sequence of hand written characters [Steinmetz et al., 2015].

The general problem of PbD is to transfer the professional skill of an instructor to a robot, while preserving the quality of the skill. The work reported in this thesis aims to the creation of a human-robot interface that feels natural, intuitive and user friendly to the human instructor in order to effectively transfer skills for physical in-contact tasks to a robot. The experiments reported in the thesis contribute to its goal by introducing and demonstrating the coordinated use of two main elements of such an interface: *Incremental Learning (IL)* and *Proactive Assistance (PA)*. As training progresses through a sequence of multiple demonstrations, the system can build up confidence about its knowledge of the task (IL). As such a confidence is built, the robot will reduce its compliance and actively make use of the accumulated information to assist the movements of its instructor. Furthmore is the force of the human exerted on the robot measured and amplified (PA), thus the effective inertia of the robot attached to the tool is reduced. This interface has the ambition to ensure minimal disturbance to the instructor by minimizing the impact of unfamiliar modifications of the tool and procedures for the execution of the task.

Therefore the approach presented by Montebelli et al. [2015] will be extended firstly, by implementing the possibility to update the model of the system incrementally. Secondly, *virtual tool dynamics* will be implemented, which will reduce the inertia of the robot by amplifying the force exerted by the instructor.

1.2 Structure

This thesis is structured in six chapters. Chapter 2 will offer an overview of PbD and training methods in use. Chapter 3 will present current research of machine learning techniques, used to refine the quality of reproduction after initial demonstrations. Chapter 4 will present the hardware and software used in this thesis to support the experimental work. Experiments and results are presented in Chapter 5. Chapter 6 contains a critical discussion and associated conclusions.

Chapter 2

Programming by Demonstration

This chapter offers an overview of how demonstration and reproduction works in PbD. First, different teaching methods will be introduced. The two main classes of methods used in the technical literature, symbolic and trajectory-based learning, are presented in detail.

2.1 Training Methods

In PbD, transfering a physical skill from a human to a robot is done by demonstrations. Several methods exist to demonstrate a skill to a robot, which can be categorized into *teleoperation*, *kinesthetic teaching* and *observational learning* [Kormushev et al., 2013].

Teleoperation is a technique, where the robot is remotely controlled by the instructor with an input device, like a joystick or haptic gloves. Teleoperating an overactuated robot, which has more than 6 DOF, is complicated, especially if smooth and dexterous motions are desired. Teleoperation is used, among others, for teaching grasping tasks, as done by Schmidts et al. [2011], who used a haptic glove to record both force and position data. The combination of both data led to higher success rates in teaching grasping movements and an improved generalization capability.

In kinesthetic teaching, the robot is grabbed and moved manually by the instructor and the movements are recorded. However, the robot needs to be small

and lightweight with a gravity-compensation controller to be effectively moved manually by a human instructor. Furthermore it can be difficult with complex robots to only move desired limbs, without moving other parts of the robot too. The aim of this thesis is to implement a system that provides assistance during kinesthetic demonstrations and reduces the complexity of kinesthetic teaching. An advantage is that kinesthetic teaching can be used to directly teach relatively complex in-contact tasks, for example wood planing as in [Montebelli et al., 2015].

Observational learning methods make use of sensors, which can be external or attached to the instructor, to perceive the actions of the instructor. These sensors can be RGBD-cameras or motion sensors worn by the instructor. Observational learning offers a teaching process which is easy and natural for the instructor and it is possible to generate very smooth and human-like motions from such teaching trials. Calinon and Billard [2007] used a combination of observational learning, with motion sensors, and kinesthetic teaching to teach smooth trajectories and then improve them by manually moving the limbs of the robot to reduce the impact of the correspondence problem.

One issue in PbD is to find a mapping between the body of the instructor and the robot that suits the execution of the task. An advantage of teleoperation and kinesthetic teaching over observational teaching is, that the robot can perceive all actions with its own sensors, eluding mapping problems between instructor and robot. Conversely, in observational learning, all actions are perceived by external sensors and thus have to be mapped to the robot.

2.2 Representing Actions

Skills can be learned in two different ways: At a more abstract level, skills can be interpreted as a concatenation of several primitive actions, each represented by a symbol. Learning on this level is called *symbolic learning*. An alternative is to represent skills at trajectory level, which is called *trajectory learning*.

2.2.1 Symbolic Learning

The principle of symbolic learning is that basic actions, identified with symbols, are defined beforehand and a demonstration is encoded as a sequence of these symbols. Symbolic learning consists of several steps, that can be seen in Figure 2.1.

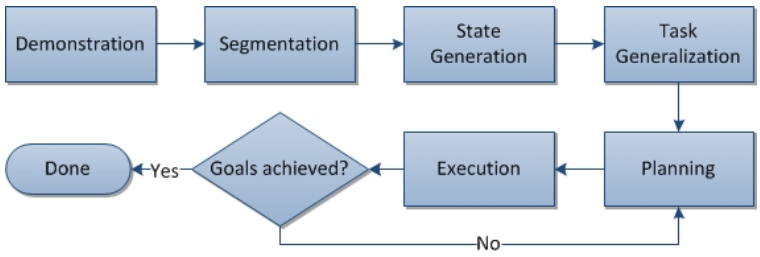

Figure 2.1: Principle of Symbolic Learning (Adapted from [Ekvall and Kragic, 2006])

Ekvall and Kragic [2006] described the principles of symbolic learning. First one or multiple demonstrations are given and perceived by the system. In the segmentation step, machine learning techniques, for example Hidden Markov Models (HMM) are used to partition the demonstration into a set of primitives. Typically, to allow the recognition of primitives, they have to be defined in advance. In the *state generation block* all given demonstrations are used to model subtasks as states. These states are used for generalization in the *task generalization block*. From the demonstrations constraints are extracted, which need to be fulfilled to execute the task properly. An example of such constraints could be to first open a bottle before pouring its content in a glass. Figure 2.2 shows how constraints are learned from demonstrations. In both demonstrations, state B comes before the states F and G. Thus it is a constraint that B has to be executed before F or G. With more demonstrations, more constraints can be learned. With both, the states and constraints, possible execution sequences are build up. If only one demonstration is provided, the system will execute the states in the same order as demonstrated.

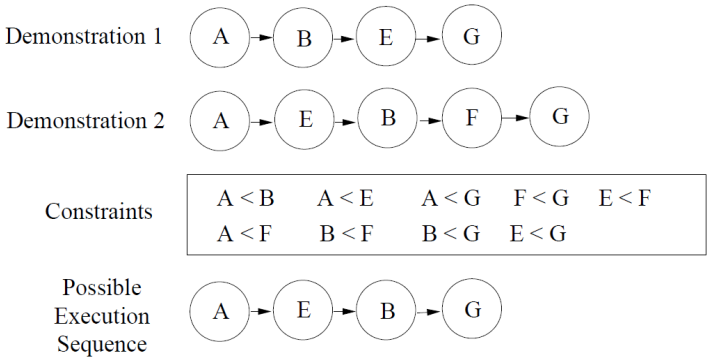

Figure 2.2: Task generalization from two demonstrations

The execution sequence is calculated in the *planning block*, where the environment is taken into account. The execution is planed in a way that the goal can be reached, for example by defining how to grab an object or by avoiding collisions given the current context. Then the sequence is executed. If the goal is not reached, the next sequence is planned until the goal is reached.

Symbolic learning enables a robot to learn whole tasks as a sequence of abstract high-level states. Thus it is easier for human user to understand, what the robot is doing and why. The disadvantage of this approach is the huge amount of prior knowledge that has to be provided for example action primitives and properties of the objects in the workspace.

2.2.2 Trajectory Learning

In trajectory learning trajectories are learned from provided demonstrations with the goal to generalize over these demonstrations and generate trajectories fulfilling the requirements of the demonstrated task. These trajectories can be represented as probability distributions or dynamic models. It is necessary to choose a space in which the robot will operate, depending on the task. Trajectories can be recorded in joint space, task space or torque space [Billard et al., 2008]. For executing a task, it might be also necessary to record additional data. For example, in-contact tasks usually require the integration of positional information with information about exerted forces [Kormushev et al., 2011, Montebelli et al., 2015, Steinmetz et al., 2015].

Trajectory encoding with Statistical Models

One possible approach to encode trajectories is to use statistical models. The hypothesis behind this approach is that although several demonstrations might differ between each other they should be very similar during crucial elements of the task. Therefore, a mean trajectory and its variance can be calculated. In this case, the variance can be interpreted as a measure for how close the reproduced trajectory has to follow the mean trajectory during the execution of the task. Portions of the trajectory characterized by high variance (i.e. less homogeneous behavior), can be exploited for fulfilling additional requirements or, in general, to react to changes in the environment, e.g. obstacle avoidance [Mühlig et al., 2009]. Invariant portions of the trajectory are presumably important to successfully reach a goal. Accordingly, they should closely match the mean trajectory. Alternative methods that have been suggested to model the probability density functions of the demonstrated trajectories are presented in the following sections.

Gaussian Mixture Models The use of a Gaussian Mixture Model (GMM) consists of two steps: Observation and learning. During the observation phase datapoints are acquired with sensors. The data is then usually preprocessed by *temporal normalization*. In fact, demonstrations typically present temporal distortions and inconsistencies, as a human will not be able to give multiple identical demonstrations. Therefore, to retrieve useful information from multiple demonstrations, the data has to be temporally normalized. For this purpose Dynamic Time Warping (DTW) is often used, where the trajectories of all demonstrations are aligned to minimize the distance between each of them [Mühlig et al., 2009]. Figure 2.3 shows the effect of using DTW on acquired demonstrations. Only after the application of DTW it is possible to find a reasonable mean and covariance. The learning phase follows the observational phase, where GMM are built with several *Gaussians*. Each Gaussian consists of a *mean vector*, a *covariance matrix* and an initial likelihood, the *prior*. The model is then created with a mixture of several Gaussians with the dimensionality of the datapoints plus time [Calinon et al., 2007, Mühlig et al., 2009]. To retrieve the parameters (mean, covariance and prior) from the mixture of all Gaussians, different algorithms can be used to train the GMM, such as the expectation-maximization (EM) algorithm. The EM algorithm maximizes the

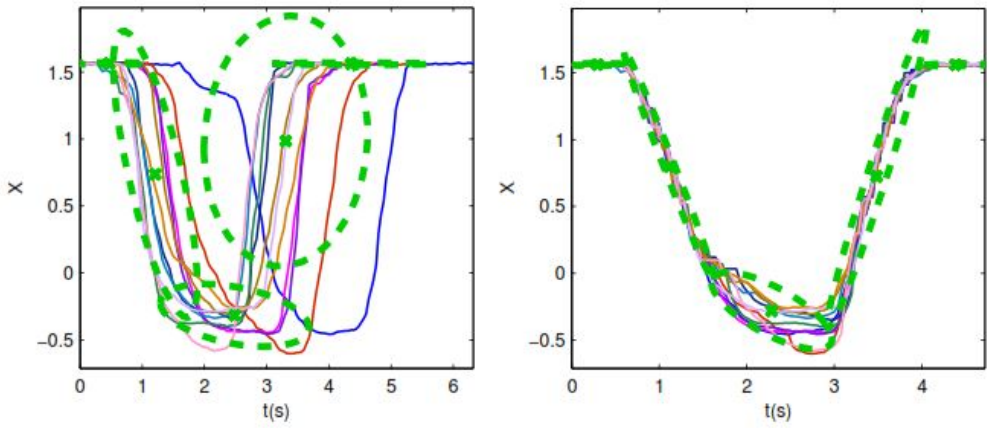

Figure 2.3: GMM without DTW (left) and with DTW (right). The green ellipses show the learned mean and covariances (Source: [Mühlig et al., 2009])

likelihood that the GMM represents the probability density function of the acquired data during the observation phase [Mühlig et al., 2009]. The Bayesian Information Criterion (BIC) is often used to determine the number of Gaussians that produces the best model and to quantify the quality of a model [Mühlig et al., 2009]. To find the best amount of Gaussians components, the number of components is increased, the GMM trained and the quality of the model then evaluated by the BIC. The smallest BIC value refers to the estimated optimal amount [Mühlig et al., 2009]. Retraining the GMM after each step is time consuming, therefore Mühlig et al. [Mühlig et al., 2009] proposed to merely approximate the ideal model, resulting in sufficient accuracy and a speedup of two orders of magnitude compared to using EM for each step.

GMM have the advantage that there is no distinction between inputs and outputs, any subset of the dimensions can be defined as the input, while the remaining subset is defined as the output. This mean in practice that, for example, conditioning on the outputs of a model of the forward dynamics of an robotic arm gives automatically a model of the inverse dynamics of that robotic arm [Cohn et al., 1996].

Locally Weighted Learning Locally Weighted Learning (LWL) is a class of techniques for the approximation of functions around the current point of interest [Atkeson et al., 1997]. These methods assign a weight to every new data set, specifying the influence of this data on the learning process. This weight is depending on the position of the input points in dataspace relative to

the point which has to be predicted. Data points close to the prediction point have higher weights [Friedman, 1995]. In contrast to model-based methods, such as GMM, where training data are generally discarded after every learning trial, LWL methods, which are memory-based, relearn from all data when a prediction is needed. This is a useful feature for IL. One method, called *Locally Weighted Regression* (LWR), implies a regression that is performed locally around the point of interest, as it can be seen in Figure 2.4 [Cohn et al., 1996]. The problem with most LWL methods is, that with an increase of dimensions the number of local models needed for accurate estimations increases exponentially. This is also true for LWR and thus it is not memory efficient and not suited for high-dimensional applications [Vijayakumar et al., 2005]. Vijayakumar et al. [2005] developed another LWL method called *Locally Weighted Projection Regression* (LWPR), which combines LWR and projection regression, which reduces the dimensionality locally. LWPR is specially developed for IL. It provides fast learning and is computationally efficient, as not all training data have to be kept in memory.

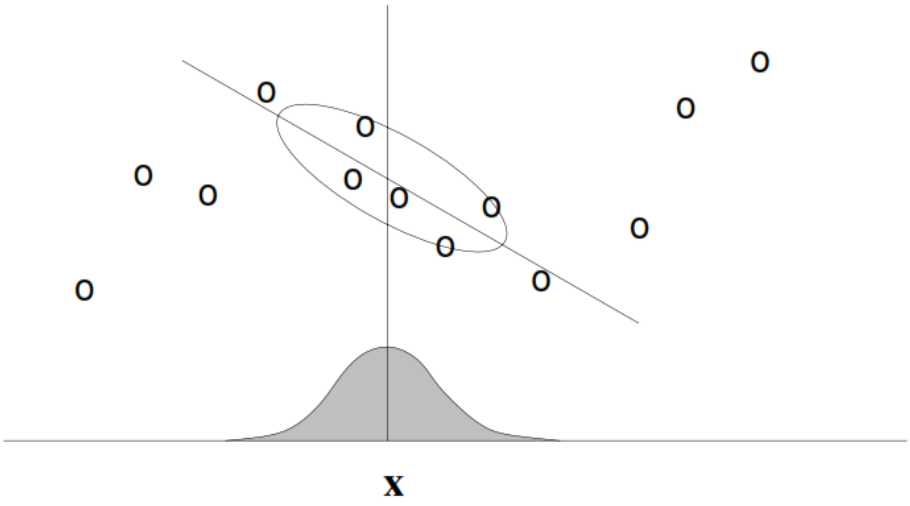

Figure 2.4: Data points are weighted depending on their distance to the point of interest x and a regression is calculated. (Source:[Cohn et al., 1996])

Hidden Markov Models Hidden Markov Models (HMM) are Bayesian networks, which represent probabilities over a sequence of observations. A HMM consists of states (denoted as 1 and 2), transition probabilities t and the probability $p_j(x)$ for an emitted observation symbol x in state j. The states are proba-

bility distributions, defined by mean and covariance matrix. The last parameter of a HMM is the initial probability π_j of being in state j. These parameters can be learned by the *EM* or *Baum-Welch Algorithm* or by optimization techniques like the Gradient Descent algorithm [Calinon et al., 2010a][Khreich et al., 2012]. HMMs are defined by two properties: *Hidden states* and the *Markov property*. Hidden means, that the emitted symbol sequence is observable, but the actual state within the sequence of states is not. The Markov property described the fact, that the current state is only dependent on the previous state [Ghahramani, 2002]. HMMs are widely used for many different purposes, like speech recognition [Rabiner, 1989], handwritten word recognition [El-Yacoubi et al., 1999] or cybersecurity [Warrender et al., 1999]. Their capability to model real-world phenomena, like human motion, in terms of simple and compact models, makes them interesting for PbD applications [Khreich et al., 2012]. Inamura et al. [Inamura et al., 2004] described in this context the *Mimesis Model*, which uses HMMs to recognize and generate human motions. Similar to GMM, where the number of Gaussians has to be defined, in HMM the number of states has to be defined. Also for HMM the BIC-criterion can be used to find the needed number of states. A drawback of HMMs is, that a high number of states is needed to correctly reproduce complex motions or a smoothing procedure is used, which reduces important peaks in the motion [Calinon et al., 2010a].

Trajectory encoding with Dynamical Systems

Dynamical systems are another possibility for learning skills at the trajectory level. They are designed to be stable and robust against perturbations. In PbD, a widely used model is Dynamic Movement Primitives (DMP)[Schaal et al., 2007].

Dynamic Movement Primitives DMPs have received a lot of attention in the robotics community in recent years. This method was introduced 2002 [Ijspeert et al., 2002a,b] and further developed by Schaal et al. [2007]. A detailed overview was presented by Ijspeert et al. [2013]. The principle of DMPs is that a simple and well understood simple attractor system (namely a spring damper system relaxing from an initial to its steady state position) is extended with forcing function terms to remap onto a desired attractor (in the case of

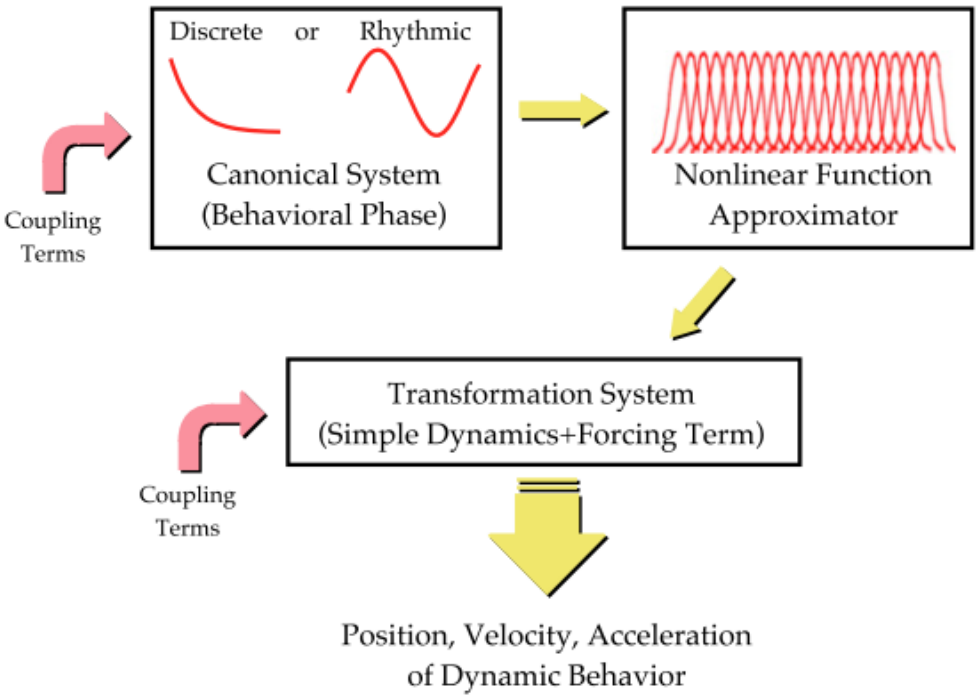

Figure 2.5: Structure of DMPs (Source:[Ijspeert et al., 2013])

PbD, the demonstrated trajectory). A DMP consists of a *canonical* and, independently for each demonstration, a *transformation system*, as it can be seen in Figure 2.5. DMPs were developed to encode both discrete and oscillating trajectories. The canonical system models the behavior of the model equations, e.g. whether it acts as a point attractor or a limit cycle. Furthermore it serves as a replacement of time to make the transformation system formally independent of time [Ijspeert et al., 2013]. The canonical system can be described as

$$\tau \dot{x} = -\alpha_x x, \tag{2.1}$$

with a time constant τ and a positive constant α_x. x is a phase variable which converges from an initial state x_0 to zero, where $x = x_0$ indicates the start and x close to zero denotes that the goal g has been reached. The transformation system consists of a simple dynamic system, like a spring-damper system, which is then transformed into a desired attractor system with forcing terms f. Formally:

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f, \tag{2.2}$$

$$\tau \dot{y} = z, \tag{2.3}$$

with positive constants α_z and β_z, desired position, velocity and acceleration $y, \dot{y}, \ddot{y}$ and forcing term f. To let y monotonically converge towards g, thus critically damping the system, β_z should be defined as $\beta_z = \alpha_z/4$. The forcing term f could be chosen, for example, as

$$f(x) = \frac{\sum_{i=1}^{N} \Psi_i(t) w_i}{\sum_{i=1}^{N} \Psi_i(t)} x(g - y_0), \tag{2.4}$$

with adjustable weigths w_i, the initial state y_0 and fixed basis functions Ψ_i,

$$\Psi_i(x) = exp\left(-\frac{1}{2\sigma_i^2}(x - c_i)^2\right), \tag{2.5}$$

where σ_i and c_i are the width and centers of the basis functions [Ijspeert et al., 2013].

To learn the parameters of the system different regression methods can be used, like LWR or LWPR [Ijspeert et al., 2013]. The advantages of DMPs are manifold. It can be used to both classify and replicate movements. Classification can be done with the parameters (weights) of the function approximator. Furthermore it is possible to decompose complex movements into a concatenation of movement primitives [Ijspeert et al., 2013]. DMPs scale well for high dimensionality. Ijspeert et al. [2013] described different methods to cope with multiple DOF. A simple method is to use one canonical system for all DOF and a transformation system for each DOF. One of the main advantages is the possibility of online modification of the primitive. As shown in Figure 2.5, coupling terms are introduced to both the canonical and transformation system. These coupling terms can be used to modify the trajectory online, for example to apply a different scaling [Ijspeert et al., 2013]. Dynamical systems are stable in the presence of small random noise, but for large perturbations, the coupling terms can be used as feedback terms to modify the trajectory accordingly [Billard et al., 2008].

Comparison

Despite the differences between described learning methods and the different use cases, almost no comparisons exist in literature. Calinon et al. [Calinon et al., 2010a] compared HMM, LWR, LWPR, DMP and Time-Dependent Gaussian

Mixture Regression (TDGMR), which uses GMM to encode the distribution of spatial and temporal variables, with each other. Five metrics are used in the comparison: Root-Mean-Square (RMS) Error, RMS Error after DTW, Norm of Jerk, learning time and retrieval duration. The RMS error evaluates how accurate in spatial and temporal terms the reproduced trajectory matches the demonstrations. RMS error after DTW emphasizes on the spatial information, as the RMS error is calculated after aligning the trajectories with DTW. The norm of jerk measures the smoothness of a trajectory, based on the derivative of acceleration. As the name implies, the learning time is the time the algorithm needs to learn the trajectory, while the retrieval duration is the computation time of the retrieval process for one iteration. The results for different number of states and seven dimensions can be seen in Figure 2.6. The RMS error before

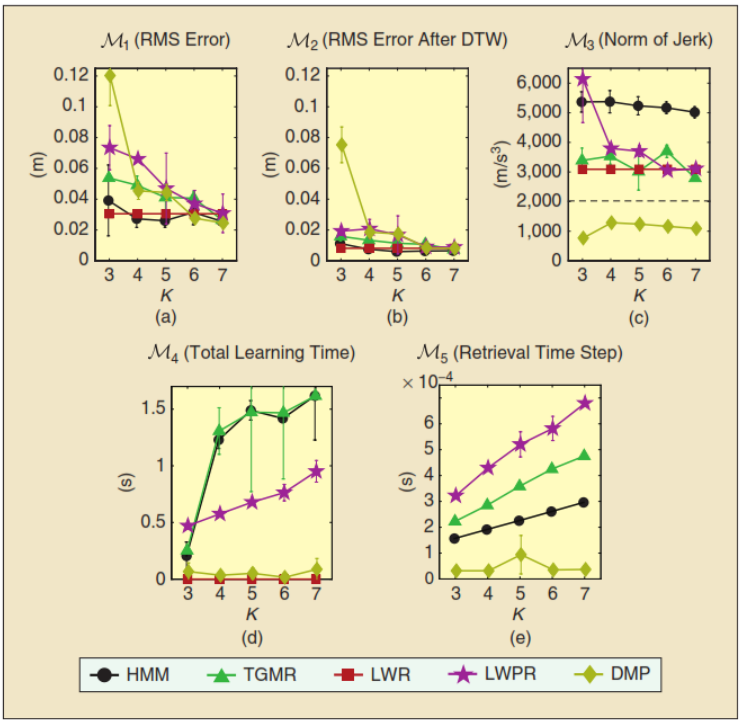

Figure 2.6: Influence of the number of states for seven dimensions on the different metrics. The dashed line in the norm of jerk represents the mean RMS jerk of the demonstrations. (Source: [Calinon et al., 2010a])

and after DTW shows only a little deviation from the learned trajectory for all examined learning methods. The jerk is smallest with DMP, which generates the smoothest trajectories. The learning time is smallest for DMP and LWR, while HMM and TDGMR, which are both trained with the EM algorithm, show the worst performance. As the EM algorithm used in this test starts its search

procedure randomly, the results are very variable. The online learning capability of LWPR needs special consideration, as ten learning trials with random data has been performed. For a single trial the learning time can be reduced by an order of magnitude. The retrieval time is low for all methods, but for LWR comparatively high with $7 \times 10^{-2}s$, so that it does not appear in the graph anymore. The retrieval time is almost constant for DMP, while it is linearly increasing with the number of states for the other methods. All methods except LWR show a low retrieval time of less than 1 ms and are thus usable for online use.

2.3 Discussion

PbD is an useful approach to teach robots new skills without experts in robotics. Different teaching approaches exist, but kinesthetic teaching proves to be intuitively usable while avoiding the correspondence problem, thus it will be the teaching approach in the process of this thesis. For learning skills two main fields, symbolic and trajectory learning, exist. Symbolic learning requires a great amount of prior knowledge and is thus not as versatile and easy to use as here intended. Trajectory learning offers a great capability of generalization and fast learning. In comparison to other considered methods, DMPs showed little deviation from the demonstrations while producing smooth trajectories with short learning and retrieval time. LWR proved to be the fastest learning method. Thus DMPs will be used to in this thesis, in combination with LWR to learn primitives from demonstrations.

Chapter 3

Post-Training Optimization

The outcome of PbD is a robot with an acquired capability to perform a task. Normally, the quality of its performance is directly related to the quality of the demonstration received by the robot during its training. This chapter will give an overview of methods that address the problem of improving the acquired performance. Two major techniques exist for this improvement. Firstly, following demonstrations, the overall quality can be further enhanced by giving the robot feedback when its execution of the task cannot be considered satisfactory. This feedback can take the form of a new demonstration for the whole task again or can be limited to the unsatisfactory parts of it. Such additional demonstrations are functional to an incremental update of the underlying model. For this reason, this supervised technique is called IL [Cho and Jo, 2013]. Secondly, reward functions can be used to allow the system to automatically explore new behaviors and determine what can be interpreted as beneficial or detrimental behavior. The former will be rewarded whereas the latter will be punished by the reward function. Thanks to the reward function, the robot will explore the states of its environment and the actions to reach them with an unsupervised trial and error method. With the reward function ranking the actions of the robot, the system can balance the exploitation of the already learned actions and the exploration of new solutions to incrementally increase its performance in the given task. This framework is called Reinforcement Learning (RL) [Sutton and Barto, 1998].

3.1 Incremental Learning

Teaching a practical skill in daily life is often done in several steps. First an instructor provides a presentation, then the learner attempts to reproduce, while supervised by the instructor. In case the learner produces a poor reenactment of the task, he or she receives feedback from the instructor about what needs to be improved. Additionally the instructor can show the demonstration again, enabling the learner to generalize over different demonstrations, as it can be seen in Figure 3.1. Such a procedure is designated in robotics as IL [Nicolescu and Mataric, 2003].

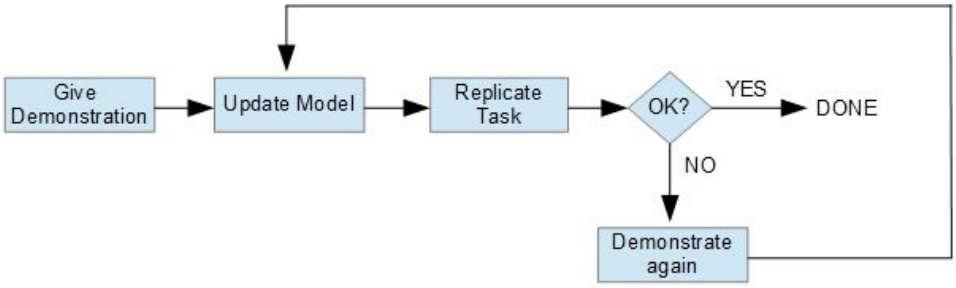

Figure 3.1: Principle of Incremental Learning

3.1.1 Feedback

The feedback loop shown in Figure 3.1 summarizes the main difference between IL and ordinary PbD approaches. When the replication of the task is unsatisfactory, multiple demonstrations of the whole task (or parts of it) can be performed to incremental update the model. Social cues can be used to highlight situations, for example by pointing or gazing, where the system has to achieve accurate reproduction of crucial details or better performance is required [Calinon and Billard, 2008]. A direct way to offer feedback is by kinesthetic teaching, as described in Section 2.1, where parts of the robot are physically moved. The method of kinesthetic teaching was used with a HOAP-3 robot by Calinon and Billard [2007]. After the initial demonstration, the robot performed the learned skill. The reproduced gestures are then fine-tuned by manually moving the limbs of the robot. The same method was used by Cho and Jo [2013] with a

NAO humanoid robot, with the difference that the robot could withdraw flawed demonstrations (see section 3.1.2).

An interesting research direction addresses the use of pointing and gazing, which attracts the attention of the robot to a certain area or situation to simplify the full potential complexity of conveying the skill. The direction of a human supervisor's pointing or gazing can be determined with external sensors like stereo cameras. The human instructor will naturally turn his head to the situation which is important for the task. Thus head tracking offers an intuitive way to focus the attention of the robot and to provide useful information. Calinon and Billard [2008] used pointing and gazing information as a prior for the statistical framework, which extracts the task constraints. According to their approach, demonstrated actions received a weighting depending on whether they occurred within the gazing area. The benefit of social cues depends on how accurately the robot can detect what is highlighted. A reliable method to estimate the information in pointing gestures was developed by Park and Lee [2011] to recognize pointing gestures, allowing an exact target selection in 99% of all cases.

A different approach to the problem of creating feedback consists in segmenting the demonstration into primitives (e.g. "take object", "move object" and "place object") and in focusing retraining on the unsatisfactory ones only. This method has a broad prevalence in recent research, for example see Wu et al. [2014] and Jain and Inamura [2014]. Jain and Inamura used IL in a sophisticated setting. The robot received demonstrations for different tasks, like nailing or cutting. The robot has to discriminate the demonstrated tasks (e.g. "Beat the nail into the wood") and determine which tool is useful for the task ("Use a tool with a large mass"), how to orient the tool (e.g. "Use the head of the hammer, not the grip") and how to use the tool ("Hit the hammer towards the nail"). Vision is used to extract the properties of a tool: form, size and area of the effective part of the tool. After learning how to use the tools, the robot has to perform a similar task with a different tool. This is done to test whether the robot has correctly learned the important functional features of a tool (e.g. the blade of a knife or the head of a hammer) and is able to choose a good strategy for the available tool, the task and spatial constraints. If the reproduction fails or is not good enough, the robot makes a query to the demonstrator to solve the current task with the given tool. This approach was tested successfully with pulling an object towards the robot by first using a stick, then a T-shaped tool

and finally rakes, where the robot had to decide which rake to use, depending on the distance to the object [Jain and Inamura, 2014].

3.1.2 Models

The use of IL requires special consideration about the learning method. A great number of studies, for example the work done by Calinon and Billard [2007] or Cho and Jo [2013], use Gaussian Mixture Model (GMM) as the underlying model to encode demonstrated tasks.

However, GMM has two drawbacks when used for incremental learning: Firstly, the model cannot be updated, but needs to be relearned and secondly, the number of Gaussian components, as described in Section 2.2.2 needs to be defined beforehand ([Cederborg et al., 2010, Cho and Jo, 2013, Wu et al., 2014]). The first drawback is computational expensive, while the second one reduces the flexibility of the model. Over recent years much research has been published about improving GMM for incremental learning.

Kristan et al. [2008] proposed an approach to incrementally add new samples to a GMM and remove old samples if necessary. With this approach it is possible to not only add new information to the model, but also remove erroneous information, which could have been introduced by a poor demonstration. The advantage of the approach is that it is not necessary to tune parameters for specific applications, the form of the probability function can be arbitrary and no time constraints are assumed. The disadvantage is that it was only tested for one dimensional problems and needs to be extended for multi-dimensional learning tasks. Cho and Jo [2013] proposed a method where the trial data is abolished once used and the number of Gaussians is refined after each trial. The temporal normalization with DTW makes it possible to compare the trajectories of different demonstrations of the same task. Evaluating the pattern similarity enables the robot to autonomously assess the quality of a teaching trial and to reject demonstrations that differ too much from the other demonstrations. This method ensures that no false information by erroneous demonstrations are introduced to the model. Cederborg et al. [2010] described an “Incremental, Local and Online formulation of Gaussian Mixture Regression (ILO-GMR)”. This approach allows to not only learn incrementally new tasks, but also sev-

eral tasks at once. Every task is assigned to a certain environment. If the environment changes during the demonstration, the system recognizes, that a new task is shown. Correspondingly different tasks are reproduced according to the environment. The advantage of this approach is its flexibility. A whole workflow can be taught to the robot with the possibility to refine single steps with additional demonstrations.

HMMs in their original form have the same two drawbacks as GMM, as the model can not be updated and the number of states needs to be defined beforehand. HMMs can handle spatial and temporal variabilities of human demonstrations well and are thus, with some modifications, an interesting solution for IL [Calinon et al., 2010a]. Khreich et al. [2012] surveyed how on-line learning techniques are used for HMMs to incrementally learn the parameters. Different approaches, like incremental versions of the EM algorithm, exist. Kulic et al. [2008] used HMM to segment motions and incrementally cluster them, based on distances measured with log likelihood. Every motion primitive is added to a hierarchical tree and represents a group of similar motions. All observed motions are encoded into a HMM, compared with existing motion primitives and grouped with the closest primitive. Local clustering is then performed on the modified group. If a subset of motions similar enough is found, this subset forms a new group, therefore a new primitive. Utilizing this method, the system learns incrementally based on the observations.
Dawood and Loo [2014] proposed the Topological Gaussian Adaptive Resonance Hidden Markov Model, where the data from the demonstration is processed into a topological map, which is used to update the state structure of the HMM. The remaining parameters of the HMM are learned with an incremental EM algorithm. This approach solves both drawbacks. The topological map defines the number of states, thus the number of Gaussians does not need to be predefined and the modified EM algorithm provides on-line learning for all other parameters.

3.1.3 Teaching various stiffnesses

Recent development in industrial robotics yields robots with controlled stiffness like the KUKA LWR4+, as described by Bischoff et al. [2010]. IL is used in combination with controlled stiffness in two ways. First, it enables the instructor

to move the robot during the reproduction to refine the task, as it will be shown in the course of this thesis. Second, the system can be trained on multiple trajectories in which different portions of the different demonstrations can be very similar or highly variable. This information can be used to tune the robot's stiffness during replication of the task.

Calinon et al. [2010b] proposed a compliance controller that is compliant in intervals of the trajectory where the variance among demonstrations is limited (this is meant to model portions of the task that pose crucial constraints on the execution) and relatively soft where the variance is small. The general goal of this approach is to enable robots to work in unstructured environments (e.g. human workplaces) in contrast to controlled production lines. In unstructured workspaces the relationship between robot, tool and environment tends to be unpredictable and variable for each trial. Nevertheless, a compliant manipulator can assure sufficient contact force, as shown by Calinon et al. [2010b] based on an ironing task. The manipulator could effectively press the iron on a plate, independent of the reciprocal position of iron and plate, without exerting harmful forces. A further result of the approach is to ensure safety for the human worker by controlling the stiffness of the robot. The robot is operated within a sequence of possible different danger levels. To assess the current danger the distance between the robot and a human is measured and the viewing angle of the human is determined by head tracking. The highest danger level occurs when the user is close to the robot and facing away. The level is lower when the user can see the robot or the robot is further away. The robot replicates the task slowly and tries to find trajectories to avoid the user if the level is high, while reproduction at a low danger level is fast and with the nominal trajectory.

Using the variance as a parameter for the stiffness makes it also possible to teach compliant behavior to the robot. Kronander and Billard [2012] described a method to incrementally teach certain stiffnesses to certain sections of a trajectory. The use of compliant behavior is described with a pouring task and with lightening a match. Transporting a bottle of soda to a empty glass should be compliant, to avoid aggressive response to perturbations, while during the pouring the manipulator should be stiff to avoid missing the glass. A good example for the benefit of compliance for in-contact tasks is lightening a match. On one hand a stiff robot might use too much force and break the match or use too less force to generate the needed friction. On the other hand is exact

position tracking difficult for a compliant manipulator. Thus the robot is taught the motion of moving toward the matchbox and of striking the match. In reproduction the robot is incrementally taught to remove the compliance during the matching by wiggling the elbow of the robot, increasing the variance. The result was a 85% success rate instead of only 15% with both only high or only low stiffness.

Lee and Ott [2011] used variable stiffness to create a motion refinement tube. The motion refinement tube is the implementation of an impedance controller that prevents the instructor from giving demonstrations too divergent from the other demonstrations by setting the stiffness according the variance of the model. A high variance at a certain point along the trajectory will result in a low stiffness at that point during the demonstration. Respectively a small variance will result in a high stiffness. Furthermore it is ensured that only parts of the robot are moved, which are necessary for the refinement, for example accidentally moving the torso when motions of the arm are refined. This property is implemented by setting a higher stiffness to joints closer to the base [Lee and Ott, 2011].

Another interesting approach that combines incremental learning and high compliance of the robot was described by Saveriano et al. [2015]. The goal was to exploit redundant DOF of a redundant robot to execute end-effector tasks, e.g. following a trajectory, and null-space tasks, which use the redundant DOF, at the same time. Instead of an impedance controller, as in [Lee and Ott, 2011], a task transition controller was implemented. First, the end-effector task is trained with kinesthetic teaching. Second, the task transition controller starts the execution of the trained motion primitive and detects interactions of the user by measuring external forces. These interaction forces are used to generate a new task. Such a task could be a certain elbow motion of a robotic arm to avoid an obstacle during execution of the end-effector task. Thus, a certain robot motion can not only be trained and refined, but also adapted to new constraints, like obstacles. The adaption uses the redundant DOF of the robot and hence it does not disturb the previously learned motion.

3.2 Reinforcement Learning

Reinforcement Learning (RL) is an unsupervised trial-and-error approach. Instead of supervising the robot and refining its actions, the robot is enabled to explore its own environment and find better ways to solve a task. The integration of RL with PbD is useful in cases, where the demonstration is not sufficient enough to fulfill a task, for example due to big changes in the environment or suboptimal demonstrations. The combination of both is also useful to limit the search space for possible solutions, as further described in Section 3.2.3. With RL the robot can learn a way to reach its goal autonomously by trial-and-error exploration of its environment and exploiting explored actions to accumulate reward, specified by a reward function.

3.2.1 General idea of RL

The core of each RL approach is the *reward function*, which maps for every state-action pair a *reward*. A *state* is a vector with all current information about the system, for example position and velocity, which can be used to predict future states. The set of all existing states of the system is called *state-space*. The goal is to accumulate as much reward as possible. One of the main challenges in RL, after defining a reward function, is to define how to accumulate this reward. Depending on the assigned value for each state, certain actions are chosen to reach the next state. An *action* is a transition between states, in robotics this can be a control signal. The mapping between states and actions, e.g. which action is chosen for a certain state is called the *policy*. A greedy algorithm is trivial and will always choose the next state with the highest reward. More sophisticated approaches will also take the rewards of upcoming states into account. This happens with a *value function*, which maps for every state an additional value. While the reward expresses what is immediately beneficial, the value specifies what is beneficial in a long term. The value is an estimation of how much reward can be accumulated starting from that state [Sutton and Barto, 1998]. In contrast to plain PbD, the policy is not obtained from the demonstration, but through exploration of possible alternatives [Argall et al., 2009]. The values of the states are added, the path through the different states yielding the highest reward will lead to the execution of the desired task. The

goal is to obtain a policy which results in such an execution [Argall et al., 2009] [Sutton and Barto, 1998].

Additionally to finding a policy, another challenge of RL is the trade-off between exploration and exploitation. On one hand the robot needs to try different actions to know which actions lead to success. On the other hand the robot needs to exploit those actions to accumulate the reward. Without exploration, no exploitation is possible. Without exploitation, no reward will be accumulated [Sutton and Barto, 1998]. Exploration is time consuming and it is necessary to limit it to an extent that the system is able to find effective actions which can be exploited. Complex systems like humanoid robots have huge state-spaces. One role of PbD within RL is to limit the state-space to states important for the task.

3.2.2 Early Work

Research for RL in robotic applications started in the 1980s with the goal of a general refinement of robot motor skills [Franklin, 1988]. Extensive research was done by Sutton and Barto [1998] who gave an overall description of RL, different methods how to obtain the policy and approaches to combine these methods. The main solution methods are Dynamic Programming, Monte Carlo Methods and Temporal-Difference Learning. Sutton and Barto distinguish between planning methods, that need a model of the environment, and learning methods, that can be used without such a model. A drawback is the inefficiency for high dimensional cases. With the increase of dimensions, the state space grows exponentially, which was called the “Curse of Dimensionality” by Bellman in 1957 [Sutton and Barto, 1998]. Exploring the whole state space is computational too expensive, but this problem is reduced with initial demonstrations. RL can be combined with PbD by demonstrating the task and thus providing prior information. These information consists of which states are important for the task and which actions are useful for these states. This limits the state-space and only states close to the demonstrated states must be explored.

3.2.3 Reinforcement Learning as Post-Training Optimization

The principle of RL aided by demonstrations is that trajectories from a demonstration are learned and generalized and the parameters of the built probabilistic model are forwarded to the RL module. The calculated trajectories are then executed and if the goal is not reached, the RL module tries to find better parameters for the model. The demonstration highlights important states and actions to limit the state-space and reduce the amount of necessary exploration.

Over the past years different approaches attempted to lower the impact of the "Curse of Dimensionality" with initial demonstrations and better learning algorithms. Peters et al. proposed the Natural Actor-Critc (NAC) algorithm for applications utilizing humanoid robots with many Degree of Freedom (DOF) [Peters et al., 2003] [Schaal et al., 2004]. The demonstration is first encoded with DMPs (see section 3.2.3) and the weights of the learned function approximator are then improved by a reward function. The policy is obtained by a natural policy gradient, which is estimated by the NAC. This method was used with a 30-DOF humanoid robot to match template trajectories.

Based on the NAC algorithm, Guenter et al. [2007] developed an approach where RL is used to cope with unforeseen obstacles. In cases where the information provided by demonstrations is not sufficient to reach the goal (final position), the robot uses RL to learn how to avoid the hurdle and reach its goal. As it can be seen in Figure 3.2, the demonstration is encoded and generalized with GMM to obtain a single model trajectory. The parameters of the GMM are the input for the RL module. The RL generates a modulation variable, which feeds the Dynamical System, where the control commands are generated. Simulations assess whether the goal is reached and finally a trajectory is executed. If the task fails, another modulation variable is calculated by the RL module. The advantage of this approach is the usage of a simulation, as hundreds of trials with the real robot are not practical and time consuming. With this method a way around the obstacle can be learned within seconds. However, this approach requires a precise model to use simulation for learning behavior. Kober et al. [2013] pointed out that only for stable tasks the learned behavior can be applied to the real world. Perturbations like contacts or friction are hard to model, thus learning in the real world gives better results than simulations. As a consequence

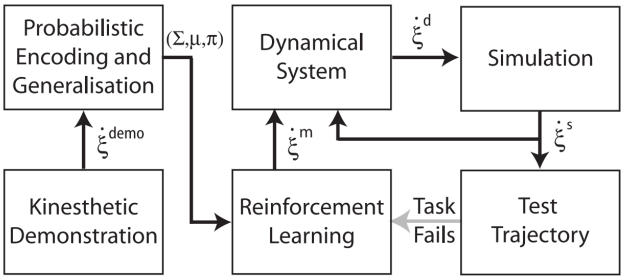

Figure 3.2: The principle of RL as post-training optimization method (Source: [Guenter et al., 2007])

simulations are not suitable for in-contact tasks, despite their advantage of fast learning. Thus RL for in-contact tasks has to be used with real world trials, which requires considerably more time.

In the work of Guenter et al. [2007] the demonstration was only a tool to limit the state space initially. In more recent research, Gräve et al. [2010] used demonstrations to highlight important states and respective actions. The proposed algorithm compares provided demonstrations with the current situation. If a demonstration was similar to the current situation, the best action for the current state from that demonstration is compared with the action the system would execute in the current state. If the two actions are too different, the algorithm asks for another demonstration. However, if the actions are similar enough, it is safe to let the robot perform RL and explore its environment. The advantage of this approach is the limitation of human demonstrations to cases where they are necessary. RL is used if new obstacles are presented and the deviation between demonstrated information and planned action is within a defined limit. With increasing number of demonstrations, the RL module could handle more situations by exploring. The goal was to pick up a cup, which was placed randomly in a 600 cm^2 workspace. Only 15 demonstrations were used to exemplify the task. A similar approach was chosen by Sakato et al. [2014]. This approach distinguishes between two cases. If there is an instructor giving a demonstration, the robot will follow the presentation and learn. If no demonstration is given, the robot will decide whether it will perform a reinforced action or a learned action, depending on the similarity of the current state with the demonstrated state.

Two main topics in recent research are the creation of robots, suited for more complex tasks than pick-and-placing tasks, for example surgical robots, and the use of DMP. Calinon et al. [2014] worked on a surgical robot that can perform endoscopic surgery. First, a surgeon teaches a robot different via-points, which the robot has to reach without damaging close organs. The problem of teaching the robot is the different embodiment of the surgeon and the flexible octopus-inspired 9-DOF STIFF-FLOP robot. Thus an intermediate step is necessary. The intermediate step is to demonstrate the task on a stiff robot. Therefore kinesthetic teaching of the task is performed on a stiff 7-DOF Barrett WAM robot in gravity compensation mode. The demonstration of the surgeon is used to extract a reward function. Second, the STIFF-FLOP robot uses this reward function and performs the task, exploiting its variable body characteristics and exploring new solutions to find a good policy. The initial demonstration of the surgeon is deficient due to the different embodiment between the WAM and the STIFF-FLOP robot [Malekzadeh et al., 2013]. These flaws are compensated with the refinement by RL, as the STIFF-FLOP learns how to execute the task with its flexible body and reaches a skill level beyond the demonstration of the surgeon. This approach was tested in two experiments, one simulated in software, one performed in a chamber filled with balloons, representing organs. The robot was able to follow the via-points much closer after the refinement than by just imitation learning [Calinon et al., 2014].

3.2.4 Reinforcement Learning with Dynamic Movement Primitives

RL can be used to refine the initial demonstration by updating the weights, which are initially set with the demonstration [Nemec et al., 2012, Pastor et al., 2012]. Current development in learning algorithms with DMP are reward-weighted averaging algorithms like Policy Improvement with Path Integrals (PI^2) or Policy Learning by Weighting Exploration with the Returns (PoWER). These methods appear to be extremely effective and outperform gradient based algorithms like NAC. Gradient based algorithms have many parameters that have to be tuned to result in fast convergence. With reward-weighted averaging algorithms only the noise variance has to be selected [André et al., 2015, Nemec et al., 2012, Pastor et al., 2012]. PI^2 and PoWER, both a direct development

of the NAC, introduce noise into the policy parameters to create new possible solutions. These algorithms can be used to learn the weigths and the goal state for the DMP. The biggest advantage of the combination of DMP and PI^2 is that it does not need any model or cost function for learning [Pastor et al., 2012]. An improvement to PI^2 is PI^2-CMA which also estimates a covariance matrix, which enables to learn inter-parameter relations to increase convergence speed [André et al., 2015]. Nemec et al. used the PoWER algorithm to teach a robot with two KUKA LWR arms to pour a certain amount of liquid in a glass, without detecting the current amount of liquid poured (only the motion of pouring was considered). Two demonstrations were used to generalize the task and reduce the dimension of the state-space and PoWER was used to optimize the volume of poured liquid, while minimizing the amout of spilled liquid for different goal volumes. The experiments demonstrated fast learning rates of the algorithm [Nemec et al., 2012]. Andrè et al. compared PoWER and PI^2-CMA in simulation. The algorithms were used to optimize the walking velocity of a bipedal robot for different inclinations of the floor. In all test the PI^2-CMA resulted in higher velocities [André et al., 2015].

3.3 Discussion

IL can be utilized not only to adapt the skills of a robot to new constraints, but also to improve the skills. Frequently used statistical learning methods like GMM and HMM have to be modified to allow incremental on-line learning. Several approaches to overcome the need of predefining the number of Gaussians/states and incrementally learning the parameters of the model can be found in the literature. With the occurrence of robots providing controllable stiffness and force sensing, new possibilities to utilize IL arose. These possibilities consist of helping the instructor to give better additional demonstrations to refine the skill of the robot and also to teach the robot various levels of stiffness for different parts of the task. RL is one of the fundamental machine learning areas, which has been used for decades now. Despite its usefulness, its full potential has not been discovered yet. The STIFF-FLOP project leads the way in exploiting RL for optimizing the skill of a robot after the demonstration phase. Currently is RL in combination with Programming by Demonstration mostly used to cope

with unforeseen events during the replication of a task, instead of refining the skill to solve the task.

The use of IL or RL for in-contact tasks is barely researched. IL seems to be best suited to refine the skills of a robot for in-contact tasks because of its supervised manner. Using RL for in-contact tasks is challenging, due to the exploration that is necessary part of RL. The problem is that the robot has to test different forces between tool and workpiece, thus endangering itself and its environment. Therefore it is mandatory to find limits for the exerted force and possibilities to distinguish between wanted and unwanted contacts. Future research is needed to solve the question how to exploit controllable stiffness of robots for in-contact tasks and how to assist the instructor during the demonstrations.

Chapter 4

System

This chapter gives an overview of the soft-, middle- and hardware used to implement an intuitively usable assistance system for PbD tasks. First the hardware will be presented. This is followed up by a description of the different software and middleware components and how they interact with each other.

4.1 Hardware

The hardware setup consists of five components: The KUKA LWR4+ robot, an ATI Mini45 F/T sensor, the KUKA Robot Controller (KRC) with attached KUKA Control Panel (KCP) and an external computer. Figure 4.1 shows how these hardware components are integrated.

Foundation of the conducted experiments is the industrial robot KUKA LWR4+, which is shown in Figure 4.1. The LWR4+ is a kinematically redundant robot with 7 DOF and is thus able to perform dexterous movements, while avoiding singularities. Two properties make this robot well suited for implementing the proactive assistance: Its lightweight design and its torque sensors in every joint with an additional force sensor in the wrist. The robot weighs only 15 kg (LWR stands for lightweight robot), while still able to handle payloads of 7 kg. This property is important to make it manually movable for a human instructor and thus suitable for kinesthetic teaching as described in Section 2.1. Another feature of the design is the round appearance of the robot, reducing the risk of

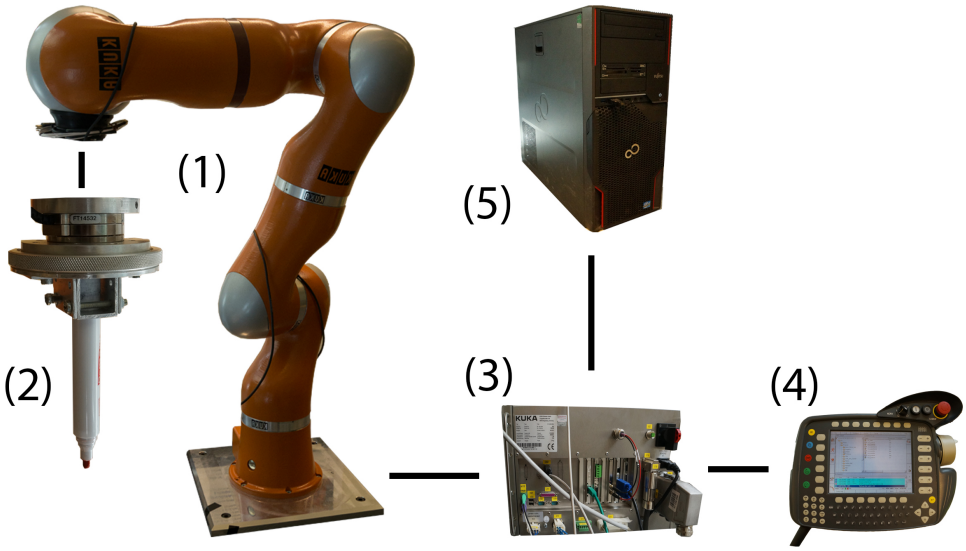

Figure 4.1: Overview of the hardware setup. The KUKA LWR4+ (1), with attached F/T sensor and tool (2), is connected to the KRC(3). Connected to the KRC is the the KCP (4) and the external computer (5).

injury for nearby persons. The second property is the robot's ability to measure torques and forces, which has several advantages. On one hand, it allows active vibration damping to ensure high precision. On the other hand, it facilitates the lightweight design with a sagging compensation of the slim and elastic joints. The robot acts as a spring-damper system, where the desired position, stiffness and damping can be defined both for joint- and Cartesian space. To control this system, the LWR4+ is equipped with three control modes: Position control, Cartesian impedance control and joint impedance control. The control cycle rate is 3 kHz in every joint and 1 kHz overall, enabling to switch the control mode within 1 ms [Bischoff et al., 2010]. These frequencies make it possible to detect and react to contacts and collisions fast, which proves not only useful in the course of this thesis, but for general aspects of physical human-robot interaction, e.g. the safety of human and robot.

An ATI Mini45 F/T sensor is attached between the wrist of the robot and the tool and allows precise measurements of exerted forces and torques within a force range of 290 N in x- and y-directions, 580 N in z-direction and for torque a range of 10 Nm in all directions. The resolution is for the force 1/8 N in all directions and for torque 1/376 Nm in x- and y-direction, and 1/752 Nm in

z-direction [ATI, 2015]. The z-axis of the sensor aligns with the z-axis of the tool frame.

The next component in Figure 4.1 is the KRC, more precisely the KR C2 lr, with the KCP, an input device for the KRC, attached. The KRC consists of a control PC, a power unit, safety logic and a connection panel. The control PC offers a Windows XP user interface and a preinstalled editor for creating, archiving and running programs written in the KUKA Robot Language (KRL). The purpose of the KRC is to control the robot and ensure safe operation.

The external computer is an octo core computer using the linux operating system Ubuntu 12.04 LTE with a Xenomai real-time kernel. The computer is used to control the robot through KUKA's Fast Research Interface (FRI)[Schreiber et al., 2010].

4.2 Middleware

The creation of robotic applications can be simplified by utilizing middleware frameworks. In the course of this thesis two frameworks where utilitzed, namely the Robot Operating System (ROS) and the Open Robot Control Software (Orocos). The most commonly used framework is ROS, which provides several useful libraries and tools for controlling robots. It is modularly structured and thus easy adaptable to many different tasks. But ROS does not support hard real-time which is required for components of this thesis, e.g. for virtual tool dynamics. Thus Orocos is used for software components which require real-time handling. These frameworks are next briefly described.

ROS was created to help programming robotic applications without "reinventing the wheel" and to reuse software components which have been shared by universities, companies and individuals [Goebel, 2013]. Software components in ROS are called nodes and use a publisher-subscriber system to communicate with each other. Nodes can publish topics with certain message types and other nodes can subscribe to these topics to get published data.

Orocos is a toolchain which can be utilized to create real-time software for robotic applications. Software created with Orocos is structured in *components* and *deployers*.

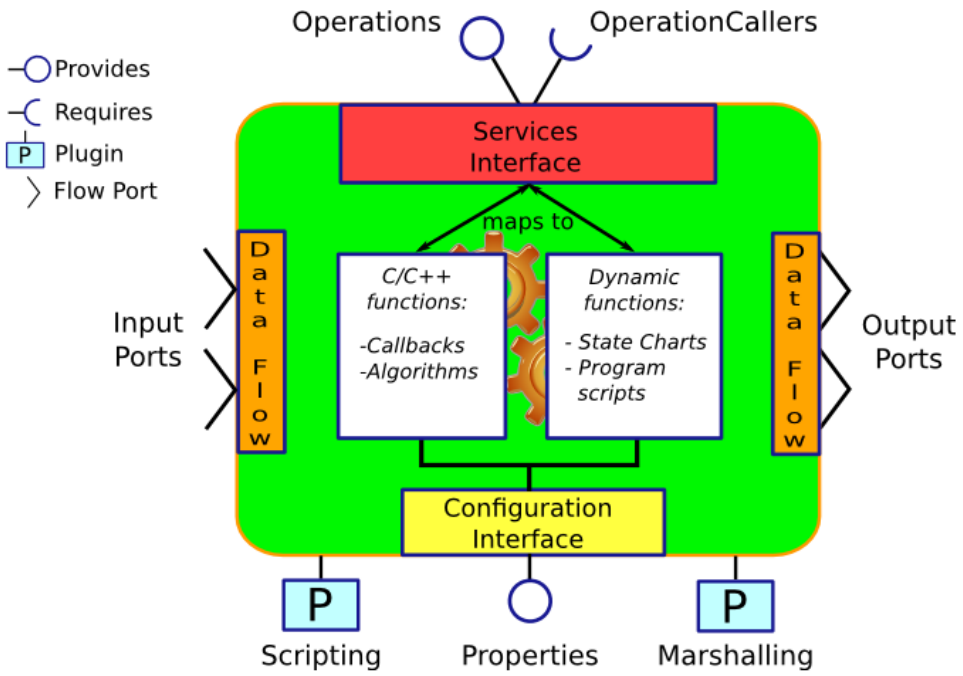

Figure 4.2: The structure of an Orocos component. (Source: Soetens [2012])

A component, abstractly described in Figure 4.2, executes one or several programs written in C/C++ or the Orocos Program Script (OPS) in a single thread. Orocos ensures lock free and thread safe communication within a single or distributed processes and deterministic execution time during this communication [Soetens, 2012]. The functionality of components is implemented in operations, which are plain C/C++ functions. These operations can be used by other components via operation callers. Communication between components is established with ports, which are defined for a certain message type.

A deployer is a file written in OPS and specifies the Orocos environment by including components, setting their *activities* and connections and starts the included components. Components can be defined as peers to each other, which is a prerequisite to use operation callers. Activities can be defined as periodic, non-periodic or slave activities which are triggered when other activities execute. The communication between components is set up as connections between ports. The deployer also offers a way to communicate with ROS by specifying a stream

from a port to a ROS topic, so that a ROS node can subscribe to the data of an Orocos port or publish data to the port.

4.3 Software

The implementation of the incremental assistance system requires a sophisticated software infrastructure that establishes a connection between the external computer and the robot, commands the robot and guides the user through the teaching process. The main components are presented in the following sections and as an overview in Figure 4.3.

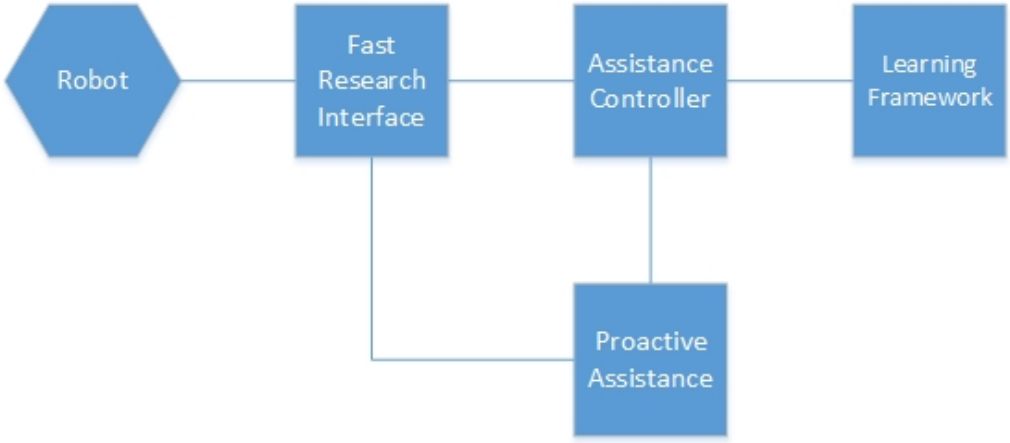

Figure 4.3: Overview of the software infrastructure

4.3.1 Fast Research Interface

The Fast Research Interface (FRI) was developed by KUKA Roboter GmbH [Schreiber et al., 2010] to provide direct low-level real-time access to the KRC and thus control the robot from an external computer. For the communication between KRC and external computer the User Datagram Protocol (UDP) is used with frequencies up to 1 kHz. The FRI is realized as a state machine providing two different modes: Monitor mode and command mode. The monitor mode allows to receive sensor and status readings from the robot, but not to command the robot. FRI is always in monitor mode when started. To be able to switch to command mode, a sufficient communication quality is required. FRI

sends a sequence of messages to the external computer and waits for an answer to evaluate the quality of the connection. In case the timing meets the desired timing specifications, the user may switch from monitor to command mode. Within the command mode the external computer can not only receive sensor readings and status information, but also set the control mode of the robot, change parameters, for example stiffness or damping values and command the robot to certain positions and forces.

4.3.2 Learning Framework

Learning primitives from demonstrations requires a learning framework, as described in Chapter 2. For the assisted incremental learning approach proposed in this thesis, an implementation of the DMP framework was utilized. This implementation was initially implemented by Steinmetz [2013] in the course of his master's thesis and is described there in detail. The imitation part of this existing framework, where learning and execution are implemented, has been largely utilized and extended where necessary. The learning framework consists of three phases: *Imitation*, *learning* and *execution*. In the imitation phase an `Imitator` object is created, which takes parameters, for example the prefix of the recorded files and parameters for the DMP, and creates a `Demonstration` object for every recorded trajectory. These demonstrations are then passed on to the learning class, which uses LWR to learn the parameters during the learning phase. These parameters are then the input for the DMP to calculate the next desired state. In the execution phase these states are interpreted, depending on whether they are represented in joint or Cartesian space, and then commanded to the robot.

4.3.3 ProactiveAssistance

The usability of kinesthetic teaching is strongly dependent on the weight and thus the inertia, of the robot. The user has to overcome this inertia to move the robot, which can result in jerky or inaccurate movements in case of high inertia. To address this problem, virtual tool dynamics were implemented which reduce the effective inertia of the robot and thus the robot feels lighter and

more movable to the user. This sort of assistance was implemented within the Orocos component `ProactiveAssistance`. An implementation in Orocos was necessary to maintain real-time assistance and to avoid delays which would disturb the user during the demonstration. First, the component sets the robot in Cartesian impedance control, to be able to change the stiffess and damping and allow kinesthetic teaching. The internal control law of the controller is [Schreiber et al., 2010]

$$\tau_{cmd} = J^T(k_c(x_{FRI} - x_{msr}) + F_{FRI} + D(d_c)) + f_{dynamics}(q, \dot{q}, \ddot{q}),$$

where τ_{cmd} is the commanded torque, J^T the transposed Jacobian , x_{FRI} the Cartesian setpoint position, $k_c(x_{FRI} - x_{msr})$ the Cartesian stiffness, F_{FRI} the Cartesian force/torque, $D(d_c)$ the damping term and $f_{dynamics}(q, \dot{q}, \ddot{q})$ the dynamic model of the robot [Schreiber et al., 2010].

The exact dynamic model of the robot is not available, however it can be said that, if frictions are ignored, the general dynamic model of the robot is of the form

$$J^T(M(x)\ddot{x} + C(x, \dot{x})\dot{x} + g(x)) = \tau_{cmd}$$

where $M(x)$ is the symmetric, positive definite inertia matrix, $C(x, \dot{x})$ is a combined matrix for the Coriolis and centrifugal terms, $g(x)$ is the gravity vector and τ_{cmd} the commanded torque [De Luca et al., 2006]. $C(x, \dot{x})\dot{x}$ and $g(x)$ are part of $f_{dynamics}(q, \dot{q}, \ddot{q})$, which, according to Schreiber et al. [2010], represents the dynamics of the robot. Placing the dynamic model in the control law gives

$$M(x)\ddot{x} + C(x, \dot{x})\dot{x} + g(x) = k_c(x_{FRI} - x_{msr}) + F_{FRI} + D(d_c) + f_{dynamics}(x, \dot{x}, \ddot{x})$$

It is assumed that $C(x, \dot{x})\dot{x}$ and $g(x)$ are equal to $f_{dynamics}(x, \dot{x}, \ddot{x})$. Then it follows that

$$M(x)\ddot{x} - k_c(x_{FRI} - x_{msr}) - D(d_c) = F_{FRI}$$

Second, the component reads the measurements of the external forces and torques F_{FRI} from the F/T sensor inside the wrist of the robot. A factor α is defined by which the measured force and torque is multiplied ($F_{FRI} := \alpha \cdot F_{FRI}$) resulting in

$$\frac{1}{\alpha}M(x)\ddot{x} = F_{FRI} + \frac{1}{\alpha}k_c(x_{FRI} - x_{msr}) - \frac{1}{\alpha}D(d_c)$$

The division of $M(x)\ddot{x}$ by α reduces the effective inertia perceived by the user and renders the robot easier to move.

This approach eases kinesthetic teaching noticeably. However, it presents a serious drawback for in-contact tasks. In fact, the friction between the tool and environment causes the tool to bounce, as the vibration caused by overcoming the friction is amplified by the system. To tackle this problem the ATI F/T sensor is utilized to detect contacts of the tool with the environment. If such a contact is detected, the Cartesian stiffness is increased in z-direction. The result is a "magnetic" effect, which keeps the tool in contact, until the user exerts force in negative z-direction, thus lifting the tool. This prevents bouncing motions effectively, while maintaining the possibility for the user to exert the desired force.

4.3.4 AssistanceController

The `AssistanceController` is the centerpiece of the software developed and tested for this thesis. It is implemented as a ROS node and offers three modes to the user:

- Training a new skill
- Add new demonstrations to an previous training
- Replaying the learned skill

Teaching the robot new skills incrementally is done with several demonstrations. First an initial demonstration is provided to the robot using `ProactiveAssistance`, as described in the previous section. This demonstration is recorded and an `Imitator` object is passed to the learning framework to calculate an initial model. The robot then sets itself back to the starting position and the first iteration of incremental learning begins. At the beginning of every iteration the stiffness of the robot is increased and the robot moves along the trajectory learned so far. This behavior constitutes help for the demonstrator in two different ways. With low stiffness settings the demonstrator perceives the robot moving slightly forward, which reduces the effort for the demonstrator to overcome the inertia and renders the robot less unsteady. With higher stiffness values the robot performs the task increasingly autonomously. The mental effort of the demonstrator is reduced from giving the whole demonstration and

ensuring that the positions and forces are demonstrated as desired to observing the robot and correcting the behavior of the robot where necessary. Due to the fact that every demonstration differs at least slightly from other demonstrations in speed and position, the learned trajectories require an alignment. After every demonstration the recorded trajectory is aligned in time with the previously recorded trajectory by warping the timelines of both trajectories.The result are two trajectories whose corresponding positions lay at the same points in time. The procedure of the alignment is described in detail in Section 4.3.4. If the user wishes to give more demonstrations, the next iteration starts. The `AssistanceController` leads the instructor through this process while commanding the robot, recording and aligning the trajectories and communicating with the learning framework. Therefore, the controller can be divided into three parts: The *finite state machine*, *imitation* and *Dynamic Time Warping (DTW)*.

State Machine

The state machine shown in Figure 4.4 consists of four states: `Start`, `Record`, `Add/Execute` and `End`. After `AssistanceController` is started the state machine is in the `Start` state. In this state the user is presented with three choices: 1. training a new skill; 2. adding new demonstrations to the current primitive or 3. executing the learned primitive.

In case the user chooses the first option, the state machine moves to the state `Record`. In this state the system is in gravity compensation mode and the user is prompted to move the robotic arm to a starting position. The recording of the initial demonstration, aided with the `ProactiveAssistance`, starts immediately after the user confirms the starting position. Once the demonstration is done, the state machine switches into the `Add/Execute` state to start the imitation and check whether the imitation was successful and whether the user wants to record an additional demonstration or replay the skill. In cases where the imitation could not be completed or where the user quits the program, the state machine transits into the `End` state, where the state machine and hence `ProactiveAssistance` is commanded to stop. If the user chooses the second option thus resuming teaching of a skill from a point he stopped earlier, the state machine transits directly from `Start` to `Add/Execute`, while defining all necessary parameters, e.g. stiffness, to proceed from the desired point. If the

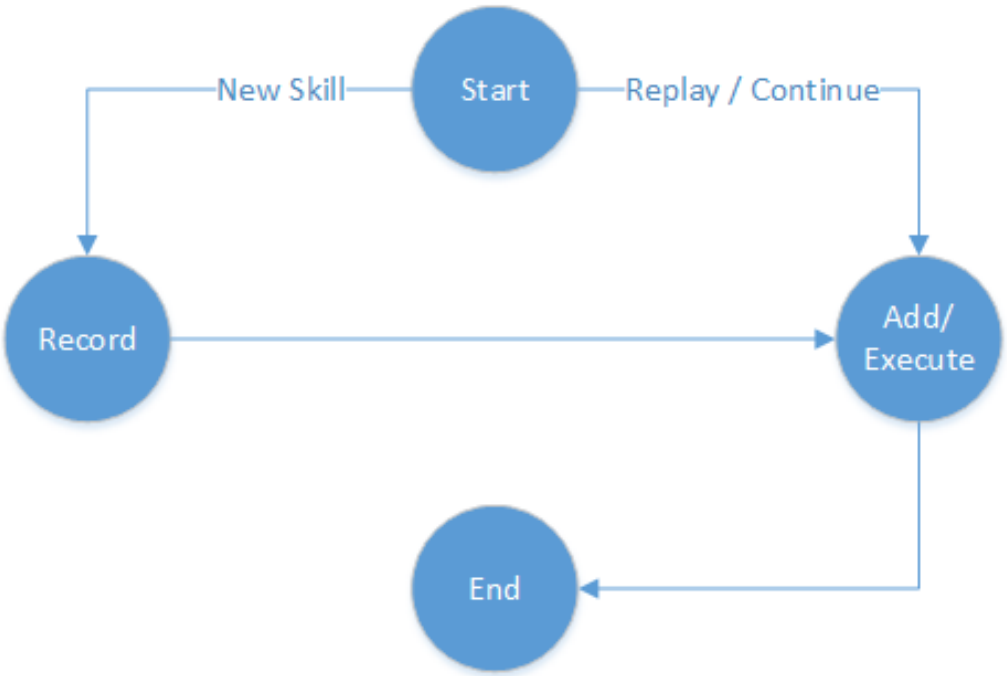

Figure 4.4: Abstract description of the state machine of `AssistanceController`.

user wants to replay a learned skill without adding new demonstrations, they choose the third option in the `Start` state. Then the state machine transits from the `Start` state to the `Execute` state while setting a high stiffness.

Imitation

The imitation module is the implementation of the incremental learning approach and has two purposes: First, replaying the current model of the motion primitive while recording the demonstration of the user and second, setting the stiffness for the current iteration. The first aspect allows to physically guide the robot during the imitation of the motion primitive. The benefit of this approach is that the robot helps the user to overcome the inertia of the robotic manipulator, which results in more natural movements. Furthermore the user can concentrate on correcting the trajectory where it was not sufficient in previous executions. This reduces the complexity of teaching the skill. The latter purpose of the imitation module is to increase the stiffness of the robot incrementally. During the second demonstration, the robot moves autonomously according to the first demonstration. Therefore, the user can move the robot

more freely, assisted by the proactive behavior of the robot. The robot's assistance grows incrementally after each demonstration, as the robot makes use of the previously received information, takes into account the user's corrections and becomes more stiff. This process helps the user to incrementally focus on the necessary corrections rather than on the execution whole task, thus increasing accuracy. The stiffness value S for the iterations is calculated as

$$S = 2^{i-1} \cdot 100\frac{N}{m}$$

where i is the current number of demonstration. A problem in incremental learning is that as the number of demonstrations increases, the relative weight of each demonstration becomes smaller. Therefore, the imitation module forgets old demonstrations and the learning framework utilizes only the last three demonstrations, which are weighed equally, to learn the parameters with LWR. After each imitation, the user has to temporarily pause the recording of new demonstrations and the imitation module will trigger the `DTW` module.

Dynamic Time Warping

A problem that appears with the approach described above is the variability of human demonstrations. Each demonstration will differ from the other ones for speed, trajectories and forces, which renders problems for the learning framework. To reduce the impact of this problem, DTW has been implemented, to align the recorded trajectories with respect to time. The orientation of the tool is set when moving the robot to the start position. The force between tool and surface is coupled with the position, as at a certain point of the trajectory a certain force has to be applied. Therefore force values are stored together with the respective positions so that force-position pairs remain the same after the time warping. Thus only the trajectories are considered for the alignment. To align two trajectories s and t, the algorithm first measures the distance between them by calculating the distances of every point of s with length N to every point of t with length M. As the original DTW algorithm was developed for the one dimensional problems of speech recognition, the distance calculation had to be extended for three dimensional trajectories. The cost function was changed to $|(s_x - t_x)| + |(s_y - t_y)| + |(s_z - t_z)|$, which takes all three dimensions into account (line 8 in Algorithm 1 in AppendixA). These distances are saved in a $N \times M$-matrix. Second, a backtracking algorithm is used to find a path, called

warppath, through the matrix that yields the shortest distances [Müller, 2007]. Every item of the warppath can be referenced by its index inside the matrix. These indices are stored and used to warp the trajectories and align them. For better illustration, DTW algorithms are sketched in Appendix A. Calculating the distance matrix is shown in Algorithm 1, calculating the path in Algorithm 2 and the warping process in Algorithm 3. With this warping process the length of the warped trajectories equals the length of the warppath and is longer than s or t. The more warping is required, the longer the warppath and the resulting trajectories are. This stretching of the trajectories results in parts where the position remains constant. To avoid that the robot stagnates at positions where it should keep moving these expendable parts are removed from the warped trajectory. An issue that might result from this approach, although it has not been observed so far, is that the execution speed might clearly differ for parts of the trajectory than the speed during the demonstration.

Chapter 5

Experiments and Results

Evaluating physical human-robot interaction is a challenge as quantitative measurements reveal the performance of the robot, but also the human side has to be considered. Therefore, the evaluation of the assistance system is done in two parts. The first part will show the benefit of the assistance on learning different geometrical shapes. The second part will utilize a user study to evaluate how intuitive and easy the system is to use and how novice users interact with the system.

5.1 Quantitative Evaluation

A quantitative evaluation of the implemented system is difficult, as the capability of the user to kinesthetically teach a task has a severe influence on the resulting learned primitive. First, the influence of DTW on the learned skill will be investigated on an example of a single teaching trial of teaching the robot to draw a square. Then drawing a square and a circle were chosen as tasks to show the performance of the system. Drawing a square, which requires straight lines and sharp angles, can show the behavior of the system in cases where the desired primitive has sudden changes of direction. Drawing a circle is a relatively difficult primitive, as a constant radius has to be maintained, and thus suitable to reveal the impact of the assistance compared to unassisted training. To cope with the difference in position over several demonstrations, every teaching trial

was performed ten times to reduce the impact of the human influence and to give meaningful results.

5.1.1 Dynamic Time Warping

The implementation of DTW was necessary as the difference in speed during demonstrations results in a poor replication of the primitive. This is due the fact that the learning system outputs the mean of the learned trajectories. The

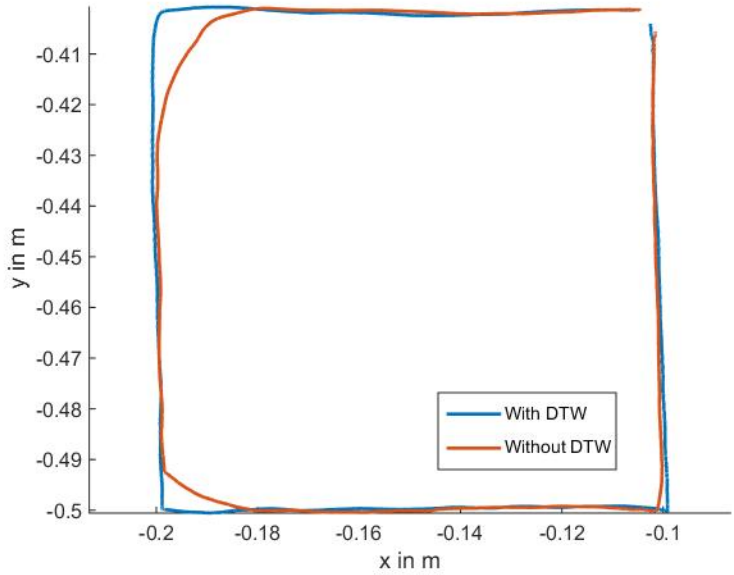

Figure 5.1: The learned motion primitive of the same demonstrations with DTW and without DTW.

assisted incremental system was used to program a square with kinesthetic teaching. Five iterations were performed, the first one fully compliant with virtual tool dynamics and then with increasing stiffness (200 $\frac{N}{m}$, 400 $\frac{N}{m}$, 800 $\frac{N}{m}$, 1600 $\frac{N}{m}$). The recorded trajectories were saved in their original form and, from the second iteration on, time-aligned with the previous demonstration. The resulting primitive, learned from the last three demonstrations, was replicated twice with a stiffness of 4000 $\frac{N}{m}$, once using the original trajectories and once using the DTW-aligned trajectories. The difference can be seen in Figure 5.1. It is apparent that the square primitive extracted using DTW (blue square) shows distinct corners. On the opposite, the primitive learned from non DTW-aligned trajectories, although extracted from the same set of demonstrations, does not

capture the quality of sharp corners. The corners are rounded and changes in direction occur too early. This emphasizes the importance of trajectory-alignment for incremental learning approaches.

5.1.2 Training the robot to draw a Square

The goal of this experiment was to measure the influence of the assistance on the end result compared to teaching trials performed without assistance in gravity compensation mode. To express the result in numbers the mean square error was chosen, which is the distance of the result compared to a perfect square.

Experiment

In order to get comparable results, a template square was programmed in KRL, representing a hardcoded trajectory. The four corner points of the template were marked on sheets of paper with 10 cm distance between them, as displayed in Figures 5.2 and 5.3. Then the robot was trained by the author of this thesis to draw a square on the prepared paper sheets with a standard marker, whose tip had a diameter of 4 mm. The experiment consisted of ten teaching trials in gravity compensation and ten teaching trials with the assistance system. The assistance system used virtual tool dynamics and set the robot fully compliant for the initial demonstration. Four iterations were performed with stiffness values of 200 $\frac{N}{m}$, 400 $\frac{N}{m}$, 800 $\frac{N}{m}$ and 1600 $\frac{N}{m}$. The teaching trials in gravity compensation were performed by grabbing the robot by its wrist. The initial demonstrations for the assisted trials were also performed by grabbing the robot by its wrist. This was necessary to not disturb the force readings of the ATI F/T sensor in z-direction so that the detection of contacts work (see Section 4.3.3). However, as the increased stiffness during the iterations prevent the tool from bouncing over the surface due to friction, the ATI F/T sensor was used to measure the interaction between the human and the robot more precisely. Therefore all iterations after the initial demonstration are performed by grabbing the tool directly.

Results

The result for the teaching trials in gravity compensation mode can be seen in Figure 5.2, which shows the result of the ten teaching trials in gravity compensation mode (orange) to the template (blue). It is apparent that it is hard

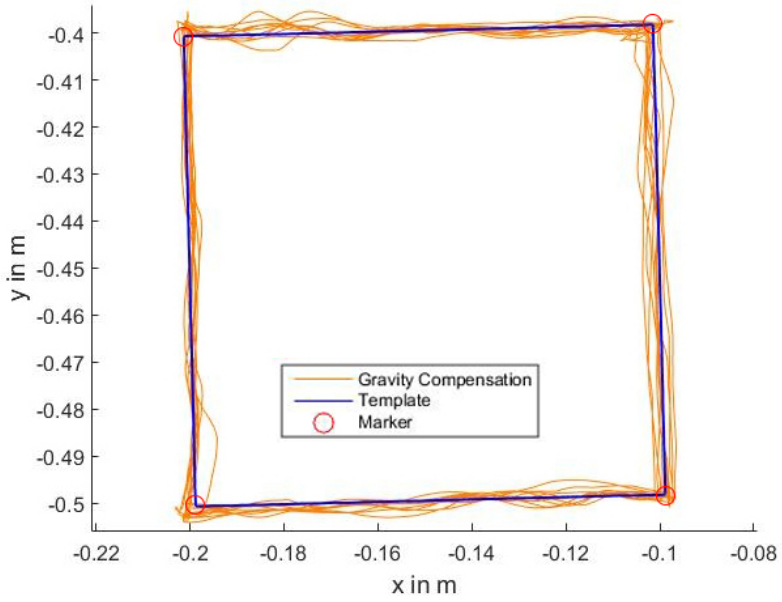

Figure 5.2: The replications of ten teaching trials in gravity compensation mode compared to the template

to move the robot only in the desired direction to connect the marked points with straight lines. The teaching trials were performed clock-wise, starting at the upper right corner and ending in the same point. Directly after the start large jerky deviations from the template are visible, as the inertia of the robot had to be overcome. To draw a straight line a robot needs to perform a synchronized movement of several joints. Kinesthetic teaching without assistance results in the passive movement of several joints that prevents the achievement of a smooth movement for the user steering the robot. The mean distance of all trajectories trained in gravity compensation from the hardcoded perfect square was 2.16 ± 1.10 mm. Figure 5.3 reports the results achieved with the incremental assistance system. The mean distance to the template was 1.96 ± 0.84 mm. The first teaching trial had a mean distance of 3.4 mm. If this result is omitted as an outlier, the mean distance of all results from the template would be 1.80 ± 0.53 mm. Taking the size of the tip of the pen into account and that only the corner points were marked, that is a very close result to what was pro-

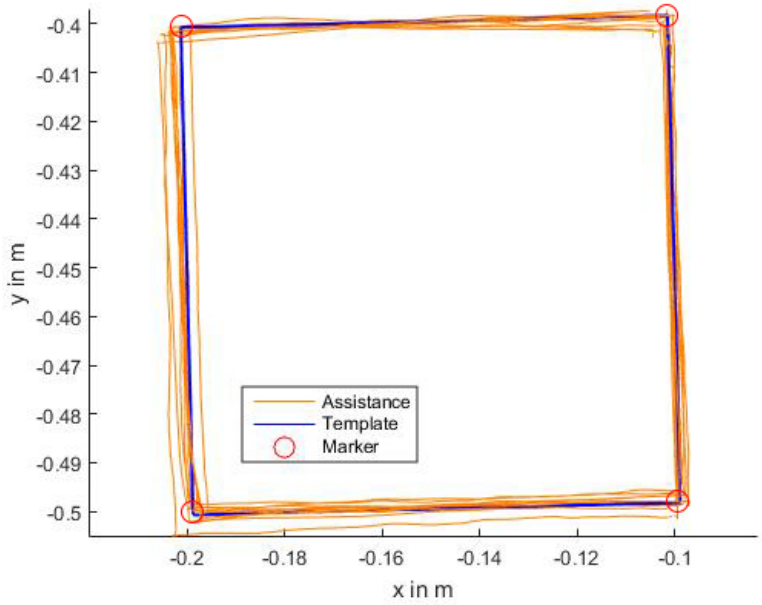

Figure 5.3: The replications of ten teaching trials with the assistance system compared to the template

grammed in KRC. The assistance system shows its strength in comparing the straightness of lines. It is apparent that the squares were much straighter than the results trained in gravity compensation mode. Training with the assistance system results in squares much more similar to the template, while showing straighter lines and more distinct corners.

The impact of the assistance can be seen clearly in Figure 5.4. The initial demonstration is very jerky, barely displaying any straight parts. However, after assisted training, the overall performance has clearly improved. The whole progress of an assisted trial, from the initial demonstration to the result, can be seen in Figure 5.5. Figure 5.5 also visualizes the interaction between human and robot. A general problem with using the force data is that the F/T sensor does not only measure the forces exerted by the user. As the pen is attached to the ATI F/T sensor, forces that occur at the tip of the tool due to friction are measured too. Using the F/T sensor in the wrist is no solution as the readings from that sensor contain also the forces exerted by the robot itself. To give a clearer figure of the interaction between the robot and the human, the force component that is not parallel to the movement of the robot was calculated. This force component better constitutes the correctional forces exerted by the

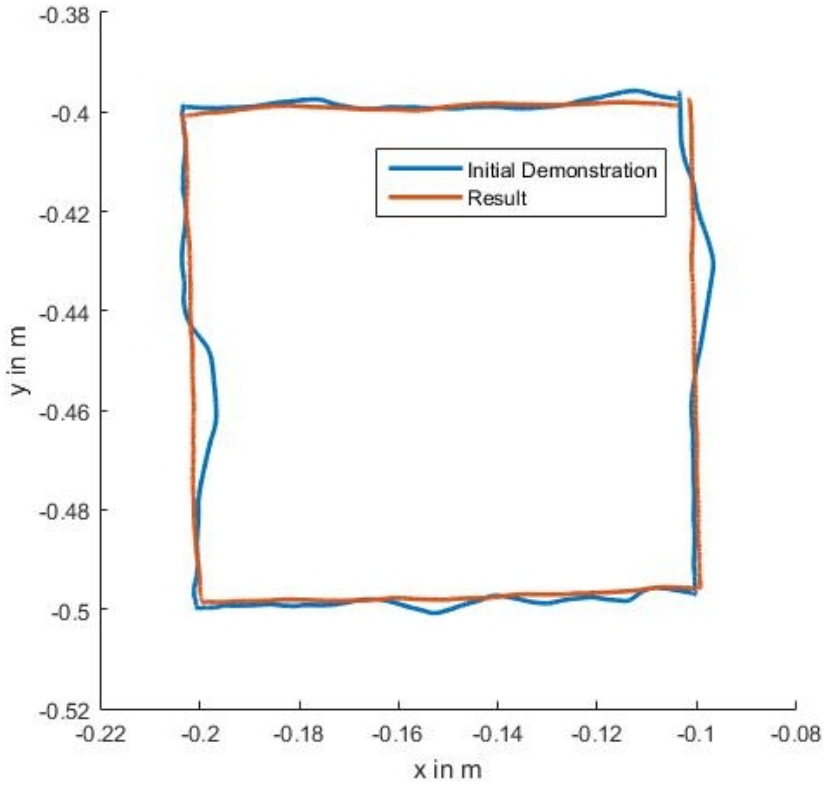

Figure 5.4: Comparison of initial demonstration and result

user. Nevertheless, this solution is not perfect, as it can be seen in the plot of the result. This last plot depicts the replication of the skill without involvement of the user and therefore shows the forces perpendicular to the movement solely exerted by the robot. Despite the mixture of origin for the measured force, the user part can be recognized in higher force readings, represented by longer arrows. It can be seen that the overall amount of exerted forces decreased with progress of the teaching trial and forces were exerted more punctually for correction purposes. This confirmed that the behavior of the user changed from giving the whole demonstration on its own to just guiding the robot and correcting where necessary.

5.1.3 Training the robot to draw a Circle

A further experiment was conducted, comparable in terms of modality and complexity, to study the impact of the assistance system on a relatively difficult task.

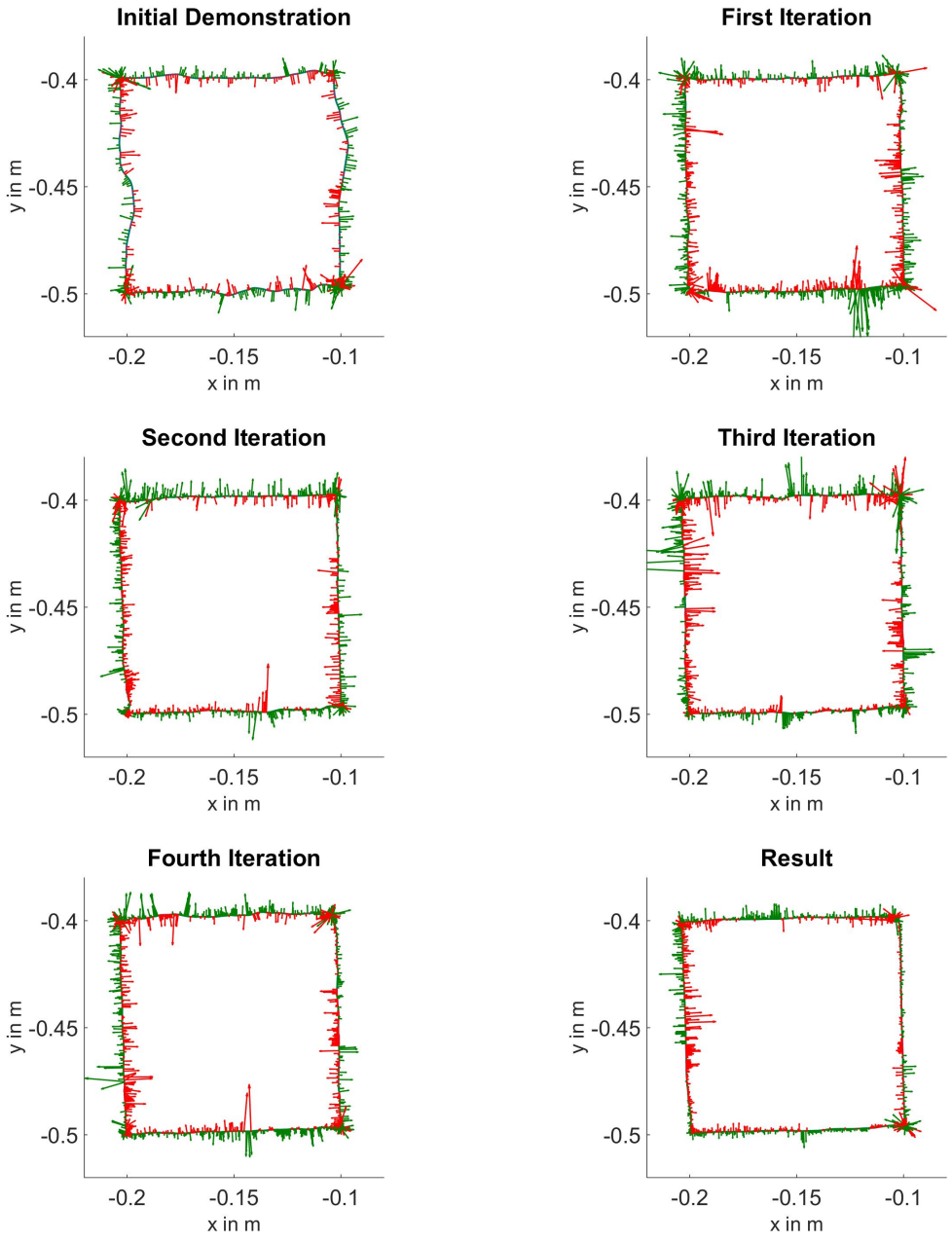

Figure 5.5: Progress of an assisted teaching trial of a square. It can be seen how the trajectory becomes a better square and the correctional forces that has been applied. Correctional forces applied towards the left side of the movement direction of the robot are marked green, forces applied to the right side are marked red.

Experiment

Similar to the previous experiment, a circle was programmed in KRL to have a reference. This reference was used to mark four points on paper, which lay on two perpendicular lines, to denote the dimensions of the circle, which had a radius of 4.9 cm. For better clarity the markers are displayed in Figure 5.6 and 5.7. Then ten circles were trained in gravity compensation mode and afterwards ten teaching trials with the assistance system were performed. The distance measurement was also chosen as a metric for this experiment. Combined with visualizations of the results it gives a good impression of the improvements.

Results

The results of the teaching trials in gravity compensation mode can be seen in Figure 5.6. The task is harder than the previous one, as apparent from the comparison of Figures 5.2 and 5.6. In fact, the shaky curves that barely make their way through the given marks could hardly be called circles. At the uppermost part of the circle the results exhibit a smaller deviation from each other. However, around the remaining marks the drawn curves lay within a range of 1 cm apart. This result can also be expressed in the mean distance from the template, which is 5.68 ± 4.78 mm. The high standard deviation is a particularly eloquent indicator of the inaccuracy of teaching trials in gravity compensation.

Better results were achieved with the aid of the assistance system, as can be seen in Figure 5.7. Although still far from perfect circles, the drawn curves were much closer to the suggested template. The top, bottom and left marks were met fairly close (considering the tip size of the pen). This was also reflected in the mean distance from the template of 3.18 ± 1.91 mm. This shows that with the aid of the assistance the user was able to draw significantly more accurate circles.

An example of the gap between initial demonstration and result, and thus the impact of the assistance system, can be seen in Figure 5.9. The general impression that the use of the assistance system results in rounder circles is affirmed. A whole teaching trial during assisted training, with visualized perpendicular

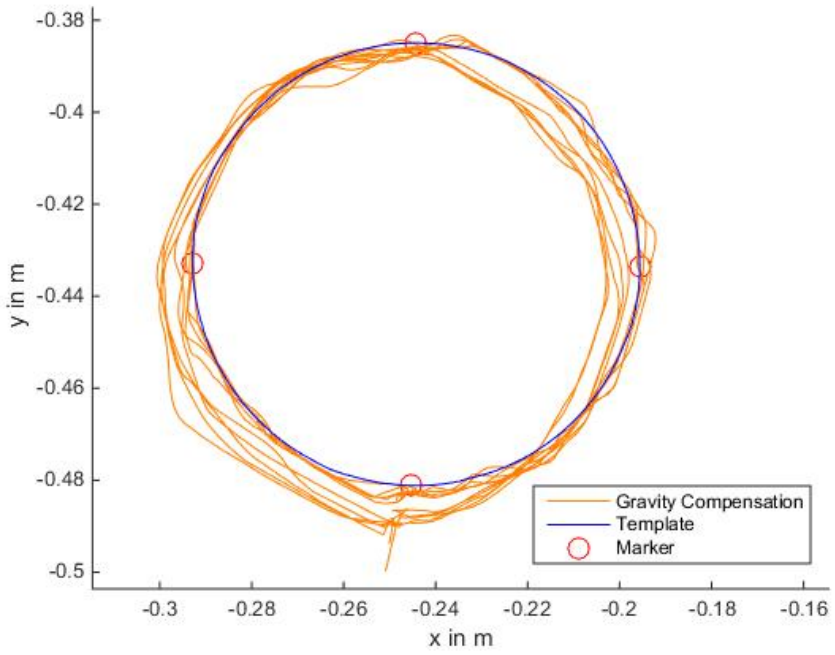

Figure 5.6: The results of the teaching trials in gravity compensation mode compared to the template

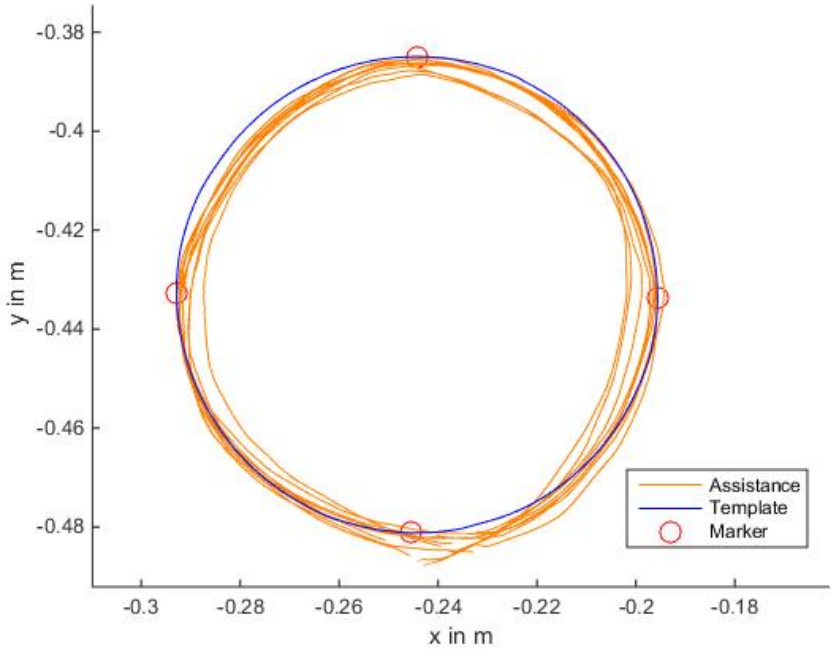

Figure 5.7: The results of the assisted teaching trials compared to the template

forces (similar to the previous case), can be seen in Figure 5.8. After the initial demonstration comparatively strong and consistent correctional forces are applied during the first iteration. During the second iteration correctional forces are mostly applied immediately after the start of the demonstration. In the following demonstrations most of the force was exerted at the end point. The force reading of the result shows the forces measured during reenactment of the learned task by the robot alone, i.e. without any direct interaction with the user.

5.2 Qualitative Evaluation

To evaluate how easy and intuitive the system is to use and to understand how novice users perform with the system, a user study was conducted, inspired by the study performed by Wrede et al. [2013]. Ten participants performed the test in two cases and were asked to take part in a survey afterwards. One participant was excluded from the analysis as a failure occurred during the teaching trial. The remaining participants were aged between 23 to 28, of whom six were males and three females. The origin of the participants is highly diverse with participants from Germany, Netherlands, India, Bangladesh, Spain and China. The exact composition of origin and educational level is reported in Figure 5.10. Seven participants stated they have prior experience with robots, of whom five have experience with robotic arms, one with robotic hands and one with simulated robots. Six participants rated their manual skills as good, while three stated they have bad manual skills.

Five hypotheses were tested:

- The assistance system is easy to use
- The assistance system is not perceived as threatening
- User have a better overall experience with the assistance system
- The robot is easier to handle with the assistance system
- The results are more accurate with the assistance system

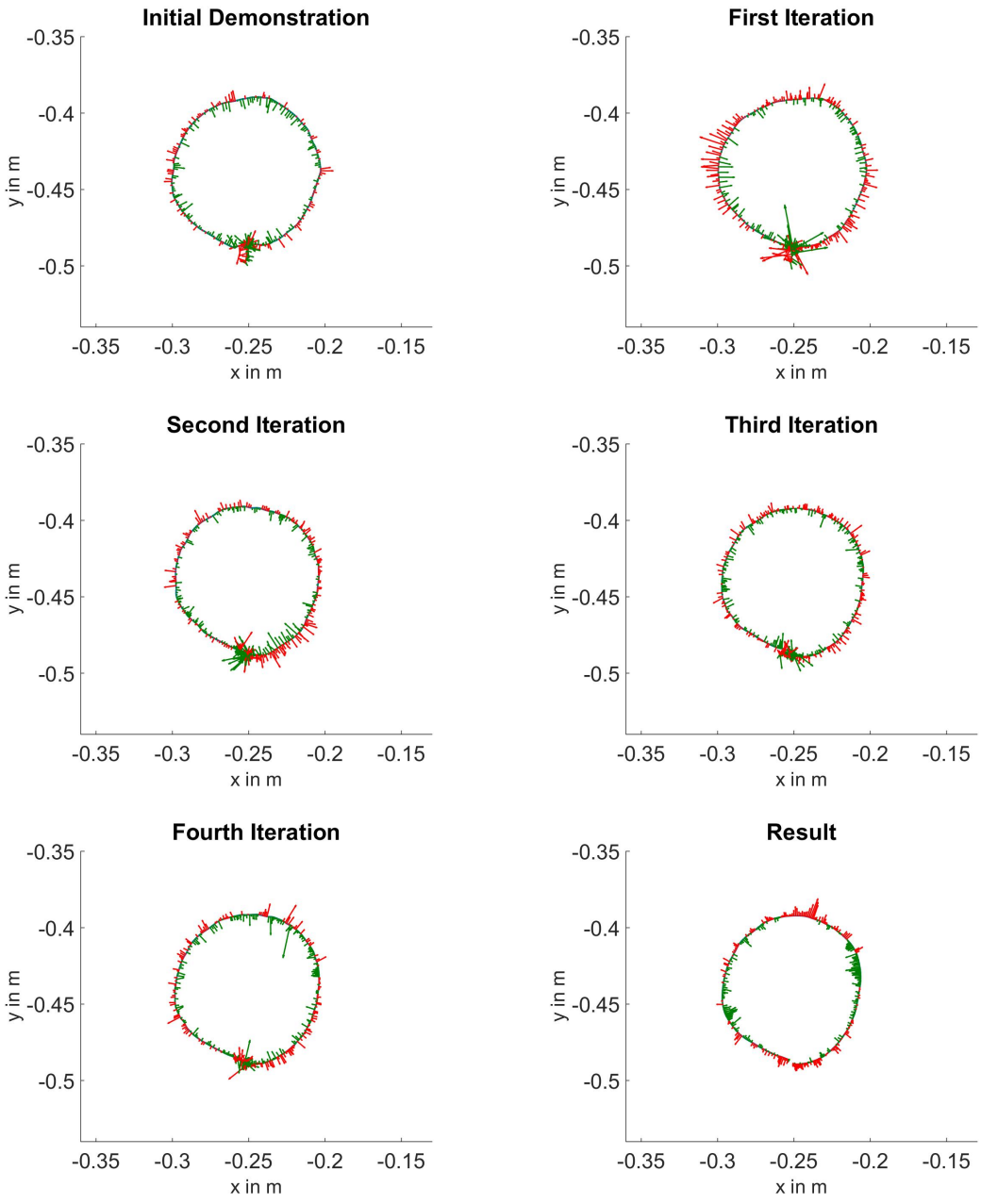

Figure 5.8: Progress of an assited teaching trial of a circle with correctional forces visualized.Correctional forces applied towards the left side of the movement direction of the robot are marked green, forces applied to the right side are marked red.

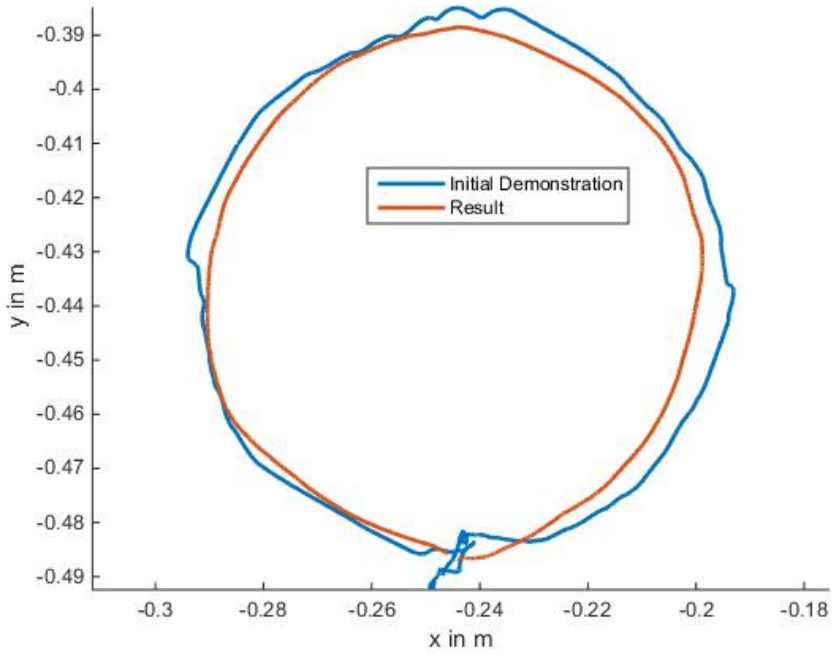

Figure 5.9: Example of the difference between initial demonstration and result of an assisted teaching trial

For comparability two tasks were studied: Case A consisted of training the robot to draw a square in gravity compensation mode and case B in training the same skill with the assistance system.

5.2.1 Setup

The study was divided into four parts. First, both cases were demonstrated to show the participants how to use the robot. In the second part, the participants had the possibility to try both cases on their own. For case A the robot was in gravity compensation mode and participants could move the robot freely. To get acquainted with the gravity compensation the users were asked to draw two squares. For case B the users were asked to draw a square using virtual tool dynamics fully compliant (i.e. the robot actively follows the force applied by the user) and then with a stiffness of 400 $\frac{N}{m}$, so the user acquired a feeling for the system. After this warm-up phase both cases were tested in teaching trials. As all participants were new to the system, learning effects have to be taken into consideration. Therefore half the participants started the teaching trials with case A, the other half with case B. This procedure was necessary to avoid biasing

the results towards the case tested last. For comparability the users were asked to train the robot to draw a square and the corner points of the square were marked on the paper where the participants drew on. For case A the users were asked to train four squares in total in gravity compensation. The user had a free choice where to grab the robot, but all participants grabbed the robot at its wrist (as it was demonstrated in the first part). Then the recorded trajectories were batch learned with the DMP framework and the result shown to the participant. For case B the participant was asked to give an initial demonstration by grabbing the robot at its wrist and then three additional demonstrations with stiffness values of 200 $\frac{N}{m}$, 400 $\frac{N}{m}$ and finally 800 $\frac{N}{m}$. The last three demonstrations were performed by grabbing the tool directly. Then the result from this teaching trial was presented to the user. The last part of the study consisted of a survey, that consisted of questions about the participants, both cases and a general comparison. The questions and the results are described in Section 5.2.2.

5.2.2 Results

The analysis of the results is divided into two parts. In the first part the result of the survey will be analyzed while the second part considers the result of the teaching trials. For most questions, participants were asked to give a rating of 1-5 with 1 as very bad or strong disagreement to 5 as very good or strong agreement. To better compare the results between case A and B a statistical hypothesis test, namely the Wilcoxon Signed-Rank Test [Wilcoxon, 1945], was performed.

Survey

The rating of the simplicity of the system was 4.00±0.87 in case A and 4.33±0.71 in case B with a p-value of 0.5625 as displayed in Table 5.1. The intuitivity was rated with 3.78 ± 0.83 and 3.89 ± 0.93 and has a p-value of 1.0000. This reflects that teaching the robot a new skill is perceived within the participants as a bit simpler with the assistance than without, while there is almost no difference in terms of intuivity. For both ratings no statistical significance can be identified. A greater difference was perceived in the quality of the result. The result with gravity compensation was rated 3.67 ± 1.22, while the result achieved with the

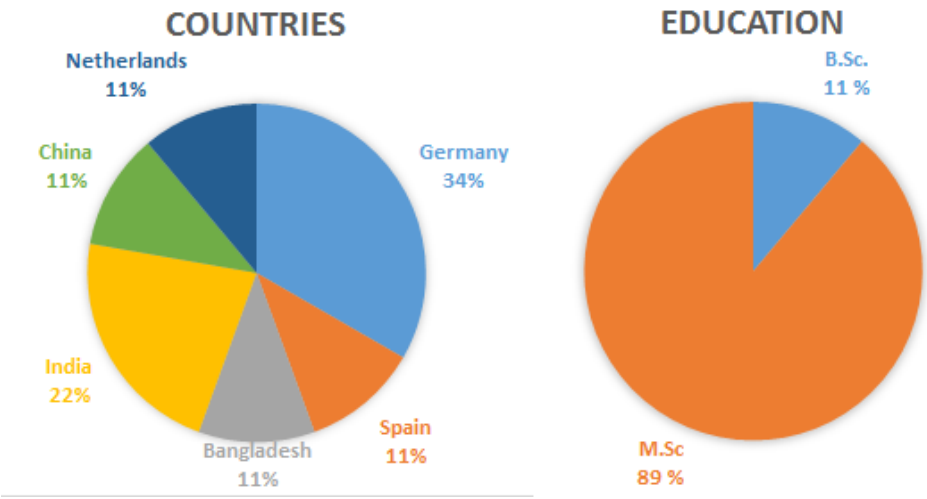

Figure 5.10: Distribution of origin and educational level of participants

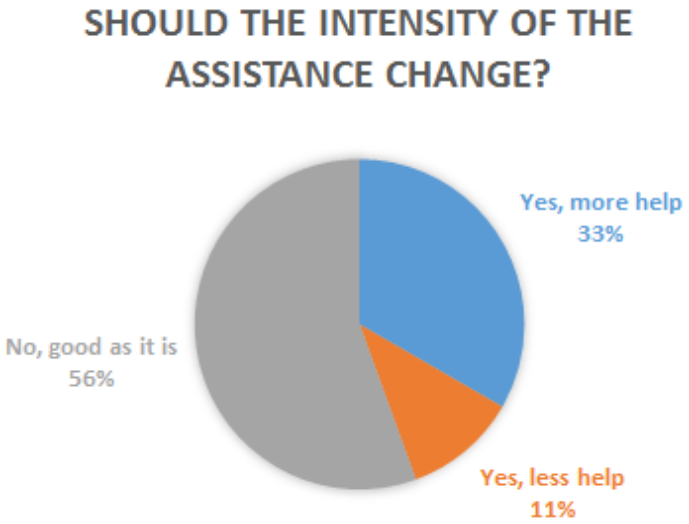

Figure 5.11: Answers for the question whether the intensity of assistance should change

	Case A		Case B		
Question	Mean	SD	Mean	SD	P-value
Simplicity of handling	4.00	0.87	4.33	0.71	0.5625
Did the handling feel intuitive	3.78	0.83	3.89	0.93	1.0000
How would you rate the result?	3.67	1.22	4.22	0.67	0.3125
Was the handling physical demanding?	1.89	1.05	2.11	1.27	0.5000
Was the handling mentally demanding?	1.78	1.30	2.00	1.32	0.7500

Table 5.1: Mean results for the experience in both cases A and B

Question	Mean	SD
How would you rate your manual skills?	3.33	1.00
Was the assistance helpful?	4.56	0.53
Did the interaction with the robot feel awkward/threatening?	1.56	0.53

Table 5.2: Additional statements of the participants

assistance was rated 4.22 ± 0.67. The variance shows that the participants were less certain how to rate the result in case A and also gave it a worse rating. In case B the result got a distinctively better rating with a higher certainty. The difference between the ratings of the result is indicative of an improvement but with a p-value of 0.3125 not statistically significant. Both physical and mental demand were rated a bit higher for the assistance system with 2.11 ± 1.27 and 2.00 ± 1.32 over 1.89 ± 1.05 and 1.78 ± 1.30. Also in these both cases are the difference not statistically significant. In general, the assistance was perceived as very helpful with a rating of 4.56 ± 0.53. 33% of all participants wished the robot would help more, 56% stated that the assistance is good as it is, while 11% would have preferred less help. For better clarity these results are summarized in Table 5.1 and 5.2 and Figure 5.11.

Teaching Trials

The teaching trials of the participants were recorded to be able to evaluate the performance of the system when utilized by novice user. The four demonstrations given without assistance were batch learned and the learned trajectory executed and recorded. These results can be seen in Figure 5.12. Figure 5.13 shows the results which were achieved with the aid of the assistance system. Despite the fact that the trajectories trained in gravity compensation mode were batch learned and thus every plotted result is the mean of four demonstrations, the lines are unsteady and curved. On average the assistance helps the user to give better demonstrations. The results lie closer together and the lines are mostly more straight. This is also reflected in the mean distance of the trained trajectory from the template. Unassisted teaching trials result in a trajectory which has an average distance of 5.09 ± 3.68 mm while the usage of the assistance results in a distance of 2.96 ± 2.50 mm. Altogether the results of case B

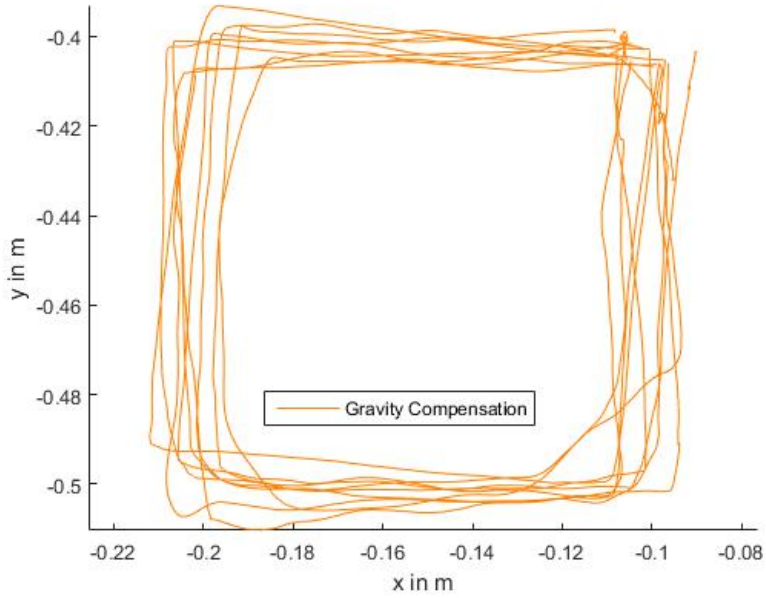

Figure 5.12: Learned trajectories based on the teaching trials of novice users in gravity compensation mode

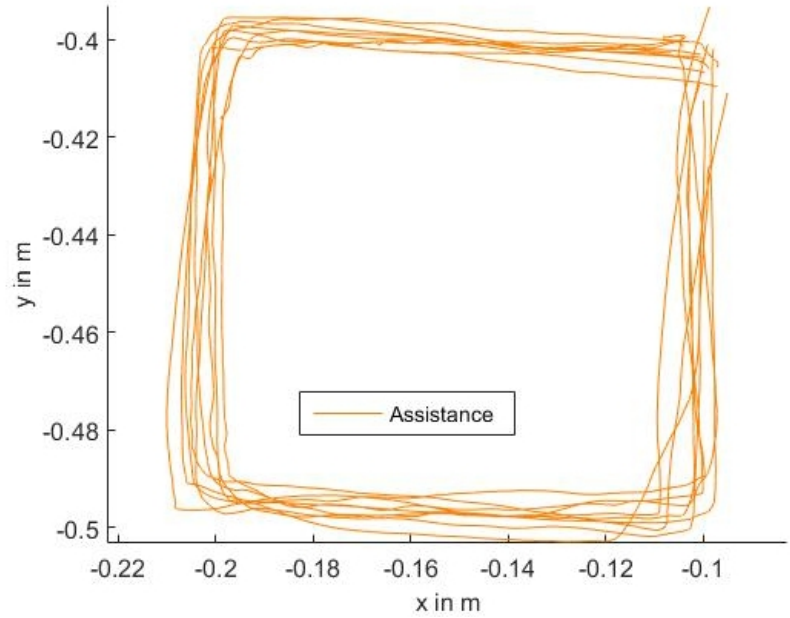

Figure 5.13: Learned trajectories based on the teaching trials of novice users aided with the assistance system

are closer to a square than the results of case A. However, a distinct difference to Figure 5.3 is visible. The teaching trials of Section 5.1 were performed by an experienced user, thus the difference is attributable to a different level of experience with the system. Nevertheless it can be stated that a distinct improvement of demonstrating skills to the robot can be achieved by novice user who use the system for the first time. The results described in Section 5.1.2 suggest that an even greater improvement is possible if the user has more time to get acquainted with the system.

5.3 Discussion

The conducted experiments presented in this chapter show clearly an improvement of programming a skill by demonstration with the assistance system over unassisted teaching trials. The quantitative evaluation shows that an experienced user can achieve results for a drawing task that can be close to the hard-coded template. To explore the interaction between robot and novice users, a user study was performed that showed a distinct improvement in the quality of the learned skill, which can be achieved without prior experience with the system or the particular robot.

Part of the quantitative evaluation was the demonstration of the necessity of an alignment of demonstrated trajectories. However, the utilized DTW algorithm is only designed to work with two trajectories. This yields a problem that emerges when the results of novice user are analyzed. Despite the use of DTW, most results exhibit rounded corners. This is due the fact that trajectory t_1 gets aligned with trajectory t_2 and, after the second iteration, the resulting alignment t_{12} with trajectory t_3. The alignment of t_{12} and t_3 is denoted as t_{23}. Naturally there is a discrepancy between the alignments t_{12} and t_{23}, which is smaller than the discrepancy between t_1 and t_3, but can be big enough to result in a smoothening of sharp changes of direction. The conclusion is that DTW can improve the quality of a trained skill greatly, as shown in Section 5.1.1, but has its limits if demonstrations differ highly in timing.

To illustrate the impact of the assistance system two different primitives were trained by an experienced user with and without the assistance system. It

is apparent that it is hard to train new skills in gravity compensation without further assistance, as the resulting trajectories tend to be unsteady and far away from desired positions. However with the aid of the assistance system results were achieved that are at least better than the results achieved with gravity compensation, for example at teaching the robot to draw circles, or even close to perfect representations of the primitive, in the case of drawing a square.

During the user study five hypotheses were tested. The first three were that the assistance system will be perceived as easy and intuitive to use and that the interaction with the robot will not be perceived as awkward or threatening. The expectations were fulfilled within the participants. The assisted system got a better rating then the unassisted method in terms of simplicity. There was almost no difference between the intuitivity of case A and B, however, both were rated good. Another result is that no participant perceived the system as awkward or threatening. The other two hypotheses were that the users prefer the assisted system and achieve desired trajectories with it. This was confirmed by the better rating for the results of the assistance system and that the assistance system was rated very helpful, as the quality of the results improved noticeably. The perception of the improvement of programming a skill from the quantitative evaluation was reinforced by analyzing the teaching trials of the participants. On average the trajectories trained with the assistance system were more steady and closer to the hardcoded template than the trajectories trained without assistance.

However, the results of the user study have to be seen as indicative, as no comparison between case A and B exhibits statistical significance. Therefore a larger user study needs to be performed to confirm the aforementioned observations. Further improvements have to be done regarding the mental and physical demand. Despite rated as not very mentally or physically demanding, case B got a slightly worse rating than case A.

The last question of the user study was how the user would describe his or her experience with the robot and what should be improved. In general the users reported a positive experience while suggesting the implementation of additional cues during the iterations to indicate the pace of demonstration the robot assumes based on the previous demonstrations. One issue that appeared during the user study is that some users gave an initial demonstration slowly,

but increased the speed of demonstration as moving the robot is easier the more assistance is offered. The stiffness of the robot is defined as $\frac{N}{m}$, meaning it becomes harder to move the robot further away from its desired position. The desired position is calculated from previous demonstrations. The consequence is that users who gave distinctly faster demonstrations than previous demonstrations ended up pressing against a resisting robot. Users more experienced with the system tended to have a better feeling for guiding and giving corrections to the robot. Two participants stated that the robot was at some point hard to move and one of them suggested smaller stiffness values for the assistance. As the tested values were relatively low it might be a good approach to increase the stiffness of the robot in the direction of movement, to make the guidance more obvious, while having less stiffness in transversal direction.

The system is currently limited to use the same orientation that has been set after moving the robot to the start position. For the aforementioned tasks this behavior reduces the mental demand as the user does not have to concentrate on keeping a desired orientation of the tool. Naturally, this is only feasible for tasks that do not require various angles between tool and surface. However, only a factor in the ProactiveAssistance component has to be changed to allow varying orientations during the demonstrations. So far, the system was only tested for drawing tasks on a flat surface. Nevertheless, an extension to allow more sophisticated tasks,e.g. planing, should not require much more than adjusting certain variables. An open question is how the system would perform for tasks that requires a certain timing, e.g. playing an instrument. The current implementation of DTW has a similar execution timing, i.e. which movements are performed at a certain point of time, than the demonstration timing, yet the difference and possible implications have to be tested.

Chapter 6

Conclusions

The goal of this thesis was the creation of a system to assist users in providing better demonstrations of a skill to a robot and thus allowing a user to train a robot in an easy, intuive and flexible way, especially for skills that are hard to program because of contact forces, uncertain and unstructured scenarios and similar reasons. From the analysis of the general framework of PbD, Incremental Learning stood out as a promising method to assist and facilitate the user during demonstrations of the task, being complementary to the natural and intuitive user experience promised by PbD. Incremental Learning is suited for usage in an assistance system, as it is a supervised method that allows a user to improve the trained skill where necessary, while having control over the forces exerted by the robot. Accordingly, the system presented in this thesis was implemented as an incremental approach, extended with virtual tool dynamics, that provided increasing assistance with increased knowledge of the task. Finally, the system is evaluated with respect to its performance, simplicity, intuitivity and usability by novice users. The experiments showed a distinct improvement of the primitives learned from assisted teaching trials over unassisted demonstrations for experienced users. Furthermore the user study revealed that the system is easy and intuitive to use and can also improve the demonstrations of novice users greatly.

The work of this thesis is based on the infrastructure implemented by Steinmetz [2013]. The contribution of this work is to extend the existing infrastructure to an assistance system. The implemented system is based on two main ideas with the aim to assist a user in giving better demonstrations. The first idea

was to reduce the effective inertia perceived by the user, making the robot easier to move. This was achieved by implementing virtual tool dynamics. The second idea was to utilize incremental learning to enable the user to physically guide the robot during the execution of tasks. This method allows to teach natural movements. Additional to assisting the user the system improves the quality of learned trajectories by using Dynamic Time Warping. Various paces during demonstrations result in a primitive that can be rather different from the intended primitive, e.g. distinctive elements like sharp angles are smoothed. Dynamic Time Warping was implemented to cope with this problem by aligning the demonstrated trajectories thus improving the quality of the learned skill.

The experiments reported in this thesis analyze both quantitatively and qualitatively the implemented approach. The quantitative evaluation examined the general performance of the assistance system. It was shown that the assistance system has a huge impact on the quality of the learned skills. The assistance system can help the user to train the robot in a way that the accuracy can be satisfactory once compared to results achieved by hardcoding the robot. The strength of the assistance system is twofold. It can be utilized by users without programming knowledge and assisted teaching trials are conducted more natural, simpler, faster and more flexibly than hardcoding the robot, even for comparably simple tasks like drawing basic geometric figures. Also teaching trials can be used to program a robot in cases not enough information is available for hardcoding, e.g. in in-contact tasks. The qualitative evaluation was performed in form of a user study to show that no knowledge of the particular robot or experience with the assistance system was necessary to achieve a swift and obvious improvement in the quality of the trained skill. The assistance system was implemented with the demand to be easy and intuitive to use, without being perceived as threatening. A further goal is to increase the accuracy of the training phase with respect to unassisted teaching trials. All these demands are positively addressed. However, a comparison of the results between an experienced user and novice users show that the system can be utilized more effective by experienced users. Nevertheless, it was shown that even just a few assisted teaching trials are sufficient to novel users to build a good feeling with the robot and remarkably improve the overall performance.

Not much research has been performed earlier to examine the possibilities of incremental learning in combination with kinesthetically teaching with state of

the art robots like the KUKA LWR4+. The technical literature reports two approaches similar to the one described in this work. Saveriano et al. [2015] utilized incremental learning in combination with a LWR4+ to kinesthetically teach trajectories and null-space motions. In their work the authors use incremental learning to add several subtasks to a main task, e.g. avoiding obstacles while following a trajectory, and to refine the learned end-effector trajectory. In abstract terms, the multi-priority kinematic controller implemented in the approach by Saveriano et al. [2015] is more sophisticated than the implemented AssistanceControl in this work, as it assigns priorities to corrections exerted by the user and groups these corrections into subtasks. The work by Saveriano et al. [2015] confirms the finding of this thesis that physically guiding the robot during execution can result in more natural movements. An earlier approach was described by Wrede et al. [2013], where a neural network was first trained to avoid obstacles and then, in a second teaching trial, to perform a certain task. This approach is not an Incremental Learning approach, as the underlying model is not updated. Thus it is not possible to improve the quality of the trained skill further after one demonstration. But similar to this work, the approach aims to reduce the mental demand on the user and helps the user to provide better demonstrations, as only the end-effector position has to be considered and not the whole joint configuration of the robotic arm. A user study, comparable to the one presented in this thesis, was performed to show how the system was perceived by novice users. This study had 49 participants, split into two groups, allowing the authors to extract useful information with the analysis of variance. Wrede et al. [2013] showed that an assistance system can have a positive effect on the usability and performance of kinesthetically teaching skills on redundant robots, which was also shown in this thesis.

6.1 Future Work

Several aspects need to be improved in order to seamlessly integrate the system in a work environment.

One of these aspects consists on providing additional cues to the user in order to make the currently learned trajectory visible. This visualization could be done by projecting the trajectory onto the surface. Another aspect would

be the different pace of demonstrations. For every point in time during the demonstration, cues could be provided that denote the position of the tool at the same time in the previous demonstration. This could help novice users to utilize the system more efficiently as discussed at the end of Chapter 5. The user friendliness should also be improved by implementing a graphical user interface, which gives clear instructions to the user and allows an easy selection of different modes and saving and reloading learned skills. All the necessary functionalities have already been implemented, but they currently have to be controlled in a terminal.

So far the system has been tested on relatively simple tasks and it has yet to be shown that the system can also be generalized to more complex tasks. As the system is implemented to transfer skills from an expert in his field to the robot, future work would consist of testing the system with skilled participants for complex tasks and comparing the results of the teaching trials with results manually achieved, without the robot involved. It would be interesting to explore whether and how much the process of transferring complex skills from an expert to the robot can be improved by the use of the assistance system.

The results of the performed user study indicated an improvement of several attributes of the assisted system over unassisted teaching trials. However, a larger pool of participants is required to show that the reported observations are not only valid within the participants of this study, but are statistical significant. Furthermore could the assessment of additional data, e.g. time required to teach skills, provide a better insight in the advantages and disadvantages of the assistance system. An interesting aspect that could be investigated is the performance development of the participants, i.e. how the quality of the demonstrations is dependent on the experience with the assistance system.

Another possible area of future research would be to develop a method that extends Dynamic Time Warping to simultaneously time align several trajectories with each other. This would further increase the reproduction quality of skills trained with different paces, as the discrepancies between several alignments, which are discussed in Chapter 5, would vanish and reduce smoothing of the learned primitives.

References

André, J., Teixeira, C., Santos, C., and Costa, L. (2015). Adapting biped locomotion to sloped environments. *Journal of Intelligent & Robotic Systems*, pages 1–16.

Argall, B., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 67:469–483.

ATI (2015). ATI Force / Torque Sensor: Mini45. `http://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini45`, referenced on 24.06.2015.

Atkeson, C., Moore, A., and Schaal, S. (1997). Locally weighted learning for control. *Artificial Intelligence Review*, 11(1-5):75–113.

Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Survey: Robot Programming by Demonstration. In *Handbook of Robotics*, volume chapter 59. MIT Press. Book winner of 2 PROSE awards (Award for Excellence in Physical Sciences & Mathematics, Award for Engineering & Technology).

Bischoff, R., Kurth, J., Schreiber, G., Koeppe, R., Albu-Schaeffer, A., Beyer, A., Eiberger, O., Haddadin, S., Stemmer, A., Grunwald, G., and Hirzinger, G. (2010). The kuka-dlr lightweight robot arm - a new reference platform for robotics research and manufacturing. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–8.

Calinon, S. and Billard, A. (2007). Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 255–262.

Calinon, S. and Billard, A. (2008). A framework integrating statistical and social cues to teach a humanoid robot new skills. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Workshop on Social Interaction with Intelligent Indoor Robots*.

Calinon, S., Bruno, D., Malekzadeh, M. S., Nanayakkara, T., and Caldwell, D. G. (2014). Human-robot skills transfer interfaces for a flexible surgical robot. *Computer Methods and Programs in Biomedicine*. In press.

Calinon, S., D'Halluin, F., Sauser, E., Caldwell, D., and Billard, A. (2010a). Learning and reproduction of gestures by imitation. *Robotics Automation Magazine, IEEE*, 17(2):44–54.

Calinon, S., Guenter, F., and Billard, A. (2007). On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, 37(2):286–298.

Calinon, S., Sardellitti, I., and Caldwell, D. (2010b). Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 249–254.

Cederborg, T., Li, M., Baranes, A., and Oudeyer, P. (2010). Incremental Local Online Gaussian Mixture Regression for Imitation Learning of Multiple Tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, Taipei, Taiwan.

Cho, S. and Jo, S. (2013). Incremental online learning of robot behaviors from selected multiple kinesthetic teaching trials. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 43(3):730–740.

Cohn, D., Ghahramani, Z., and Jordan, M. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

Dawood, F. and Loo, C. K. (2014). Autonomous motion primitive segmentation of actions for incremental imitative learning of humanoid. In *Robotic Intelligence In Informationally Structured Space (RiiSS), 2014 IEEE Symposium on*, pages 1–8.

De Luca, A., Albu-Schaffer, A., Haddadin, S., and Hirzinger, G. (2006). Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1623–1630.

Ekvall, S. and Kragic, D. (2006). Learning task models from multiple human demonstrations. In *Robot and Human Interactive Communication, 2006. RO-MAN 2006. The 15th IEEE International Symposium on*, pages 358–363.

El-Yacoubi, A., Gilloux, M., Sabourin, R., and Suen, C. (1999). Unconstrained handwritten word recognition using hidden markov models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 21, page 752–760.

Franklin, J. (1988). Refinement of robot motor skills through reinforcement learning. In *Proceedings of the 27th IEEE Conference on Decision and Control; Austin, TX, USA; ; 7 December 1988 through 9 December 1988*.

Friedman, J. H. (1995). Intelligent local learning for prediction in high dimensions. In *International Conference on Artificial Neural Networks (ICANN 95)*.

Ghahramani, Z. (2002). Hidden markov models. chapter An Introduction to Hidden Markov Models and Bayesian Networks, pages 9–42. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Goebel, P. (2013). *ROS By Example*.

Gräve, K., Stückler, J., and Behnke, S. (2010). Learning motion skills from expert demonstrations and own experience using gaussian process regression. In *ISR/ROBOTIK*, pages 1–8. VDE Verlag.

Guenter, F., Hersch, M., Calinon, S., and Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *RSJ Advanced Robotics*, pages 1521–1544.

IFR (2014). Statistical Department, World Robotics, Executive Summary WR 2014.

Ijspeert, A., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Comput.*, 25(2):328–373.

Ijspeert, A., Nakanishi, J., and Schaal, S. (2002a). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1398–1403.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002b). Learning rhythmic movements by demonstration using nonlinear oscillators. In *In Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2002*, pages 958–963.

Inamura, T., Toshima, I., Tanie, H., and Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *I. J. Robotic Res.*, 23(4-5):363–377.

Jain, R. and Inamura, T. (2014). Learning of usage of tools based on interaction between humans and robots. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*, pages 597–602.

Khreich, W., Granger, E., Miri, A., and Sabourin, R. (2012). A survey of techniques for incremental learning of hmm parameters. *Inf. Sci.*, 197:105–130.

Kober, J., Bagnell, D., and Peters, J. (2013). Reinforcement learning in robotics: A survey. (11):1238–1274.

Kormushev, P., Calinon, S., and Caldwell, D. G. (2011). Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603.

Kormushev, P., Calinon, S., and Caldwell, D. G. (2013). Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148.

Kristan, M., Skocaj, D., and Leonardis, A. (2008). Incremental learning with Gaussian mixture models. In *Computer Vision Winter Workshop 2008*, pages 25–32. Slovenian Pattern Recognition Society, Ljubljana, Slovenia.

Kronander, K. and Billard, A. (2012). Online learning of varying stiffness through physical human-robot interaction. In *ICRA*, pages 1842–1849. IEEE.

Kulic, D., Lee, D., Ott, C., and Nakamura, Y. (2008). Incremental learning of full body motion primitives for humanoid robots. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 326–332.

Lee, D. and Ott, C. (2011). Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, 31(2-3):115–131.

Malekzadeh, M., Bruno, D., Calinon, S., Nanayakkara, T., and Caldwell, D. (2013). Skills transfer across dissimilar robots by learning context-dependent rewards. In *IROS*, pages 1746–1751. IEEE.

Mühlig, M., Gienger, M., Hellbach, S., Steil, J., and Goerick, C. (2009). Task-level imitation learning using variance-based movement optimization. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 1177–1184.

Müller, M. (2007). Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg.

Montebelli, A., Steinmetz, F., and V., K. (2015). On handing down our tools to robots: Single-phase kinesthetic teaching for dynamic in-contact tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. In press.

Nemec, B., Forte, D., Vuga, R., Tamosiunaite, M., Worgotter, F., and Ude, A. (2012). Applying statistical generalization to determine search direction for reinforcement learning of movement primitives. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 65–70.

Nicolescu, M. and Mataric, M. (2003). Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 241–248, New York, NY, USA. ACM.

Park, C.-B. and Lee, S.-W. (2011). Real-time 3d pointing gesture recognition for mobile robots with cascade hmm and particle filter. *Image Vision Comput.*, 29(1):51–63.

Pastor, P., Kalakrishnan, M., Meier, F., Stulp, F., Buchli, J., Theodorou, E., and Schaal, S. (2012). From dynamic movement primitives to associative skill memories.

Peters, J., Vijayakumar, S., and Schaal, S. (2003). Reinforcement learning for humanoid robotics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids2003)*.

Rabiner, K. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE*, pages 257–286.

Sakato, T., Ozeki, M., and Oka, N. (2014). Learning through imitation and reinforcement learning: Toward the acquisition of painting motions. In *2014 IIAI 3rd International Conference on Advanced Applied Informatics*.

Saveriano, M., An, S., and Lee, D. (2015). Incremental kinesthetic teaching of end-effector and null-space motion primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Schaal, S., Mohajerian, P., and Ijspeert, A. (2007). Dynamics systems vs. optimal control — a unifying view. In Paul Cisek, T. D. and Kalaska, J. F., editors, *Computational Neuroscience: Theoretical Insights into Brain Function*, volume 165 of *Progress in Brain Research*, pages 425 – 445. Elsevier.

Schaal, S., Peters, J., Nakanishi, J., and Ijspeert, A. (2004). Learning movement primitives. In *International Symposium on Robotics Research (ISRR2003*. Springer.

Schmidts, A. M., Dongheui, L., and Peer, A. (2011). Imitation learning of human grasping skills from motion and force data. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1002–1007.

Schreiber, G., Stemmer, A., and Bischoff, R. (2010). The Fast Research Interface for the KUKA Lightweight Robot. In *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010)*.

Soetens, P. (2012). *The Orocos Component Builder's Manual Open RObot COntrol Software 2.6.0*.

Steinmetz, F. (2013). *Programming by Demonstration for in-contact tasks using Dynamic Movement Primitives*. PhD thesis, School of Electrical Engineering, Aalto University.

Steinmetz, F., Montebelli, A., and Kyrki, V. (2015). Simultaneous kinesthetic teaching of positional and force requirements for sequential in-contact tasks. Submitted to IEEE-RAS Humanoids 2015.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. The MIT Press.

Vijayakumar, S., D'Souza, A., and Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634.

Warrender, C., Forrest, S., and Pearlmutter, B. (1999). Detecting intrusions using system calls: alternative data models. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, page 133–145.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Wrede, S., Emmerich, C., Grünberg, R., Nordmann, A., Swadzba, A., and Steil, J. (2013). A user study on kinesthetic teaching of redundant robots in task and configuration space. *Journal of Human-Robot Interaction*, 2:56–81.

Wu, Y., Su, Y., and Demiris, Y. (2014). A morphable template framework for robot learning by demonstration: Integrating one-shot and incremental learning approaches. *Robotics and Autonomous Systems*.

Appendix A

Dynamic Time Warping

```
Input: trajectory s,t
Output: DTW
DTW = array(N,M);
DTW[0,0]=0;
for i = 0 to N do
    for j = 0 to M do
        minimum = inf;
        dist = |(s_x − t_x)| + |(s_y − t_y)| + |(s_z − t_z)|;
        if i>0 then
            minimum = DTW[i-1, j];
        if j>0 then
            minimum = min(minimum,DTW[i-1,j-1]);
        if i==0 && j==0 then
            DTW[i,j] = dist;
        else
            DTW[i,j] = minimum + dist;
        end
    end
end
```

Algorithm 1: Calculation of the distance matrix

```
Input: DTW
Output: WarpPath
vector<x,y> WarpPath;
i = N-1;
j = M-1;
while true do
    if i==0 && j==0 then
        break;
    if i==0 then
        j–;
    else
        if j==0 then
            i–;
        else
            min = inf;
            if DTW[i-1][j]<min then
                min = DTW[i-1][j];
                i–;
            if DTW[i][j-1]<min then
                min = matrix[i][j-1];
                j–;
            if DTW[i-1][j-1]<min then
                i–;
                j–;
        end
    end
    WarpPath.add(i,j);
end
```

Algorithm 2: Calculation of the warppath

```
Input: s,t,WarpPath
Output: s_dtw,t_dtw
for i = 0 to WarpPath.length do
    s_dtw[i] = s[WarpPath[i].x];
    t_dtw[i] = t[WarpPath[i].y];
end
```

Algorithm 3: Warping the trajectories