Efficient and Intuitive Teaching of Redundant Robots in Task and Configuration Space

by **Christian Emmerich**

Ph.D. Thesis
Faculty of Technology, Bielefeld University
Juli 2015

Printed on permanent paper according to ISO 9706.
Gedruckt auf altersbeständigem Papier nach ISO 9706.

Acknowledgement

Over the recent years I have received constant support and encouragement from a great number of people to whom I am greatly indebted.

First, I would like to thank my supervisor and mentor Prof. Jochen J. Steil without whose guidance and patience this dissertation would not have been possible. His commitment to machine learning and cognitive robotics has been a major driving force during my research, and his valuable advice always helped me to not get stuck in a local optimum of my learning curve.

Although a dissertation is an individual work, I believe that research in robotics is teamwork; and I had the great opportunity to be part of the team CoR-Lab. I want to thank the reservoir group for many, many fruitful discussions and valuable feedback about my thoughts and theories, particularly Felix Reinhart and Klaus Neumann with whom I had the pleasure to jointly gain insight in the essentials of neural networks. I also would like to thank all members of the cognitive systems engineering group for their great support and shared knowledge. I have learned a lot about sustainable and robust software engineering from you. In addition, as part of the team FlexIRob@Harting I also appreciated the collaboration with Stefan Krüger, Arne Nordmann, Sebastian Wrede, Ricarda Wullenkord and Agnes Swadzba. It was a pleasure to work with you. Furthermore, I want to thank the members of the secretary's office, especially Anke Kloock, Heike Leifhelm and Ruth Moradbakhti, who have always been there for me in case of urgent needs and took care about the administrative aspects of my Ph. D.

Doing a Ph. D. is an intense and rewarding journey which turns out to be an invaluable experience when being shared with colleagues that become friends. I had the luck to walk this way with Andre Lemme, Arne Nordmann and Stefan Rüther with whom I share many great memories - in our offices during joyful and inspirational discussions, in the lab during endless coding sessions, and during unforgettable vacations. Thank you for everything.

Last but far from least, I want to deeply thank my family and my friends for their emotional support, particularly my love and solid rock Saskia for being always by my side.

Abstract

A major goal of current robotics research is to enable robots to become co-workers that learn from and collaborate with humans efficiently. This is of particular interest for small and medium-sized enterprises where small batch sizes and frequent changes in production needs demand a high flexibility in the manufacturing processes. A commonly adopted approach to accomplish this goal is the utilization of recently developed lightweight, compliant and *kinematically redundant robot platforms* in combination with state-of-the-art human-robot interfaces.

However, the increased complexity of these robots is not well reflected in most interfaces as the work at hand points out. Plain *kinesthetic teaching*, a typical attempt to enable lay users programming a robot by physically guiding it through a motion demonstration, not only imposes *high cognitive load* on the tutor, particularly in the presence of strong environmental constraints. It also neglects the possible reuse of (task-independent) constraints on the *redundancy resolution* as these have to be demonstrated repeatedly or are *modeled explicitly* reducing the efficiency of these methods when targeted at non-expert users.

In contrast, this thesis promotes a different view investigating human-robot interaction schemes not only from the learner's but also from the tutor's perspective. A *two-staged interaction structure* is proposed that enables lay users to transfer their *implicit knowledge* about task and environmental constraints *incrementally* and *independently of each other* to the robot, and to reuse this knowledge by means of *assisted programming* controllers. In addition, a path planning approach is derived by properly exploiting the knowledge transfer enabling *autonomous navigation* in a possibly confined workspace without any cameras or other external sensors. All derived concept are *implemented and evaluated* thoroughly on a system prototype utilizing the 7-DoF KUKA Lightweight Robot IV. Results of a large *user study* conducted in the context of this thesis attest the staged interaction to *reduce the complexity* of teaching redundant robots and show that *teaching redundancy resolutions is feasible* also for non-expert users.

Utilizing properly *tailored machine learning algorithms* the proposed approach is completely data-driven. Hence, despite a required forward kinematic mapping of the manipulator the entire approach is *model-free* allowing to implement the derived concepts on a variety of currently available robot platforms.

Contents

List of Figures

Chapter 1

Introduction

> "The KEY characteristic of robots is versatility; they can be applied to a large variety of tasks without significant redesign. This versatility derives from the generality of the robot's physical structure and control, but it can be exploited only if the robot can be programmed easily. In some cases, the lack of adequate programming tools can make some tasks impossible to perform. In other cases, the cost of programming may be a significant fraction of the total cost of an application. For these reasons, robot programming systems play a crucial role in robot development." [1]

1.1 Motivation: Flexible Production in Industry 4.0

With the introduction of the first industrial robot, namely the Unimation Unimate, to automation and industrial manufacturing in the early 1960s a new era in automated production began. As part of the third industrial revolution, robots increasingly started to take over simple manufacturing tasks that they could perform faster, better or for a longer period of time as well as tasks that were harmful, physically strenuous or dull for humans. Such tasks include welding, varnishing, or grinding of workpieces, simple automated assembly processes as well as automated placement and transportation. As a result of this developments, some branches in today's industry such as the automotive industry are highly automated with production lines equipped with hundreds of robots and only few humans to maintaining and supervising them. Once engineered, installed and programmed for a specific task these automated production lines run 24/7 for several months if market conditions demand and supply of resources permits. By this means producers currently reach automation rate of up to 95 percent in some subsections of the production line.

While being highly efficient and cost-minimal when running for a long time, this concept of highly specialized production processes becomes infeasible in changing markets and for small and medium-sized enterprises, where small batch sizes and frequent changes in production needs demand a high flexibility in the manufacturing processes. This flexibility is an important aspect from the economics point of view reducing inventory costs and minimizing the implementation time for new products. Various types and definitions of flexibility are intensively discussed

since more than 30 years [2]. *Machine flexibility* and *material handling flexibility* are regarded as key types of flexibility referring to the various types of operations that a robot can perform and its ability to move different part types efficiently for proper positioning and processing without a prohibitive effort for switching between different operations. Both directly relate to the robotic component systems of a manufacturing system. Hence, the idea of flexibility in manufacturing is by far not a new concept and multi-purpose, multi-axis robots with automatic tool changers and easy programming interfaces are envisioned as means to meet these requirements since then. Still, the topic of flexible production systems is an ongoing topic also in contemporary academic, political and economical efforts as part of the so-called *fourth industrial revolution* or *Industrie 4.0*[1] in which modular systems that can be adapted to changing requirements by means of intuitive human-machine interfaces are considered as a core design concept [3, 4].

Fortunately, various recent developments in robotics hardware allow a new kind of collaborative robots to made their way to industry. In contrast to traditional robots being big, strong and robust, they are compact and lightweight while allowing dexterous motions and physical interaction with humans. Advanced force-torque sensing integrated into compact actuation units [5, 6] with variable stiffness [7] has led to the development of compliant force-controlled robot manipulators such as the KUKA Lightweight Robot IV [8] shown in Fig. 1.1. Also new developments in the design and production of passively compliant robots such as Festo's Bionic Handling Assistant [9] allow for safe direct physical human robot interaction. The importance of these advanced sensing and interaction capabilities is two-fold. On the one hand, they serve as safety features allowing robots to be implemented flexibly alongside and in close collaboration with humans without any need for safety fences, guards or mutually exclusive safety regions. On the other hand, they offer an intuitive interaction interface for humans to program and adapt robots to frequently changing tasks or environments which has been the rationale

Fig. 1.1: Close physical interaction between human and the force-controlled KUKA Lightweight Robot IV.

[1] Although the initiative "Industrie 4.0" is not well-known outside of German-speaking areas, similar concepts such as "Industrial Internet", "Advanced Manufacturing" or "Smart Industry" can be identified in other areas [3].

of teach-in procedures for traditional robots since long. The process of a human tutor physically guiding a robot manipulator through a motion is termed *kinesthetic teaching* [10] and is considered as a key concept for intuitive programming-by-demonstration approaches [11]. Complementary, there is a tendency to increase the number of degrees of freedom (DoF) towards kinematically *redundant manipulators* [12], i.e. robots with more joints than actually needed to solve a specific task. "It is not difficult to discover that a six-degree-of-freedom geometry can no longer be considered a general purpose manipulator."[13]. These systems with seven DoF or even more provide a great degree of flexibility for the realization of a variety of tasks and complex applications. Hence, in order to utilize robots as general-purpose tools rather than as engineered for a specific task, redundancy is a key ingredient for flexibility as already shown industrial scenarios [14] but also concerning service robotics [15, 16]. As a result many collaborative robots (Universal Robots UR5) with 7-DoF kinematics (KUKA LWR iiwa, Rethink Robotics Sawyer), multi-arm solutions (ABB YuMi, Rethink Robotics Baxter) or even mobile platforms (Bosch APAS, Kawada Industries NEXTAGE) are ready-made products for application in manufacturing industry [17].

While each of the aforementioned improvements undoubtedly increase the machine flexibility as discussed above, in practice some of them seem contradictory. A machine flexibility enthusiast would say, the more degrees of freedom a robot employs the better. But in the context of today's manufacturing systems also higher-level flexibility requirements play an important role, which relate to the variety of products that a system can produce *without major setups* and *the ease* to add new products [2]. Setting up the system according to new tasks or constraints must not involve inordinate amounts of time and cost. However, the gained dexterity of redundant robots requires additional modeling steps, i.e. the definition of explicit criteria for redundancy resolution e.g. for obstacle avoidance in cluttered environments or confined workspaces. These criteria are typically not easily accessible to process experts and even less to naive users.[2] Hence, programming experts or roboticists are required for adaptation to new tasks resulting in high costs and long setup times limiting the wide-spread application of these systems. As a result, facilitating efficient (re-)configuration of advanced robot systems towards (new) tasks or environments is still regarded as a major challenge of current robotics research [18]. Exploitation of their gained flexibility in industrial and service applications also for small and medium-sized enterprises calls for improved, intuitive programming methods feasible also for non-experts.

[2] Throughout this thesis the terms "naive" or "lay" users and "non-experts" are used interchangeably and relate to users that might be domain or process experts but - in contrast to programming experts or roboticists - typically have no background in modeling and programming complex robot motions.

1.2 Problem Statement: Teaching of Redundant Robots is Difficult

Consider a realistic scenario for instance in production and assistance at manual workplaces, where non-dynamic task space constraints will typically be present in the form of fixed obstacles in the workspace, in form of a wall, a ceiling or a restrictions not to reach into forbidden regions. And these constraints may change from time-to-time and workplace to workplace which requires to reprogram new tasks, redundancy resolutions and even path planning of an assistive robot. As stated above, the usual way of addressing this problem is to exploit the compliance features of the robot manipulators to facilitate close physical Human-Robot Interaction [19, 10] (pHRI). Utilizing this interaction interface and applying basic programming-by-demonstration methods, non-experts are encouraged to kinesthetically teach-in new tasks. The underlying assumption is that they implicitly model the required criteria according to *environmental constraints* during the teach-in of a task space trajectory. However, these approaches to kinesthetic teaching do not reflect the standard technical approaches to redundancy control, which typically separate task-space planning and constraint resolution in configuration space. As I will discuss systematically in Chap. 2 most of these programming-by-demonstration approaches rely on trajectory-based teaching, either in the high-dimensional configuration space (joint space) of the robot or in the task-space only but then taking redundancy resolution for granted.

The problem addressed in this thesis is that, while standard control for redundant robots clearly distinguishes task and configuration space for redundancy resolution, when utilizing kinesthetic teaching users typically have to deal with the complexity of *simultaneously* considering the redundancy resolution in *configuration space* and the specification of trajectories in *task space*. Furthermore, dedicated interaction support for kinesthetic teaching of constraints in *confined spaces* has not yet been considered. Kaber and Riley demonstrated in [20] for a teleoperation task "that operator performance and workload are significantly affected by whether joint or world mode (i.e. end-effector position) control is required [... and that] for example, world mode can reduce task completion times, but may also increase the number of contact errors when working in confined spaces" [21]. I hypothesize that, analogously to teleoperation, kinesthetic teaching of a redundant robot arm in joint control mode, i.e. the robot is freely movable in all joints, is a difficult task, particularly for non-expert users. The lack of separation of task-space programming and redundancy resolution causes a *high complexity*, unsuited for utilizing simple programming-by-demonstration approaches such as kinesthetically teaching joint space trajectories. In addition the implicit encoding of the redundancy resolution has to be taught repetitively in every demonstration as it is not separated from the task.

1.2.1 FlexIRob@Harting: Kinesthetic Teaching in Confined Spaces is Difficult

In order to further support this hypothesis, in the following I briefly report results of a kinesthetic teaching experiment conducted as part of a large user study on physical human-robot interaction [22, 23] in the context of this thesis.

The experiment was conducted with 24 workers from a medium-sized manufacturing company – HARTING KGaA [24] – in Germany. Most of them had no practical experience with robots. They were asked to perform an adapted version of the wire loop game[3], a classical teach-in, together with a redundant robot, namely the 7-DoF KUKA Lightweight Robot IV, in a confined workspace. The setup consists of two styrofoam objects placed in the robot's workspace (see Fig. 1.2): One object represents fixed physical obstacles in the robot's workspace such as walls or racks that permanently constrain its movements. The other object constitutes a styrofoam parcours representing the target trajectory that is to be taught to the robot by the user. The participants were asked to guide the end-effector along the parcours but without getting the robot into contact with the environmental obstacles. The interaction controller used for this experiment is *gravity compensation* which is similar to the original gravity compensation controller provided by the robot [25], and which was re-implemented for technical reasons (cf. Chap. A for details). That is, during the interaction the users physically manipulate the robot in joint control mode, i.e. particularly no inverse kinematics controller or redundancy resolution to control the end-effector is provided. For a detailed description of the experiment and the data acquisition please confer to Chap. B.

The experimental results reveal the systematic deficit of the participants to successfully teach the robot system the desired trajectory in the constrained environment. Most of the them were not able to accurately follow (and therefore teach) the styrofoam parcours, which is indicated by a high task-space error of 0.12 ± 0.11 meters averaged over all participants. Fig. 1.3 shows exemplary trajectories of four participants indicating very different teach-in experiences and success. Whereas few users achieved a high task-space accuracy, e.g. `user04` with a maximum deviation to the target of approx. 0.05 meters, most of them failed to simultaneously find a valid joint configuration in the confined workspace and move the end-effector accurately along the desired trajectory. As a result, they deviated from the target movement up to 0.52 meters (see `user01` or `user02`). Furthermore, two participants aborted the wire loop game due to the difficulty, e.g. `user03`. Concerning the avoidance of environmental obstacles, only two users did not cause collisions between the robot arm and the obstacle. That means that independent from the task-space accuracy 22 of 24 participants failed to kinesthetically teach the robot a target trajectory without colliding with its environment. Finally, the results reveal an average teaching time of 93.4 ± 44.5 seconds. Rather than the pure average

[3] The original wire-loop game consists of a metal loop and a serpentine length. The loop has to be guided along the wire without touching it.

Fig. 1.2: Snapshots of the participants' teaching experience during the kinesthetic teaching experiment.

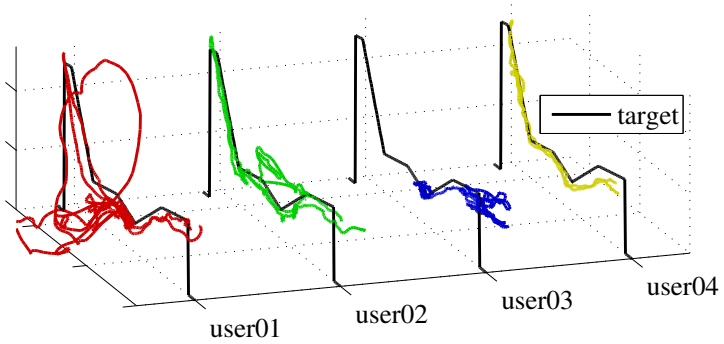


Fig. 1.3: Exemplary teach-in trajectories of four participants of the kinesthetic teaching experiment.

value, here the high variance across the participants is interesting. Whereas the fastest user managed to complete the task in 27 seconds, the longest teach-in took about more than 3 minutes.

The experiment demonstrates that, although it is possible (and the results showed that few of the participants managed) to solve the described kinesthetic teaching task in joint control mode, most users failed to maneuver the redundant robot in the confined workspace along a target trajectory. In [21] the authors hypothesize that "the operator may have good global situational awareness on the end goal for the manipulator, but may suffer from poor local situational awareness on the position of each manipulator joint", which is clearly illustrated in Fig. 1.2 (bottom right). The complexity of the teaching task may be caused by higher demands on the attentional system to switch the focus between task and environment (e.g. see Fig. 1.2 top left, top right) and/or even in increased physically exhaustive handling (e.g. see Fig. 1.2, bottom center). I conclude from this experiment that kinesthetic teaching of a redundant robot is difficult if non-trivial redundancy resolution is required.

1.3 Contribution and Goal of Thesis

The problems discussed above indicate that intuitive human-robot interaction controllers are needed that exceed plain kinesthetic teach-in at the kinematic level and reduce the complexity of teaching redundant robots. Therefore,

> *the overarching goal of this thesis is to develop concepts for an intuitive human-robot interaction to efficiently bootstrap and exploit redundancy resolutions in order to simplify teaching of redundant robots in changing environments.*

For the purpose of applicability to a variety of scenarios, these concepts should rely neither on external vision sensors nor on geometric models of the robot and the environment nor on other *explicit* criteria, since those often are not available in realistic applications. An important aspect of this work is therefore rather to develop methods enabling a human tutor to transfer his or her *implicit* knowledge about environmental constraints or personal preferences to the robot by providing demonstrations. Inspired by previous work on learning inverse kinematics [26, 27, 28, 29], a key concept is therefore to efficiently learn and encode redundancy resolutions from user demonstrations only relying on the robot's proprioceptive data gathered through the physical human-robot interaction. Based upon that, a staged interaction scheme is developed that allows users to incrementally provide such demonstrations and to benefit from them subsequently in order to simplify the teaching procedure.

Throughout this thesis, the required machine learning algorithms, interaction designs and interaction controllers are introduced, implemented and thoroughly evaluated. Aiming at a user-centered interaction concept, this work contributes

an in-depth analysis of the derived concepts from the users' point of view evaluating the tutors' teaching experience and success with state-of-the-art metrics for human-robot interaction systems [21].

1.4 Outline

The remainder of this work is structured as follows. As preliminaries for the derivation of the proposed concepts, in Chap. 2 I first discuss and analyze methods concerning inverse kinematics for redundant manipulators and programming-by-demonstration approaches, both with respect to flexibility and from the user's viewpoint. Subsequently, in Sect. 2.3 I present the proposed approach consisting in a dedicated set of high-level interaction controllers, a structured interaction workflow separated into *CONFIGURATION* and *PROGRAMMING* stages to incrementally teach and exploit redundancy resolutions, as well as the required learning methodology.

Throughout this thesis, these concepts are implemented and evaluated on our system prototype FlexIRob [30] utilizing the compliant 7-DoF-KUKA Lightweight Robot IV. The hardware setup and software abstractions utilized for this implementation are given in Chap. A. As already done in Sect. 1.2.1, in the further course of this work I will report results from the user study FlexIRob@Harting [22]. However, for the sake of readability the detailed study design, such as questionnaire design and course of demo for the participants, is put to Chap. B

In the following this work proceeds with Chap. 3 that analyzes the proposed method from a machine learning viewpoint: Neural-network-based learning approaches are discussed and proposed, in order to evaluate them with respect to their applicability and generalization abilities in this context. In contrast, Chap. 4 and Chap. 5 investigate the proposed concepts from a user-centered perspective. They specifically address their feasibility for non-experts and shed light on the implicit knowledge that is transferred to the robot controllers by means of the introduced interaction concept. Based on these findings an adopted variant of the *CONFIGURATION* stage is presented in Chap. 6 that improves on the issues learned from the user study by introducing haptic feedback to the tutor into the interaction loop. A further development that aims at increasing the robot system's autonomy, and thereby decreasing interaction effort required by the user, is developed in Chap. 7. It presents a method for model-free, autonomous path planning in the relevant, explored areas of the workspace, and is therefore interesting from an application-point-of-view. Chap. 8 finally presents a short, artificial scenario with a simulated 9-DoF manipulator. It is designed to test the proposed concepts in terms of scalability to other, more complex, but also less intuitive robot platforms. Finally, Chap. 9 summarizes and concludes the results obtained in this thesis.

Chapter 2

Kinesthetic Teaching of Redundant Robots in Confined Spaces

This chapter introduces the general approach proposed in this thesis to enable lay users intuitive teaching of redundant robots in confined spaces. The method consists of a coherent set of kinesthetic-teaching-based interaction controllers, machine learning tools to learn and encode user-taught redundancy resolutions, and hierarchical inverse kinematics controllers to embed the learned redundancy resolutions. Therefore, this chapter first briefly reviews standard robot control techniques for redundant manipulators, to then propose null-space projection methods to be utilized throughout this thesis. Second, related programming-by-demonstration (PbD) approaches are systematically discussed with respect to their interaction strategies regarding the user's viewpoint, and redundancy control. Finally, the approach of this thesis is derived as a combination of a hierarchical robot controller, methods for learning redundancy resolutions from demonstrations, and a structured human-robot interaction design.

2.1 Inverse Kinematics for Redundant Robots

The kinematics control problem of robotic manipulators can be formulated as finding proper *joint space* configurations $\mathbf{q}(t), \mathbf{q} \in \mathbb{R}^N$ of a robotic manipulator that realize a given desired *task space* trajectory $\mathbf{x}(t), \mathbf{x} \in \mathbb{R}^M$. *Redundancy* generally refers to the property of a manipulator to inhibit infinite solutions to the posed control problem and can be expressed in terms of the dimensions of the robot's task and configuration space. A robot is said to be redundant w.r.t. a given task, if $M < N$ [12, 13]. Assuming the general task of how to position a robot's end-effector in a 6-dimensional translational-rotational space, a device with more than six degrees of freedom, i.e. $N > 6$, is called *kinematically redundant* [12]. While the former definition is more general and describes redundancy as a relation between robot and task, the latter can be utilized even if a specific task is not (yet) known and is therefore the reference definition used throughout this thesis. The task space used in this work will typically relate to the position $\mathbf{x} \in \mathbb{R}^3$ of the robot's

end-effector or will include the rotational components $\mathbf{x} \in \mathbb{R}^3 \times SO(3)$, where $SO(3)$ refers to the group of all rotations in $\mathbb{R}^3$ (special orthogonal group). Hence, in this thesis $M = 3$ or $M = 6$.

Of special interest in this context are the forward kinematic mapping

$$\mathbf{x} = K_{\text{fwd}}(\mathbf{q}) \tag{2.1}$$

and the differential forward kinematics

$$\dot{\mathbf{x}} = J(\mathbf{q})\dot{\mathbf{q}} \tag{2.2}$$

of a robot where the Jacobian $J \equiv J(\mathbf{q}) = \frac{\partial K_{\text{fwd}}}{\partial \mathbf{q}}$ is a configuration dependent $(M \times N)$-matrix. Concerning typical rigid body industrial manipulators these mappings are well defined and known. For non-redundant robots they typically form the basis to directly calculate a (mostly) closed-form solution of the inverse kinematics mapping

$$\mathbf{q} = K_{\text{inv}}(\mathbf{x}) \quad \text{with} \quad \mathbf{x} = K_{\text{fwd}}(K_{\text{inv}}(\mathbf{x})). \tag{2.3}$$

However, redundant robots form a special challenge in solving this coordination problem. Since $M < N$ an infinite number of solutions exists and therefore criteria to select one of these are necessary. In the following I briefly discuss traditional approaches to the inverse kinematics problem for redundant robots and motivate null-space projection methods for further utilization in this work. For a more detailed and exhaustive overview the reader may refer to [13, 31].

While there exist methods to construct direct *inverse kinematics functions* such as Eq. (2.3) also for redundant manipulators [32], these rely on the construction of feasible invertible workspaces [13] and an analytical definition of a differential inverse function, requiring an in-depth mathematical knowledge of the specific task and the robot's kinematics. Hence, from a user-centric point of view they are not suitable in the context of this thesis.

One traditional approach of solving the redundancy is to explicitly *augment the task* $\mathbf{x}_a = (\mathbf{x}^\mathsf{T}, \mathbf{x}_c^\mathsf{T})^\mathsf{T}$ with additional $P = N - M$ constraints on the joint variables such that the resulting augmented task space spans the full N dimensions. From these constraints extended or augmented Jacobians $J_a \in \mathbb{R}^{N \times N}$ [31] are built that are typically non-singular and can be used to derive a joint space trajectory $\mathbf{q}(t)$ by inverting an adapted version of the mapping in Eq. (2.2)

$$\dot{\mathbf{x}_a} = J_a(\mathbf{q})\dot{\mathbf{q}} \tag{2.4}$$

and integrating the obtained values over time. However, these methods rely on either defining explicit additional tasks $\mathbf{x}_c(t), \mathbf{x}_c \in \mathbb{R}^P$ or at least an additional explicit constraint function $c(\mathbf{q}) : \mathbb{R}^N \mapsto \mathbb{R}$ on the joint variables exhibiting constraints to resolve the redundancy. With this additional goals the possibility of algorithmic singularities raises. These are situations where the augmented kinematics problem is singular while the problem of solely tracking the desired task

space trajectory $\mathbf{x}(t)$ is not and typically relate to contradicting tasks for the constraints and the end-effector [31]. Considering arbitrary tasks space trajectories "it is quite difficult to select an ad hoc constraint task to be satisfied together with the original task, unless one has enough insight into the specific problem" [13], which generally renders these methods unsuitable for non-expert users in the context of this thesis.

Other methods to redundancy resolution utilize *optimization techniques* exploiting the additional degrees of freedom for minimizing a given cost function $H(\mathbf{q}) : \mathbb{R}^N \mapsto \mathbb{R}_{\geq 0}$ along the desired task $\mathbf{x}(t)$. From a practical viewpoint in the context of this thesis optimization methods can be classified in either *offline* or *online* methods. Offline methods such as global optimization approaches or methods derived from optimal control theory (e.g. [33, 34]) require heavy computations and a pre-defined task space trajectory [13]. Although they provide global optimal solutions they are impractical for application in human-robot interaction scenarios. In contrast, online or instantaneous methods can be applied in real-time and without an a priori definition of a task only locally optimizing a performance criteria during a desired motion. Hence, throughout this work I only focus on these latter methods. Such criteria are for instance avoidance of singularities, minimal joint velocity or minimum energy consumption, and can be implemented based on *simple Jacobian-based techniques*. These rely on inverting the mapping Eq. (2.2)

$$\dot{\mathbf{q}} = K(\mathbf{q})\dot{\mathbf{x}}, \tag{2.5}$$

where a typical choice for the generalized inverse K is based on the pseudo-inverse of the Jacobian $J^\dagger = J^\mathsf{T}\left(JJ^\mathsf{T}\right)^{-1}$. For instance, having $K = J^\dagger$ results in the minimum 2-norm solution that generates minimum norm joint velocity solutions. Other work in that direction proposes to use different norms such as the weighted-norm solution $K = W^{-1}J^\mathsf{T}\left(JW^{-1}J^\mathsf{T}\right)$ to distribute joint velocities differently e.g. to locally minimize kinetic energy [35] or damped least-squares solutions $K = J^\mathsf{T}\left(JJ^\mathsf{T} + \mu^2\mathbb{I}\right)^{-1}$ for robust singularity avoidance where $\mu(\mathbf{q})$ is a measure for singularity [13]. While these methods are simple to implement and provide general applicability to redundancy resolution they lack the flexibility required to embed user preferences of how the redundancy should be resolved.

In order to overcome this lack in flexibility the *null-space projection* approach can be utilized. It formulates a more general solution to Eq. (2.2)

$$\dot{\mathbf{q}} = J^\dagger\dot{\mathbf{x}} + \left(\mathbb{I} - J^\dagger J\right)\dot{\mathbf{q}}_c, \tag{2.6}$$

where additional constraints $\dot{\mathbf{q}}_c \in \mathbb{R}^N$ on the joint velocities are projected in the null-space of the Jacobian J, hence shaping the robot's motion $\mathbf{q}(t)$ while not affecting the produced task trajectories $\mathbf{x}(t)$. This separation of concerns is advantageous because it allows to flexibly plug in any desired constraints on the robot's motion without disturbing its reliability in task space. In contrast to the task augmentation methods discussed above, a strict prioritization prevents the

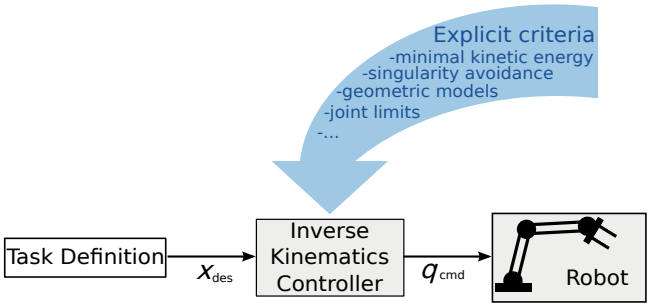


Fig. 2.1: Sketch of traditional approaches to redundancy resolution such as task augmentation and null-space projection methods. Besides an explicit definition of a task, explicit analytical criteria are required to resolve the manipulator's redundancy.

joint constraints (secondary task) from disturbing the accuracy of the task space trajectory (primary task). This advantage has led to the development of a large number of approaches using *gradient projection*. Again by means of a properly modeled scalar cost function $H(\mathbf{q})$, redundancy resolution can be implemented by calculating $\dot{\mathbf{q}}_c = \frac{\partial H}{\partial \mathbf{q}}$. By this means, a manipulator's redundancy can be used for join-limit avoidance, maximizing manipulability or maneuverability [13], real-time obstacle avoidance [36, 37, 38, 39] or maximizing kinematic fault-tolerance [40]. Another approach utilizes Eq. (2.6) to include hard constraints in the joint space that must be satisfied to fit a manipulator's physical limits such as joint ranges and maximal velocity or acceleration [41]. Still, with all methods listed above the redundancy is resolved based on an explicit criterion that has to be modeled as an analytical expression in terms of the joint variables $\mathbf{q}$ as sketched in Fig. 2.1. However, the null-space projection approach Eq. (2.6) offers great flexibility to include user-specified redundancy resolutions in terms of "customized" joint velocities $\dot{\mathbf{q}}_c$ as I will show in Sect. 2.3. In addition, it is very simple to implement for any rigid-body manipulator and thus serves as basis for the work in this thesis.

It is worth mentioning that based upon this method or derived from other approaches such as higher-order differential kinematics, also more advanced techniques have been developed e.g. using task priority controllers allowing to define multiple tasks [31] and even to switch their priority smoothly during execution [42]. But in the context of this thesis they do not provide more flexibility or allow more intuitive programming for non-expert users. They rather impose higher requirements in terms of computational costs.

In contrast to analytic approaches, an important area of work is concerned with learning inverse kinematics of redundant robots [43]. This is typically conducted in cases where the derivation of analytical models is difficult or rather impossible such as non-rigid robots [44] or very difficult as for humanoid robots [26, 27]. However, only a limited number of these consider the introduction of

additional constraints for redundancy resolution. An exemplary approach in this category is constrained supervised learning [45] which explicitly models configurational and temporal constraints for solving inverse kinematics problems. A more recent approach allows for dynamic redundancy resolution [46] by learning the forward model and the respective projectors required for additional redundancy constraints. However, the main goal of these methods is to bootstrap an inverse model for control rather than learning different redundancy resolutions by means and for the sake of human-robot interaction.

2.2 Kinesthetic Teaching as an Intuitive Programming Interface

To overcome the necessity of explicit criteria, *programming by demonstration (PbD)* has become an inevitable method as it "offers an implicit means of training a machine, such that explicit and tedious programming of a task by a human user can be minimized or eliminated" [10]. A central concept in this research field is a human teacher providing demonstrations in a human-robot interaction scenario from which a problem-specific control policy for the robot is derived. Therefore, the different PbD approaches can be categorized according design choices regarding problem space representation, policy design and learning algorithm as well as the demonstration method [47]. Since a key question of this thesis is an intuitive interaction and teaching of redundancy resolutions from the user's viewpoint, the focus in this section lies on the latter, i.e. the different interaction strategies for the human teacher to provide demonstrations.

According to [47] the strategies in question can be divided into *imitation* and *demonstration* strategies. Imitation approaches record data by observing the human teacher with cameras or other sensors that allow to measure the human demonstrations. The robot manipulator is not involved during the execution of these demonstrations. One crucial disadvantage of this strategy is the correspondence problem, which is described as a possible mismatch between the sensorimotor spaces of the learner and the teacher [48]. To overcome the correspondence problem typically a mapping has to be specified to allow the robot learning from external observations which is a non-trivial problem.

In contrast, during demonstration strategies the states and actions to learn from are recorded from the robot itself. This often is referred to as self-imitation or self-supervised learning. Prominent examples are *teleoperation* and *kinesthetic teaching*. While teleoperation approaches have been applied successfully in a huge variety of applications [47] ranging from simple manipulation tasks to surgical robotics applications [49], direct and physical manipulation of a robot, i.e. kinesthetic teaching, provides a more user-friendly interface [10]. Concerning non-expert users, recent studies indicate such a direct physical human-robot interaction to be favorable [50]. By kinesthetic teaching the teacher provides state sequences in the

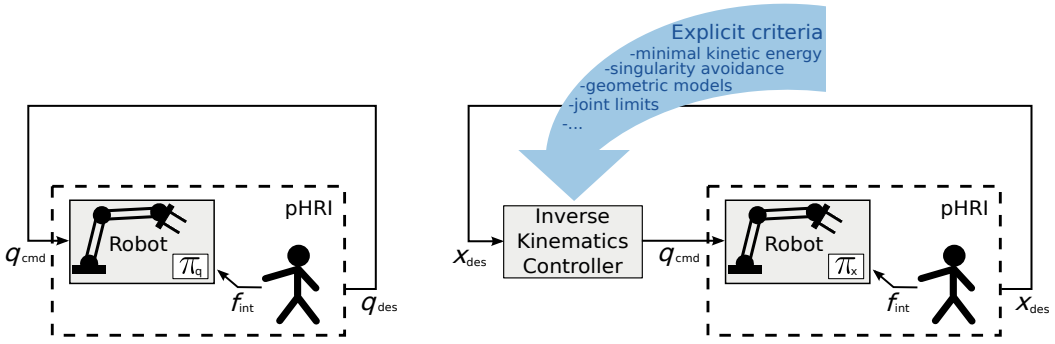
robot's sensorimotor space which can be used directly for learning. No additional mapping is required. Regarding industrial manufacturing, a recent field study with non-expert users revealed that kinesthetic teaching serves as a suitable input device for demonstrating manual assembly tasks which can be learned quite fast by human demonstrators [51]. The work in [52] motivates kinesthetic teaching as input device also from a social interaction perspective. It presents studies with social animals that use the concept of *moulding* or *putting through*, i.e. the process of putting the learner through a set of teacher-defined actions and states which are available to the learner, to speed up the learning process. Following this argumentation, kinesthetic teaching seems to be a natural and intuitive way for humans to teach robots [53].

2.2.1 Demonstrating in Configuration Space

From a user perspective existing PbD approaches can be categorized into two classes. The first class covers all methods where the human teacher physically operates the robot in configuration space. That is, in order to guide the robot through a desired motion demonstration the user has to take into account all joints of the robot as sketched in Fig. 2.2(a). Technically, physical guidance can be accomplished by using back-drivable motors, passively compliant robots or active compliance using force-torque sensors integrated into the robot's actuators [54, 5, 6]. From a low-level control point of view, the interaction forces f_{int} applied by the user to the robot's links and joints are transformed to a new desired joint position $\mathbf{q}_{\text{des}}$ by means of a control policy π_q. For rigid-body manipulators this is typically realized by means of proper admittance or impedance controllers [55, 25, 56]. For passively compliant robots π_q is a combination of low-level controllers and the inherent compliance of the robot structure e.g. as in [57]. Mostly, these approaches realize some form of *gravity compensation controller* where the robot compensates its own gravity forces but does not counteract externally applied forces/torques allowing the user to move it freely.

In the literature, this approach is often utilized in the context of humanoid robotics as it is also seen as a social interaction device [53]. By kinesthetic demonstration in joint space humanoid robots have been taught to imitate human gestures [58], move chess pieces [59] and perform simple manipulation [60] as well as household tasks such as pouring or spooning sugar [61]. Other applications refer to humanoids learning drumming and ball throwing [62] as well as reproducing and sequencing even bi-manual skills [63]. But also single robotic arms have been used in this context. In [64] a 7-DoF manipulator is taught ironing tasks and opening doors which consists of positional and force skills. Also more recent work [65] uses kinesthetic teaching for instructing the industrial 7-DoF KUKA Lightweight Robot IV with basic real-world manipulation tasks.

However, these approaches suffer from high complexity imposed on the user. As discussed in Sect. 1.1, the configuration space of robots in the context of this



(a) Demonstrating in configuration space: The user interacts with the robot by applying forces f_{int} to its limbs and joints which then are transformed to a desired motion $\mathbf{q}_{\text{des}}$ by the specific interaction controller π_q available on the robot. Hence, from a user perspective the teacher simultaneously operates all joints through the desired motion.

(b) Demonstrating in task space: The interaction between user and robot reduces to task space (typically end-effector space). The user-applied force f_{int} is used by the specific interaction controller π_x to calculate only a desired task space position $\mathbf{x}_{\text{des}}$. Therefore, explicit criteria are needed to account for the manipulator's redundancy.

Fig. 2.2: Schematic view on programming-by-demonstration approaches with different demonstration strategies.

thesis is typically high-dimensional, with seven or more DoF the teacher has to operate simultaneously. As the case study of kinesthetic teaching in confined spaces in Sect. 1.2.1 shows, this interaction often is too complex and imposes high mental load. The work in [66] investigates kinesthetic teaching of a humanoid robot to stand up, walk and even to perform a head stand. While it shows that, in combination with optimization techniques, teaching these tasks is possible, it reports that a single teaching procedure can take up to 15 minutes. As most of the discussed approaches rely on trajectory-based teaching, the users even have to plan in this high-dimensional space which renders the teaching task even more complex. According to [64], kinesthetic teaching in configuration space might also not be suitable for all kinds of robots since it tends to decorrelate the movement: "The main reason is that it is easier for the user to move the motors one by one during demonstration than executing a natural coordinated movement." [64]. Also Calinon *et al.* already stated in their pioneering work [59] that "[...] it remains difficult to generalize the whole body motion because it is not possible to control simultaneously more DOFs (e.g. moving the two arms and two legs of the robot simultaneously)." Furthermore, from a practical point of view it is worth mentioning that due to the lack of separation both task definition and environmental constraints have to be re-taught when either of them is changed. Reusing either of them is therefore not straightforward if possible at all.

2.2.2 Demonstrating in Task Space

In contrast, in the second class the teaching process is restricted to task space, meaning that the user only interacts with the robot by positioning its end-effector instead of manipulating the entire robot structure, see Fig. 2.2(b). The forces f_{int} applied by the human operator to the robot's end-effector are measured or estimated and used by the specific low-level controller π_x (cf. e.g. [67, 68, 69] or [70] for an overview) to calculate a desired Cartesian position $\mathbf{x}_{\text{des}}$ of the end-effector. Technically this is typically realized by mounting force/torque sensors at specific joints or the end-effector of a robot and was a typical way to realize some kind of teach-in procedures in early industrial programming-by-demonstration approaches [71]. These include spot or arc welding, loading and unloading numerical control machine tools or basic assembly operations. An interesting overview about very early work in that direction is given in [72]. But also very recent work utilizes this approach for teaching industrial tasks such as manual assembly [51, 73] or screw tightening [74] as well as wheel mounting to a car body [69]. In [75] this interaction strategy was employed to teach a humanoid robot surface cleaning motions, while it balances its lower body. the

On the one hand these techniques reduce the teaching complexity from a possibly high-dimensional joint space to a typically three- or six-dimensional task space as sketched in Fig. 2.2(b). On the other hand, regarding the problem of redundancy resolution they face the same issues as traditional approaches discussed in Sect. 2.1. Either the demonstrations are recorded in free, non-confined spaces where additional constraints on the redundancy can be neglected or they explicitly have to model criteria for redundancy resolution beforehand or during task execution.

2.3 Proposed Method: Incremental Kinesthetic Teaching in Task and Configuration Space

As outlined in Sect. 1.3, the main goal of this thesis is to enable non-experts fast and intuitive programming of redundant robots. In order to circumvent explicit modeling, I propose a set of learning and control approaches that allow users to define environmental constraints and specific task space trajectories *solely based on kinesthetic teaching*. I meet the fact that trajectory-based kinesthetic teaching of redundant robots in confined spaces results in a high cognitive load for the user - as demonstrated in Sect. 1.2.1 and discussed in Sect. 2.2.1 - by systematically reducing the complexity. The key idea is to *separate teaching of (task-independent) constraints from the actual task* the robot should execute in end-effector space allowing an incremental, simplified procedure that improves the user's teaching experience.

Fig. 2.3 provides a high-level view on the proposed approach. As discussed above, typical programming-by-demonstration methods either neglect the prob-

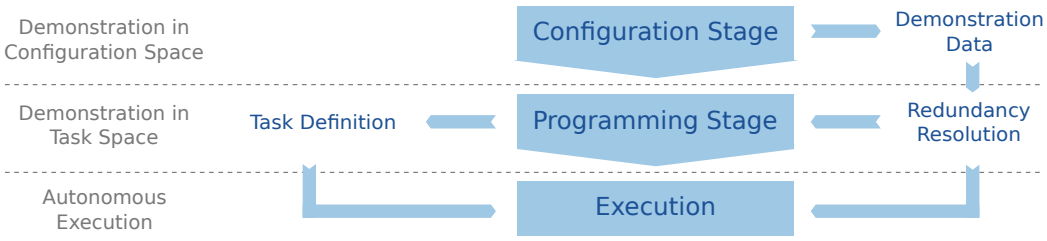


Fig. 2.3: Schematic view on the proposed incremental kinesthetic teaching procedure. Teaching of redundancy resolution is separated from demonstrating the actual task. In a dedicated *CONFIGURATION* stage the teacher records demonstration data that is utilized to infer a customized fixed redundancy resolution. For the subsequent *PROGRAMMING* phase this is embedded in the robot's control architecture to allow recording of a kinesthetic demonstration in task space. Finally, the resulting task definition and redundancy resolution are used to execute the desired motion while satisfying the taught constraints.

lem of redundancy resolution or apply explicit criteria during the execution or reproduction of a task [64], i.e. after teaching. In contrast, in this procedure the redundancy resolution needs to be defined first. In the *CONFIGURATION* stage the user provides demonstrations of valid or preferred redundancy resolutions in selected parts of the robot's workspace. By means of machine learning techniques these demonstrations are properly encoded and generalized to infer a fixed redundancy resolution over the entire workspace which subsequently can be embedded into a hierarchical control architecture. Hence, the redundant manipulator can be controlled arbitrarily in task space while resolving the redundancy online according to the taught constraints. In particular, an assisted interaction controller is derived that allows an assisted programming in the next stage, i.e. the *PROGRAMMING* phase. The kinesthetic interaction between human and robot reduces to task space because the redundancy is resolved online according to the taught constraints. By this means a task definition is provided. Finally, the task can be reproduced autonomously by the robot using an hierarchical control approach incorporating the environmental constraints defined by the user.

In the following, I introduce and discuss the required concepts for the proposed approach. These consist in machine learning algorithms to learn and encode user-specific redundancy resolutions, a hierarchical control concept using null-space projection methods as discussed in Sect. 2.1 and dedicated interaction controllers for kinesthetic teaching in task and configuration space.

2.3.1 Kinesthetic Teaching of Redundancy Resolutions

The main goal of the *CONFIGURATION* stage is to enable the human teacher to transfer its implicit knowledge about *environmental constraints* to the robot in an efficient and intuitive manner.

Intuitiv transfer of this knowledge is accomplished by a programming-by-demonstration approach that infers the redundancy resolution from demonstration data $\mathcal{D} = (\mathbf{x}_l, \mathbf{q}_l)_{l=1,\dots,L}$ gathered through kinesthetic teaching. It particularly utilizes the compliance features of the used robot platform in a demonstration-in-configuration-space strategy as discussed in Sect. 2.2.1. However, in contrast to those methods, teaching in this stage is not trajectory-based. I rather follow the idea that the constraints imposed on the robot are caused by geometric relations between the manipulator and its environment. They can relate to physical obstacles in the robot's workspace (e.g. cluttered workspaces, walls), areas not to be covered by its links (e.g. for safety regions in close human-robot collaboration) or user-preferred redundancy resolutions (e.g. changing from left-handed to right-handed in hand-over tasks [76]). Hence, they can be encoded *location-based* by means of a mapping

$$\mathbf{q}_c : \mathbb{R}^M \mapsto \mathbb{R}^N \quad \text{with} \quad \mathbf{q}_c \equiv \mathbf{q}_c(\mathbf{x}_{\text{des}}), \tag{2.7}$$

that maps desired task space positions $\mathbf{x}$ to a corresponding preferred joint configuration $\mathbf{q}_c$.

This is advantageous also in terms of *efficiency*. The idea of selecting a single, fixed solution to the inverse kinematics problem rather than taking all solutions into account before choosing the "optimal" one simplifies both the teaching procedure, as only one solution needs to be shown, and the learning process, as learning and encoding of multi-valued inverse function still is an ongoing research question [77]. This is also the rational behind past and current work on efficient bootstrapping of an inverse kinematics function for complex, e.g. high-dimensional [26] or non-rigid [44] robot structures. Another prerequisite for an efficient teaching procedure is to avoid exhaustive sampling of the entire workspace or along constraints therein. For this purpose, the interaction is structured such that the human tutor needs to provide data only in few relevant areas of the workspace. The recorded data can be regarded as key-postures in the robot's workspace that need to be generalized over the entire workspace similar to the concept of key-frames that are used to encode trajectories [60]. Then, proper machine learning approaches are required in order to learn and generalize a global redundancy resolution function $\mathbf{q}_c(\cdot)$ from only these few demonstrations $\mathcal{D} = (\mathbf{x}_l, \mathbf{q}_l)_{l=1,\dots,L}$. The introduction, analysis and discussion of such approaches is the main goal of Chap. 3. Subsequently, the learned redundancy resolution can be embedded into the robot's control architecture using hierarchical controllers as discussed in the following and sketched in Fig. 2.6.

The Non-Convex-Solution-Sets Problem

Learning redundancy resolutions from the provided demonstration data $\mathcal{D}$ from a machine learning view-point is loosely connected to learning an inverse model from which a mathematical problem arises. Given a redundant and non-linear

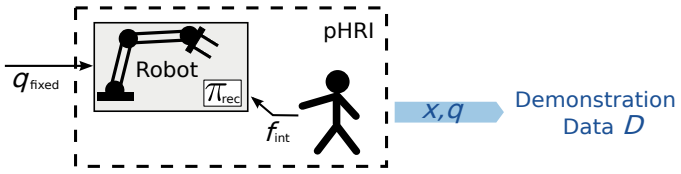


Fig. 2.4: Proposed interaction controller for *compliant recording* of training data $\mathcal{D}$ in the *CONFIGURATION* phase. Given a desired fixed posture $\mathbf{q}_{\text{fixed}}$ the controller π_{rec} allows physical deflection of the robot's end-effector by means of the forces f_{int} while trying to keep the robot configuration as close as possible to $\mathbf{q}_{\text{fixed}}$.

forward model of a robot manipulator as considered in this thesis, the inverse kinematics problem exhibits multiple, non-linear solution sets [77, 78]. This prohibits learning from arbitrary data $\mathcal{D} = (\mathbf{x}_l, \mathbf{q}_l)_{l=1,\dots,L}$ that include demonstrations $\mathbf{x}_{\text{des}} = K_{\text{fwd}}(\mathbf{q}_i) = K_{\text{fwd}}(\mathbf{q}_j)$ with $\mathbf{q}_i \neq \mathbf{q}_j$ as averaging these examples can lead to undesired actions $\hat{\mathbf{q}}$ with $K_{\text{fwd}}(\hat{\mathbf{q}}) \neq \mathbf{x}_{\text{des}}$ [79].

Different strategies have been developed to circumvent this issue. Typical approaches recover invertibility of the problem by strong biases of the inverse model or incorporating temporal context [77]. Another approach is to restrict inverse learning to data incorporating only a single solution. In [78] a bootstrapping paradigm is developed that selects data to be incorporated into the training set based on a reasonable weighting scheme accounting for efficiency and intention direction. In [26] the authors transform the problem to a convex one in the temporal domain by considering the inverse problem only locally in the vicinity of a particular $\mathbf{q}$.

The method to solve this issue proposed in this thesis is motivated by the latter two approaches. By a structured user interaction and dedicated interaction controller shown in Fig. 2.4 the tutor is forced to record training data $(\mathbf{x}_l, \mathbf{q}_l)_{l=1,\dots,L}$ only locally around a desired, fixed joint configuration $\mathbf{q}_{\text{fixed}}$. The interaction model to achieve this goal is described in the following.

Interaction Model

The interaction model designed for the *CONFIGURATION* stage consists of the two-staged interaction workflow as shown in Fig. 2.5, the respective interaction controllers *gravity compensation* π_q and *compliant recording* π_{rec} as shown in Fig. 2.2(a) and Fig. 2.4, respectively, as well as interaction triggers `on_affected` and `on_converged` to switch between the two sub-stages *APPROACHING* and *RECORDING*.

First, the robot is switched to *gravity compensation* π_q, triggered by the tutor through touching the robot and thus applying external forces to it (`on_affected`). This mode enables the tutor during the *APPROACHING* phase to move all joints easily while maneuvering the robot to a desired training area according to con-

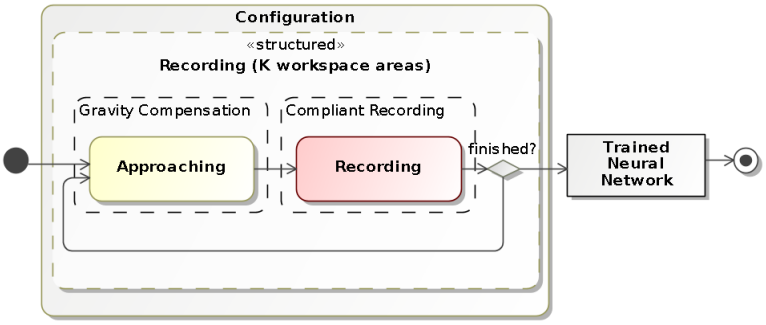


Fig. 2.5: Proposed interaction workflow for kinesthetic teaching of redundancy resolutions during *CONFIGURATION*. Transitions between the individual substages are realized based on the triggers `on_affected` and `on_converged`.

straints in the environment, e.g. to avoid collisions with fixed physical obstacles by selecting a certain elbow configuration. Second, when the system recognizes that the robot is no longer moved by the user for a short period of time and thus has reached its desired joint configuration $\mathbf{q}_{\text{fixed}}^{(k)}$, it is switched to the *compliant recording* controller π_{rec} (`on_converged`). Third, in order to start the actual *RECORDING* the user again applies external forces to physically deflect the robot from $\mathbf{q}_{\text{fixed}}$ (`on_affected`) and record training data $\mathcal{D}^{(k)} = (\mathbf{x}_l^{(k)}, \mathbf{q}_l^{(k)})_{l=1,\dots,L^{(k)}}$. Last, once released for a short period of time, the system stops recording (`on_converged`). As for the feedback about the current system state, the transitions based on the interaction trigger `on_affected` can be naturally perceived by the tutor through the physical interaction. In contrast, transitions based on `on_converged` cannot be felt through touch as the user holds the robot in a still position or even releases it. Therefore, these are acknowledged by the robot by means of a short acoustic signal to provide an alternate minimal feedback. The described procedure can be repeated for as many training areas K the human tutor finds relevant. At the end of the *CONFIGURATION* stage the data $\{\mathcal{D}^{(k)}\}_{k=1,\dots,K}$ is passed to a machine learning algorithm according to Chap. 3 for data-driven learning of the taught constraints resulting in the learned redundancy resolutions $\mathbf{q}_c(\cdot)$ generalized over the entire workspace.

2.3.2 Hierarchical Control

In order to realize the proposed approach, a control concept is required for the *PROGRAMMING* and *EXECUTION* stages that is able to fuse taught redundancy resolution $\mathbf{q}_c(\cdot)$ and desired task space trajectories. In this thesis, I follow the idea of hierarchical control concepts that are regarded as crucial to flexibly compose complex motions from individual control aspects [80], e.g. to synthesize motions of humanoid robots [81]. The particular solution used in this thesis is

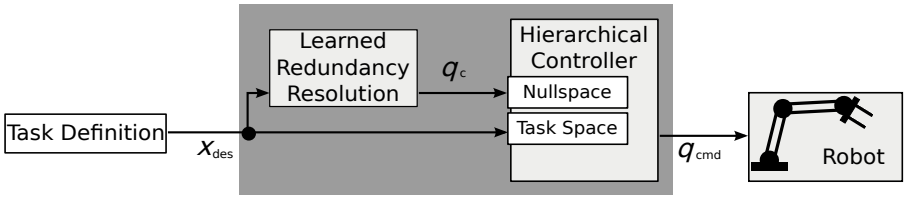


Fig. 2.6: Hierarchical control concept utilized in this thesis. The user-taught environmental constraints are embedded in the control architecture in form of a learned function $\mathbf{q}_c(\cdot)$ that maps task space positions $\mathbf{x}_{\text{des}}$ to constrains $\mathbf{q}_c$ in joint space. A null-space projection controller then fuses desired task space motions $\mathbf{x}_{\text{des}}$ with desired redundancy resolution $\mathbf{q}_c$ to a $\mathbf{q}_{\text{cmd}} = \mathbf{q} + \dot{\mathbf{q}}$ according to Eq. (2.11).

the control basis framework (CBF) of Grupen *et al.* [82, 83] that provides online-capable methods to combine planning and closed-loop feedback controllers. It realizes multi-objective control by combining control primitives in a prioritized manner by projecting lower-prioritized actions in the null-space of the higher-prioritized ones.

Throughout this thesis, the desired task space motion is regarded as prioritized over the taught redundancy constraints for two reasons. First, in the context of industrial robotics it is regarded as the main task that the robot should perform as accurate as possible - independent of the taught environmental constraints or user preferences. This is of particular interest during the *PROGRAMMING* phase, where the user interacts with the robot's end-effector in task space. Requesting a desired end-effector motion which is not accurately executed by the utilized controller might result in jerky motions that disturb a natural interaction experience. Second, from a practical viewpoint the to-be-executed task space trajectories are expected to not contradict with the taught constraints as the former are defined by kinesthetic teaching *while* the latter already are embedded in the control architecture.

Hence, the proposed controller relies on the null-space projection method in Eq. (2.6). The primitive control aspects utilized in this thesis are simple square potentials $\nabla\phi$ with

$$\phi_x(\mathbf{x}) = \kappa_x(\mathbf{x}_{\text{des}} - \mathbf{x})^{\mathsf{T}}(\mathbf{x}_{\text{des}} - \mathbf{x}), \tag{2.8}$$

$$\phi_q(\mathbf{q}) = \kappa_q(\mathbf{q}_c - \mathbf{q})^{\mathsf{T}}(\mathbf{q}_c - \mathbf{q}), \tag{2.9}$$

that allow quadratic tracking of the desired task space position $\mathbf{x}_{\text{des}}$ and the user taught redundancy resolution $\mathbf{q}_c \equiv \mathbf{q}_c(\mathbf{x}_{\text{des}})$, respectively [83]. The resulting control law can be written as

$$\dot{\mathbf{q}} = J^{\dagger}\nabla\phi_x + \left(\mathbb{I} - J^{\dagger}J\right)\nabla\phi_q \tag{2.10}$$

$$= \kappa_x J^{\dagger}(\mathbf{x}_{\text{des}} - \mathbf{x}) + \kappa_q\left(\mathbb{I} - J^{\dagger}J\right)(\mathbf{q}_c(\mathbf{x}_{\text{des}}) - \mathbf{q}), \tag{2.11}$$

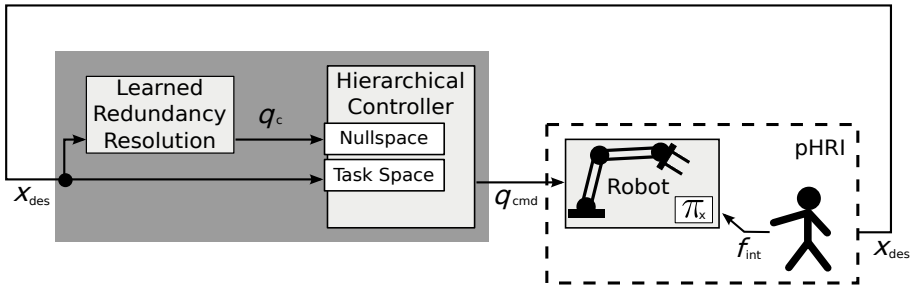


Fig. 2.7: Proposed *assisted gravity compensation* controller for the *PROGRAMMING* stage following the demonstration-in-task-space strategy as discussed in Sect. 2.2.2. Utilizing the low-level controller π_x it allows free interaction with the robot's end-effector while controlling the redundancy resolution according to the learned function $\mathbf{q}_c(\cdot)$.

where $\kappa_x > 0$ and $\kappa_q > 0$ are the coefficients of the respective potentials. The desired joint space motion is then obtained by integrating the calculated $\dot{\mathbf{q}}$ over time. This embeds the learned redundancy resolution into the control architecture as sketched in Fig. 2.6 allowing arbitrary task space movements while respecting the taught null-space constraints and an assisted interaction controller as introduced in the following.

2.3.3 Assisted Programming in Task Space

The idea of the *PROGRAMMING* phase is to provide assistance to the teacher during the kinesthetic demonstration of the task. Utilizing the previously learned redundancy resolution $\mathbf{q}_c(\cdot)$ embedded into the hierarchical control concept described above, the human-robot interaction can be simplified following the demonstration-in-task-space paradigm as discussed in Sect. 2.2.2. Exploiting the robot's compliance features by means of the interaction forces and the low-level controller π_x, the user generates a desired change in the position of the end-effector. The redundancy is resolved online during that interaction using the hierarchical controller and the embedded mapping $\mathbf{q}_c = \mathbf{q}_c(\cdot)$ as shown in Fig. 2.7. This allows *free movements in the task-space* while simultaneously *controlling the joint-space* according to the constrained environment. During the kinesthetic teaching of the actual task, the user can concentrate on providing proper, trajectory-based task space demonstrations because the robot assists in resolving the redundancy.

Interaction Model

An exemplary interaction model for the *PROGRAMMING* phase to test the derived concepts is implemented as shown in Fig. 2.8. Like the preceding stage it consists of two sub-stages *APPROACHING* and *TEACH-IN* that are engaged and altered by the user using the triggers `on_affected` and `on_converged` but with

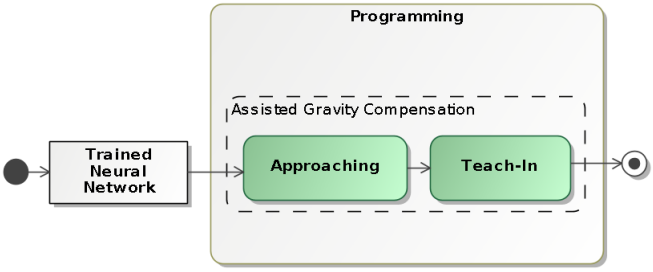


Fig. 2.8: Proposed interaction workflow for assisted *PROGRAMMING* using the trained neural network embedded into the hierarchical controller. Transition between the individual sub-stages *APPROACHING* and *TEACH-IN* is again realized based on the triggers `on_affected` and `on_converged`.

the difference that both sub-stages utilize the derived *assisted gravity compensation* sketched in Fig. 2.7.

Touching the robot (`on_affected`) triggers the system to switch to *APPROACHING*, during which the user can guide the robot to a desired starting position $\mathbf{x}_{\text{start}}$. Once this position is reached by halting the robot for a short period of time (`on_converged`), the tutor can enter the *TEACH-IN* (`on_affected`) and start recording a task space trajectory $\mathbf{x}(t), t = 1, \ldots, T$ that reflects desired temporal and positional aspects of the task the robot should perform. Recording is stopped by using the trigger `on_converged` once more. Again, the transitions based on `on_converged` are acknowledged by the system by means of a short acoustic signal. The described procedure can be repeated for as many demonstrations as required. The recorded data can then be employed for arbitrary state-of-the-art policy learning or reinforcement learning approach [10, 84, 85, 86, 87] to learn and improve on the specific task.

2.3.4 Implementation

In order to test and evaluate the proposed approach all the described concepts above are implemented and tested throughout this thesis by means of the robotic system prototype FlexIRob [30] which is based on the KUKA Lightweight Robot IV shown in Fig. 1.1. The LWR IV is a redundant manipulator with seven degrees of freedom and an impedance-based control scheme [25]. The latter enables active compliance of the manipulator as basis for the proposed interaction controllers utilized for the desired physical human-robot interaction. Besides entering and leaving the *CONFIGURATION* and *PROGRAMMING* phases the entire workflow is based on the physical interaction triggers `on_affected` and `on_converged` which are realized by measuring externally applied torques and the rate of motion over time, respectively. The mentioned exceptions are `start_configuration`, `finished_configuration`, `start_programming`, `finished_programming` and are

realized with a simple graphical user interface. Despite the haptic and acoustic feedback about the current system state, a visualization displays robot, recorded training data and the current state.

In order to present the analysis and evaluation of the derived interaction concepts in a coherent way both from the machine learning viewpoint and the human perspective, their implementations are put to the respective following chapters. As efficiency of the presented teaching procedure plays a central role, Chap. 3 presents, discusses and evaluates proper neural-network-based learning algorithms with respect to their feasibility regarding the recorded training data and generalization abilities. In order to shed light on the teaching experience and success from the tutoring perspective, the *CONFIGURATION* and *PROGRAMMING* stages are evaluated in dedicated chapters 4 and 5, respectively, where also the implementation of the proposed controllers is briefly shown. A detailed description of the FlexIRob system setup is given in Chap. A including the hardware setup, the used middle-ware and employed software abstractions for learning-from-demonstration scenarios.

2.4 Discussion and Related Work

This chapter proposed an incremental procedure to reduce the complexity of kinesthetic teaching of redundant robots. It relies on the idea of splitting the entire teaching procedure into teaching constraints for the manipulator's null-space and a dedicated teaching procedure of the task. This separation of concerns regarding task and null-space motion is not new and was proposed to be utilized in physical human-robot interaction since the emergence of dexterous and compliant robots [88, 89]. However, on the one hand, learning and generalization of motions was not regarded in that work. On the other hand, those early approaches neglected the human perspective concerning efficiency and intuitivity of pHRI.

The approach presented in this chapter aims at systematically reducing the teaching complexity for the user with an incremental approach allowing to separately teach task and environmental constraints. Such incremental approaches have been studied also in other work either to simplify the teaching procedure or to incrementally refine motions online.

In [90] the authors propose an interesting incremental design for simplifying learning by demonstration on a humanoid robot. It relies on a first stage, where observational learning or imitation learning results in a first hypothesis of a motion, and a second stage, during which the learned motion is refined using kinesthetic teaching. Hence, the process is simplified for the teacher by using its own embodiment in the first stage neglecting complex physical guidance of the robot. While this approach has been shown to work on a high-dimensional humanoid robot, additional technical means are required to observe the human's motion, and a mapping is needed to solve the correspondence problem as briefly discussed in Sect. 2.2.

An alternative scheme [91] to allow user-defined redundancy resolution realizes bio-mimetic models using Bayesian modeling techniques to build an inverse kinematics solver based on null-space projections preferring human-like joint configurations for redundancy resolution of humanoid robot arms. While this idea is similar to the presented concept of learning redundancy resolutions, it still relies on observational imitation learning which is not feasible in the context of this thesis.

Another interesting approach contributing to the field of incremental robot programming in physical interaction is introduced in [64]. Here, teaching of force and positional skills is separated into two dedicated demonstration phases. First, only the positional part of the skill is taught via kinesthetic teaching on a 7-DoF manipulator. Second, this skill is augmented with force terms being taught on a separate input device while already executing the positional skill on the robot. While that work is conceptually very similar to the approach in this thesis, it looks differently on the advantages and problems of redundancy. The authors explicitly separate kinesthetic teaching for the task from the advantage of exploiting redundancy for task execution only: "This advantage does not relate to or affect the teaching, but matters only during the reproduction of the task." As discussed in Sect. 1.2 and demonstrated in Sect. 1.2.1 this does not hold for teaching in confined spaces.

The work in [92] approaches the complexity of teaching by incrementally refining an initially taught motion online by means of physical interaction. Therefore, an interaction model is derived that based on the interaction forces dynamically changes task priorities enabling users to correct robot motions in end-effector space or null-space during execution. However, the initial motion demonstration is still trajectory-based using the demonstration-in-configuration strategy discussed in Sect. 2.2.1 and therefore does not help to reduce the complexity of kinesthetic teaching in confined spaces.

Other work addresses the complexity of kinesthetic teaching by a key-frame-based approach such as provided by the commercially available KUKA Lightweight Robot IV [8] and a similar approach on the humanoid robot Simon [15]. Using the demonstration-in-configuration-space strategy they allow to configure constraints for movement generation and program tasks simultaneously but in a simplified way removing the complexity of trajectory-based teaching methods. While they have shown to improve the interaction experience [15], the implicit encoding of the redundancy resolution has to be taught repetitively in every demonstration as it is not separated from the task.

In contrast, the approach presented in this chapter allows to incrementally define null-space constraints for the robot which then can be reused for different tasks. As I will show in the following the entire teaching procedure requires only a few minutes even for non-experts, and the separation of task and environmental constraints allows to re-program only either of both if demanded. The interaction is designed to provide as natural interaction interfaces as possible mostly relying on haptic and acoustic feedback about the current system state.

Chapter 3

Learning, Encoding and Generalizing Redundancy Resolutions

A central idea of the proposed approach to allow intuitive and efficient teaching of redundant robots is to enable users to *provide demonstrations* of the perceived environmental constraints, rather than *explicitly modeling or programming* them. While the user's perspective during that *CONFIGURATION* phase is investigated in Chap. 4, the focus of this chapter lies on the learning point of view. With the interaction model presented in Sect. 2.3 several challenges are imposed on the learning system. Demonstration data is provided in only few training areas of the robot's workspace to reduce teaching time and avoid exhaustive sampling. These areas might be spread far apart or close together depending on the environmental constraints and user preferences. In addition, training data within these areas might vary structurally from user to user, e.g. in exploration volume and amount of samples. Hence, an important aspect in this context is the ability to generalize the constraints conveyed in the data to unknown parts of the workspace such as between training areas and beyond. Hence, the goal of this chapter is to analyze learning approaches with different generalization abilities according to these needs to answer the question, how machine learning can be utilized to efficiently and robustly learn, encode and generalize the environmental constraints from the user demonstrations. The hypothesis is that

> *machine learning approaches based on neural networks are a feasible tool to efficiently learn and generalize environmental constraints from only few demonstrations.*

In order proof this hypothesis, in the following neural network based approaches are introduced and discussed in Sect. 3.2. As the structured user interaction offers prior knowledge about the to be learned model, an adopted version of the local linear map (LLM) is proposed in Sect. 3.2.2. The considered learning algorithms then are evaluated thoroughly with respect to their generalization capabilities in Sect. 3.3. A final discussion of the presented results then closes this chapter. We begin these considerations with a brief description of the implementation of the interaction model from the machine learning view-point.

3.1 Interaction Model

Implementing the interaction model of the *CONFIGURATION* stage as described in Sect. 2.3.1 from the machine learning viewpoint relates to formalizing the learning problem and embedding the learned function into the hierarchical controller.

The recorded demonstration data $\mathcal{D}$ gathered through the human-robot interaction consists of the task space positions $\mathbf{x} \in \mathbb{R}^M$ and corresponding joint configurations $\mathbf{q} \in \mathbb{R}^N$ of the robot. The structure of the *CONFIGURATION* stage enables the tutor to demonstrate redundancy resolutions in several $k = 1, \ldots, K$ areas of the workspace each with $L^{(k)}$ data pairs $(\mathbf{x}_l^{(k)}, \mathbf{q}_l^{(k)})_{l=1,\ldots,L^{(k)}}$. Hence,

$$\mathcal{D} = \{\mathcal{D}^{(k)}\}^{k=1,\ldots,K} = (\mathbf{x}_l^{(k)}, \mathbf{q}_l^{(k)})_{l=1,\ldots,L^{(k)}}^{k=1,\ldots,K} \tag{3.1}$$

or, if the distinction between different training areas is not important,

$$\mathcal{D} = (\mathbf{x}_l, \mathbf{q}_l)_{l=1,\ldots,L}, \tag{3.2}$$

where L then refers to the total number of training samples in all recorded training areas.

As stated and motivated in Sect. 2.3.1, redundancy resolution in this thesis is encoded location-based, hence as a mapping $\mathbf{q}_c : \mathbb{R}^M \mapsto \mathbb{R}^N$ with $\mathbf{q}_c \equiv \mathbf{q}_c(\mathbf{x}_{\text{des}})$ providing preferred joint configurations $\mathbf{q}_c$ for given task space position $\mathbf{x}_{\text{des}}$. In order to simplify the notation in the following and emphasize the utilized learning algorithms as parameterized input-output mappings, I change the notation throughout this chapter: inputs (task space position) will remain denoted by $\mathbf{x}$ but outputs (joint configurations) will be denoted as $\mathbf{y}$. With this notation the learning problem refers to the question how to infer a function $\hat{\mathbf{y}}(\cdot)$ from the provided training data as an approximation of $\mathbf{y}(\cdot)$ which can be used to predict outputs $\mathbf{y}$, i.e. valid joint angles, for new inputs $\mathbf{x}$, i.e. task space positions. As will be discussed in Sect. 3.2, artificial neural networks $\hat{\mathbf{y}}(\cdot) \equiv \hat{\mathbf{y}}(\cdot, \omega)$ are used as parameterized models in this thesis where ω refers to the set of adjustable parameters of a particular class of neural networks characterized by learning algorithm and architecture.

The learning problem can be formalized as minimizing the mean squared error

$$E(\omega) = \frac{1}{L_{\text{tr}}} \sum_{l=1}^{L_{\text{tr}}} \|\hat{\mathbf{y}}(\mathbf{x}_l, \omega) - \mathbf{y}(\mathbf{x}_l)\|^2 = \frac{1}{L_{\text{tr}}} \sum_{l=1}^{L_{\text{tr}}} \|\hat{\mathbf{y}}(\mathbf{x}_l, \omega) - \mathbf{y}_l\|^2 \tag{3.3}$$

between a parameterized model $\hat{\mathbf{y}}(\cdot, \omega)$ and the unknown function $\mathbf{y}(\cdot)$. The latter is accessible and evaluable only by means of the training data $(\mathbf{x}_l, \mathbf{y}_l), l = 1, \ldots, L_{\text{tr}}$ which are gathered through the structured human-robot interaction. Minimization is done with respect to the parameters ω of the model. Learning is organized in epochs, as one-shot batch learning, or a mixture of both depending on the specific learning algorithm used.

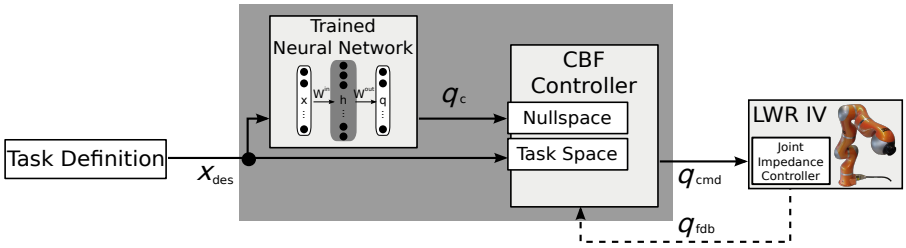


Fig. 3.1: Implementation of the *hierarchical control* concept with the learned neural network embedded into the control architecture.

Once suitable parameters ω have been found, the neural network can be embedded in the proposed hierarchical control structure as sketched in Fig. 3.1 exemplarily with an extreme learning machine (ELM, cf. Sect. 3.2.1). The robot platform used for evaluation in this thesis is the KUKA Lightweight Robot IV [5] integrated into our system prototype FlexIRob [30]. For system integration reasons (avoid control mode switching on the low-level controller) we utilize the joint impedance controller [25] provided by the robot. The hierarchical controller is realized as a closed-loop controller, incorporating feedback about the current joint position $\mathbf{q}_{\text{fdb}}$. Embedded into this control architecture, the learned redundancy resolution allows to execute arbitrary task space movements while simultaneously complying to the user-taught constraints.

3.2 Learning Redundancy Resolutions with Neural Networks

Learning redundancy resolutions from demonstrations is inspired by research on inverse kinematics learning of humanoid robots. Due to the high number of DoFs of such robots, research in this direction typically needs to tackle the problems of redundancy resolution and computational efficiency. In [26] the authors use a local learning approach called Locally Weighted Projection Regression (LWPR), which approximates a global, non-linear function by means of piecewise linear models, and which has been used on a wide variety of learning tasks such as learning inverse dynamics of robot manipulators [93] or generating movement primitives for a soccer-playing robot dog [94]. From a neural learning perspective, this algorithm can be seen as an abstraction of local linear maps (LLM) [95] and Kohonen's self-organizing maps [96] where individual neurons are responsible for only a local subset of the network's input space. Although the LLM approach was also successfully utilized in [97] for bootstrapping inverse kinematics of a high-DoF robot arm, the authors in [27] argue that "[b]y definition, these schemes are local and can not extrapolate to untrained regions". In contrast, that work proposes a global learning scheme employing a fully recurrent neural network based on the idea

of reservoir computing [98, 99] constituting a clever combination of random projections in high-dimensional, spatial-temporal representations and efficient, linear read-out learning. By this means the authors could demonstrate that learning of whole body control [27] and pose-constraint movements [100] of humanoid robots is feasible with high accuracy and generalizes even to completely untrained inputs.

However, the domain of this theses strongly differs from the experiments reported in the aforementioned contributions. Here, we deal with learning redundancy resolutions and not specifically inverse kinematics. Hence, high accuracy of the learned inverse kinematics mapping is not mandatory and is not even feasible in this context. Per design, the proposed, intuitive human-robot interaction needs to be efficient and can not be exhaustive like e.g. grid-based sampling. The user typically visits only few relevant parts of the entire task space, but does not even try to sample all regions systematically. Hence, the learner should rather generalize from few demonstrations to the entire obstructed workspace. This is different to the work mentioned above where either a "motor babbling" approach [26] generates enough training data, or generalization is tested only in small, obstacle-free regions in front of the upper part of the humanoid body [27].

Therefore, both learning paradigms (local vs. global) are analyzed in several experiments. These experiments will investigate the generalization abilities of the approaches concerning interpolation and extrapolation as well as how they perform with inconsistent and very few training data.

3.2.1 Global Learning with Random Projections

In the last decade machine learning techniques based on random projections have been proposed and have attracted a lot of attention. Reasons for that were limitations of the present state-of-the-art methods in learning with neural networks. Those methods such as the well-established multi-layer-perceptron (MLP) [101] typically rely on building a stack of multiple, consecutive, internal representations and the application of gradient descent allowing to adapt all parameters of the network according to a given supervised error signal. However, they are subject to fundamental computational limitations. Gradient descent and tedious error-backpropagation through multiple hidden layers suffer from high computational complexity, vanishing gradients and the problem of getting stuck in local optima or plateaus of the given error function [101].

In contrast, machine learning with random projections is typically based on a single layer of fixed, randomly generated features, i.e these features are not adapted during learning. As opposed to random projections for dimensionality reduction, which have been considered much earlier [102, 103], it is characteristic for such new approaches to use high-dimensional projections. Learning in these approaches is restricted to a linear read-out of the hidden representations and is typically solved with regression methods. This procedure is easy-to-use and computationally highly efficient, allowing for very efficient processing of large and

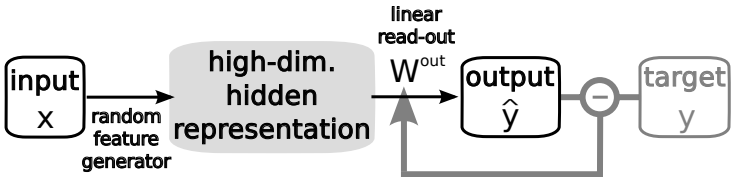


Fig. 3.2: Schematic view on machine learning with random projections. By means of a fixed, random feature generator the input $\mathbf{x}$ is transformed into a high-dimensional hidden-state representation which then is read-out linearly with a projection matrix W^{out}. Supervised learning affects only the parameters W^{out}.

high-dimensional data sets [104]. Two main principles have been developed in this context: extreme learning machines (ELMs) [105] and the reservoir computing (RC) paradigm [106] such as the backpropagation-decorrelation model [98] briefly described below. The main difference between both frameworks lies in the recurrence of the underlying neural network. Whereas reservoir networks are recurrent and comprise a so-called "reservoir" of randomly interconnected neurons, ELMs are single hidden layer feed-forward networks. Still, both approaches randomly initialize the free parameters of the feature generating part of their processing model and restrict learning to linear methods of the output function as sketched in Fig. 3.2. In [107, 108] we investigated the relations between recurrent networks and their corresponding feed-forward "alter egos". It is worth mentioning, that - despite often stated differently in the literature - model selection is also an issue for this approaches, i.e. generalization capabilities differ with varying hyper parameters such as the number of hidden units. However, we also found synergies between the recurrence in reservoir networks and the IP-learning rule as introduced in the following to remedy this situation.

Backpropagation-Decorrelation (BPDC)

Inspired by the work [27] of Rolf *et al.*, a first implementation of the approach proposed in this thesis was done in [109] using a backpropagation-decorrelation network (BPDC) [98]. It consists of a recurrent neural network (RNN) architecture (cf. Fig. 3.3) combined with efficient supervised read-out learning and local unsupervised adaption of the neurons' excitability. Besides the input and output layer $\mathbf{x} \in \mathbb{R}^M$, $\mathbf{y} \in \mathbb{R}^N$, the network architecture comprises a fixed hidden recurrent neural network layer $\mathbf{h} \in \mathbb{R}^R$, the reservoir. In order to derive the network's update equations, the holistic network state at time step k is denoted by

$$\mathbf{z}(k) = (\mathbf{x}(k)^T,\, \mathbf{h}(k)^T,\, \mathbf{y}(k)^T)^T,$$

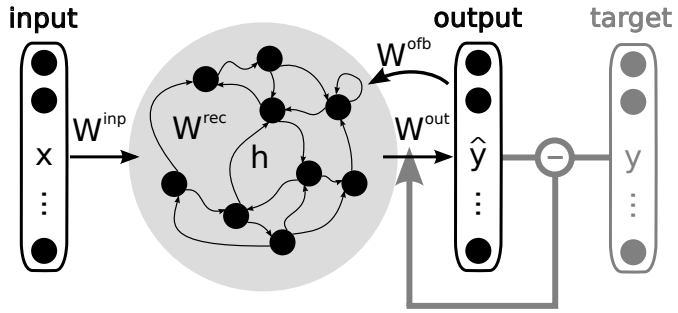


Fig. 3.3: The reservoir network architecture with respective input $\mathbf{x}$, hidden $\mathbf{h}$, and output $\mathbf{y}$ layer. Learning affects only the output weights W^{out}.

and the matrix $\mathsf{W}^{net} \in \mathbb{R}^{(R+N)\times(M+R+N)}$ capturing all connection strength sub-matrices between neurons in the RNN is defined by

$$\mathsf{W}^{net} = \begin{pmatrix} \mathsf{W}^{\text{in}} & \mathsf{W}^{\text{rec}} & \mathsf{W}^{\text{fdb}} \\ 0 & \mathsf{W}^{\text{out}} & 0 \end{pmatrix}.$$

The network's dynamics are then governed by the following update equations

$$\mathbf{s}(k+1) = (1-\Delta t)\,\mathbf{s}(k) + \Delta t\,\mathsf{W}^{net}\mathbf{z}(k) \tag{3.4}$$

$$\mathbf{z}(k) = \mathbf{f}(\mathbf{s}(k)), \tag{3.5}$$

where for small Δt continuous time dynamics are approximated, and $\Delta t = 1$ results in the standard discrete dynamics which is used in this thesis. The states $\mathbf{z}$ are obtained by applying activation functions $z_i = f_i(s_i)$ component-wise to the neural activations s_i. For the reservoir neurons parametrized logistic activation functions $f_i(x) = (1 + \exp(-a_i x - b_i))^{-1}$ are used while the output neurons have the identity as activation function, i.e. are linear neurons. During the learning process, only the output connections $\mathsf{W}^{\text{out}} \in \mathbb{R}^{N\times R}$ projecting to the output neurons are trained by error correction as illustrated in Fig. 3.3, i.e. the adjustable parameters according to the supervised learning are $\omega = \mathsf{W}^{\text{out}}$ for this approach. All other weights are initialized randomly according to a uniform distribution with small weights and remain fixed. Hence, the random-feature-generating part in this approach consists of a combination of a random projection of the inputs to the hidden states via the randomly initialized input matrix W^{in} and the subsequent use of the non-linear activation functions f.

Given the training data, supervised learning of the read-out weights W^{out} is done by the backpropagation-decorrelation learning rule, an efficient supervised online training scheme introduced in [98], which for this network configuration simplifies to

$$\Delta w_{ij}(k) = \frac{\eta}{\|\mathbf{h}(k-1)\|^2}\, h_j(k-1)\,(y_i(k) - \hat{y}_i(k)), \tag{3.6}$$

where $y_i(k)$ is the desired target value of output neuron i at time step k. Additionally to the supervised synaptic learning, an unsupervised learning method called intrinsic plasticity (IP) is used. IP was first introduced in [110] in the context of feed-forward networks. The biologically inspired learning rule changes the neurons' gains a_i and biases b_i of the logistic activation functions in order to optimize information transmission within the network. That is, neurons should be active only sparsely according to an approximately exponential distribution. In the following the online gradient rule with learning rate η_{IP} and desired mean activity μ is shown:

$$\Delta b_i(k) = \eta_{IP} \left(1 - \left(2 + \frac{1}{\mu}\right) h_i(k) + \frac{1}{\mu} h_i(k)^2\right), \tag{3.7}$$

$$\Delta a_i(k) = \eta_{IP} \frac{1}{a_i(k)} + s_i(k) \Delta b_i(k). \tag{3.8}$$

It is worth mentioning, that although this self-adaptation rule adapts the random-feature generating part of the network, this learning process is unsupervised, i.e. depends only on the network's inputs and not on the supervised error signal. Therefore, no tedious error-backpropagation is involved here. It is local in time and space, thus efficient to compute, and therefore does not contradict with the argumentation about random projection based approaches in the last section. In [111] IP has been applied to reservoir networks and was shown to tune the reservoir's neurons to an optimized working regime while improving robustness of the networks' performance. In addition, the work in [107] investigates the role of IP in static reservoir networks and reveals synergy effects with the recurrence of these networks improving the robustness of model selection parameters.

Learning with the BPDC network starts after all training data has been recorded by the human tutor during the *CONFIGURATION* stage and is organized in epochs. Each epoch constitutes a sweep through the training data set while adapting the network according to Eq. (3.6) and Eq. (3.7) and a subsequent sweep for online evaluation of that epoch. The learning stage stops if a certain stopping criterion such as a maximum number of epochs N_E or a reasonable measured output error $E(\omega)$ is achieved.

Results for learning of redundancy resolutions with backpropagation-decorrelation networks will be presented in Sect. 3.3.1.

Extreme Learning Machine (ELM)

In 2004, Huang *et al.* introduced the extreme learning machine (ELM) [105, 112]. As depicted in Fig. 3.4, it consists of a three-layered feed-forward neural network with a high-dimensional hidden layer providing a random projection of the input through fixed randomly initialized input weights W^{in}. Learning is reduced to computing a simple generalized inverse by linear regression. ELMs thus train much faster than traditionally trained backpropagation networks, and even performed

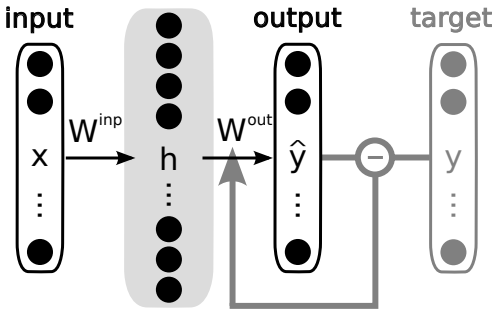


Fig. 3.4: The ELM learning scheme consisting of a single hidden layered feed-forward neural network with. Again learning affects only the output weights W^{out}.

better on most of the tasks reported in [105]. It has also been shown in [105] that a randomly created ELM with hidden layer size R is able to perform any mapping consisting of R observations. ELMs are thus in theory universal function approximators, like e.g. MLPs, if permitting an arbitrary number of training samples and any hidden-layer size. The activations of the ELM input, hidden and output neurons are again denoted by $\mathbf{x}$, $\mathbf{h}$ and $\mathbf{y}$, respectively, and the connection strengths are collected in the matrices W^{in} and W^{out} denoting the input and read-out weights. Like in the backpropagation-decorrelation approach, parameterized activation functions $f_i(x) = (1 + \exp(-a_i x - b_i))^{-1}$ transform the synaptic sum of the hidden-layer neurons component-wise to a high-dimensional hidden state representation

$$\mathbf{h}(k) = \mathbf{f}(\mathsf{W}^{\text{in}}\mathbf{x}(k)), \tag{3.9}$$

and the network's output is given by the linear read-out function

$$\hat{\mathbf{y}}(k) = \mathsf{W}^{\text{out}}\,\mathbf{h}(k). \tag{3.10}$$

Again, the input weights and the activation function parameters a_i, b_i are initialized randomly, typically according to a uniform distribution, and stay fixed during learning.

Given the set of training examples $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_{L_{\text{tr}}}, \mathbf{y}_{L_{\text{tr}}})$ the ELM is trained by minimizing the mean squared error in Eq. (3.3) between the target outputs and the actual network outputs by means of linear regression on the output weights, i.e. as with the BPDC approach all learning parameters are $\omega = \mathsf{W}^{\text{out}}$. The network's states $\mathbf{h}_l$ as well as the desired output targets $\mathbf{y}_l$ are collected in a state matrix $\mathsf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_{L_{\text{tr}}})$ and a target matrix $\mathsf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{L_{\text{tr}}})$ for all $l = 1, \ldots, L_{\text{tr}}$, respectively. The minimizer of Eq. (3.3) is the least squares solution

$$\mathsf{W}^{\text{out}} = \mathsf{Y}\mathsf{H}^{\dagger}, \tag{3.11}$$

where $\mathsf{H}^{\dagger}$ is the pseudo-inverse of the matrix H. However, as with other neural network approaches and as intensively discussed in [107] model selection is an

issue also for ELMs. The networks tend to overfit the data, particularly if the task does not comprise many training samples. In order to circumvent this issue, an output regularization technique, which is based on Tikhonov regularization [113] and well known under the terms *ridge regression* or *weight decay*, can be applied. It introduces a regularization parameter ε in the error function

$$E(\mathsf{W}^{\text{out}}) = \frac{1}{L_{\text{tr}}} \sum_{l=1}^{L_{\text{tr}}} \|\hat{\mathbf{y}}(\mathbf{x}_l, \mathsf{W}^{\text{out}}) - \mathbf{y}_l\|^2 + \varepsilon \|\mathsf{W}^{\text{out}}\|^2, \tag{3.12}$$

which penalizes large weights W^{out}. The regularized minimizer then becomes

$$\mathsf{W}^{\text{out}} = \mathsf{Y}\mathsf{H}^{\mathsf{T}} \left(\mathsf{H}\mathsf{H}^{\mathsf{T}} + \varepsilon \mathbb{I} \right)^{-1}, \tag{3.13}$$

where $\mathbb{I} \in \mathbb{R}^{R \times R}$ refers to the identity matrix. By this means, strong oscillations of the network's output function are avoided and the risk of over-fitting is reduced [107]. As a side effect, this solution is also numerically more stable because of the better conditioned matrix inverse. A suitable regularization parameter ε needs to be chosen carefully since too strong regularization results in poor performance, and on the other hand, a too small value of ε does not avoid over-fitting [107]. The influence of this parameter in the context of this thesis will be elaborated in Sect. 3.3.3.

3.2.2 Local Learning with Local Linear Maps (LLM)

In contrast to global learning schemes such as the ELM or BPDC approach, when learning motor-coordination with local linear maps [114], adapting to new training samples x, y only has local influence on the to-be-learned function $\hat{\mathbf{y}}(\cdot, \omega)$. Locality hereby refers to a distance metric $d_X(\cdot, \cdot)$ in input space i.e. task space. Learning in one area of the task space does not affect the learned mapping $\hat{\mathbf{y}}$ in the rest of the workspace, which can be beneficial e.g. for re-learning the redundancy mapping in one part of the workspace without changing the already learned mapping anywhere else.

The LLM approach used in this thesis is motivated by the approach in [97] and consists of a set of linear functions $\mathbf{h}^{(k)}(\mathbf{x})$, which are centered around prototype vectors $\mathbf{c}^{(k)}$ in task space and active only in their close vicinity. This is achieved by defining a distance parameter d and employing Gaussian responsibility functions $g(\mathbf{x})$ around their centers $\mathbf{c}^{(k)}$. The output of the final function $\hat{\mathbf{y}}$ is calculated as a linear combination of the local linear functions $\mathbf{h}^{(k)}$ weighted by their respective

responsibility and normalized with a soft-max normalization term $n(\mathbf{x})$:

$$\hat{\mathbf{y}}(\mathbf{x}) = \frac{1}{n(\mathbf{x})} \sum_{k=1}^{K} g\left(\frac{\mathbf{x}-\mathbf{c}^{(k)}}{d}\right) \mathbf{h}^{(k)}\left(\frac{\mathbf{x}-\mathbf{c}^{(k)}}{d}\right) \tag{3.14}$$

$$n(\mathbf{x}) = \sum_{k=1}^{K} g\left(\frac{\mathbf{x}-\mathbf{c}^{(k)}}{d}\right) \tag{3.15}$$

$$g(\mathbf{x}) = e^{-\|\mathbf{x}\|^2} \tag{3.16}$$

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathsf{W}^{(k)}\mathbf{x} + \mathbf{b}^{(k)} \tag{3.17}$$

The centers $\mathbf{c}^{(k)}$, the number K of models and the distance parameter d are hereby determined by the structured user interaction described in Sect. 3.1: In contrast to [97] where d is fixed a priori and new models as well as their centers $\mathbf{c}^{(k)}$ are created online and automatically depending on the data distribution in task space, in this thesis the models are created on demand by the user during the interaction before the learning phase. Each time the human tutor guides the robot to a new training area (*APPROACHING*) and starts recording training data (*RECORDING*) with an initial position $(\mathbf{x}^{\text{init}}, \mathbf{y}^{\text{init}})$, a new model is created at center $\mathbf{c}^{(K+1)} = \mathbf{x}^{\text{init}}$ and initialized with $\mathbf{b}^{(K+1)} = \mathbf{y}^{\text{init}}$ as well as $\mathsf{W}^{(K+1)} = 0_{N\times M}$. After all training data in all training areas has been recorded by the user, a suitable distance parameter d is calculated from the centers $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(K)}$ by

$$d = \max_{i,j=1,\ldots,K} \left\{ d_x(\mathbf{c}^{(i)}, \mathbf{c}^{(j)}) \mid \forall k \neq i,j : \ (\mathbf{c}^{(i)} - \mathbf{c}^{(k)})^{\mathsf{T}}(\mathbf{c}^{(j)} - \mathbf{c}^{(k)}) \geq 0 \right\}. \tag{3.18}$$

As illustrated in Fig. 3.5, d refers to the diameter of the largest of all Thales circles around two centers $\mathbf{c}^{(i)}, \mathbf{c}^{(j)}$ that do not comprise any other centers $\mathbf{c}^{(k)}$. This choice accurately relates d to the distribution of training areas in task space which is unkown beforehand. A fixed, too small value of d is inappropriate when the human tutor provides only few training areas which are far apart from each other because it results in a winner-takes-all approach with probably abrupt changes at the borders of the Voronoi regions. Contrarily, in the case of many training areas, which are close to each other, a fixed, too high value of d deteriorates the locality of the approach, since the influence of the local models $\mathbf{h}^{(k)}$ is scaled by the responsibility functions $g(\mathbf{x})$ to a larger area of the workspace.

Hence, the free parameters left for supervised online learning of this model are the weights and the bias of each linear model k, i.e. $\omega = \{\mathsf{W}^{(k)}, \mathbf{b}^{(k)}\}_{k=1,\ldots,K}$. Learning is organised in epochs as discussed in Sect. 3.2.1, and the parameters $\mathsf{W}^{(k)}$ as well as $\mathbf{b}^{(k)}$ are updated in each time step according to a gradient descent approach on the quadratic error function $E(\omega)$:

$$\Delta\mathsf{W}^{(k)} = -\eta \frac{\partial E(\omega)}{\partial \mathsf{W}^{(k)}} \qquad \text{and} \qquad \Delta\mathbf{b}^{(k)} = -\eta \frac{\partial E(\omega)}{\partial \mathbf{b}^{(k)}} \tag{3.19}$$

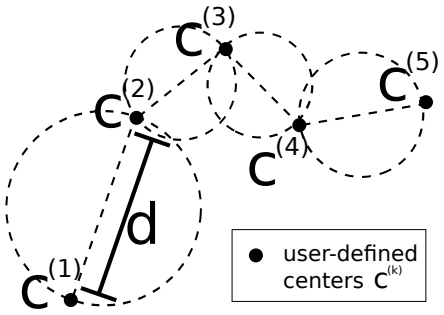


Fig. 3.5: Two-dimensional illustration of how the distance parameter d is obtained. The Thales circles of $(\mathbf{c}^{(1)}, \mathbf{c}^{(2)})$, $(\mathbf{c}^{(2)}, \mathbf{c}^{(3)})$, $(\mathbf{c}^{(3)}, \mathbf{c}^{(4)})$ and $(\mathbf{c}^{(4)}, \mathbf{c}^{(5)})$ are the only ones fulfilling the right-hand side of Eq. (3.18). The value of d is given as the diameter of the largest among those.

3.3 Generalization Capabilities

In the following, I report several experiments that are conducted to assess the generalization abilities of the proposed learning approaches from the machine learning point of view. An interesting issue, particularly with respect to the human-robot interaction, is the number and quality of the provided training data. On the one hand, the interaction should be as intuitive and efficient as possible. Explicit requirements like a specific form, structure or number of training data are undesirable as well as exhaustive sampling of the robot's workspace. On the other hand, the provided training data should exhibit enough characteristics about the user-defined null-space constraints to enable the learning approaches to infer a generalized model of these. By design of the *CONFIGURATION*, the robot's workspace will be partitioned into areas where training data has been provided. The generalization abilities - of the entire robotic system itself and of the different learning approaches compared to each other - will be evaluated within the training areas (reproduction), in between these areas (interpolation) and beyond (extrapolation). For that purpose, the following generalization measurements will be used throughout his section:

Task space accuracy $E_{\mathbf{task}}$: To what extend is the task performance of the whole system in Cartesian space affected by the null-space constraint taught by the user? Given a set of points in task space $\mathbf{x}_1, \ldots, \mathbf{x}_L$ being set as targets and fed to the controller architecture described in Sect. 3.1 to be realized by the robot as actual measurable end-effector positions $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_L$, the task space accuracy can be measured as the mean Euclidean distance $\|\cdot\|$ between the corresponding points:

$$E_{\text{task}}(\mathbf{x}_1, \ldots, \mathbf{x}_L) = \frac{1}{L} \sum_{l=1}^{L} \|\mathbf{x}_l - \hat{\mathbf{x}}_l\| \tag{3.20}$$

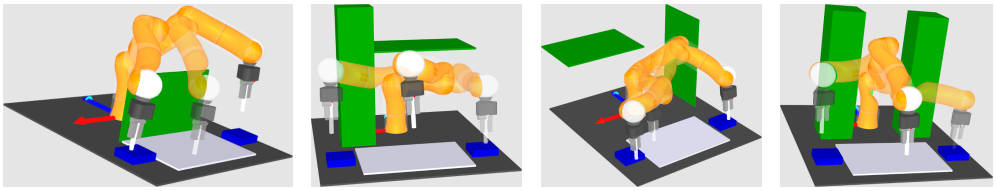
The hypothesis of this thesis is that - independent of the particular learning algorithm - the task-space performance is not affected since the hierarchical controller prioritizes the Cartesian task over the joint-space task.

Monitoring collisions E_{coll}: This measurement is used to evaluate how well the user-demonstrated null-space constraints are learned and generalized to unknown areas of the workspace. A direct calculation of the "deviation from trained user-constraints" in joint-space is not suitable because (a) the desired constraints would have to be expressed formally in joint-space, (b) a corresponding meaningful distance measurement would have to be defined, and (c) ground truth data for that kind of evaluation is not available in untrained areas. Instead, intending that the taught null-space constraints convey information about how to avoid the obstacles in the robot's workspace, by monitoring the number of collisions between the robot manipulator and its environment during a desired end-effector trajectory a more tangible measurement is used. Collisions are detected automatically by the simulator described in Chap. A - either purely in simulation or while the actual robot performs the desired motion. Hence, evaluations with this measurement again are on the whole-system level.

3.3.1 Task Space Accuracy vs. Null-space Constraints

The first evaluation focuses on the generalization ability on the whole-system level and was conducted in [109]. The adressed questions are: (i) Does the system respect the taught null-space constraints and (ii) to what extend is the task performance in Cartesian space affected by those constraints, i.e. what is the task performance?

For the experiment, a various set of workspace configurations is defined as depicted in Fig. 3.6. Each of them comprises a left and a right training area (above the blue boxes, respectively), which are placed at both ends of the robot's evaluation workspace. Hence, each scenario contains an untrained area in between (*interpolation area*) where the system has to generalize the taught constraints to unseen data points. Four different scenarios (A-D) are defined that comprise reasonable obstacle setups that might occur in real world applications: In setup A, depicted in Fig. 3.6(a), the robot has to reach over an obstacle in its entire workspace. In setup B, cf. Fig. 3.6(b), the robot has to reach around an obstacle in the left training area and in the rest of the workspace the elbow movement is constrained in height. Setup C, cf. Fig. 3.6(c), constrains the height of the elbow movement on the left side and forces an "elbow up" configuration on the right side to avoid collisions with the wall. The robot has to perform a change of its elbow configuration while moving through the interpolation area. Setup D, cf. Fig. 3.6(d), is the most difficult of the four depicted due to its narrow obstacle setup. The robot has to reach around obstacles in both training areas and has to move the elbow



(a) Setup A (b) Setup B (c) Setup C (d) Setup D

Fig. 3.6: Different workspace configurations used for evaluation. During *CONFIGURATION* desired elbow configurations are taught to the robot through kinesthetic teaching to avoid collisions with obstacles.

Tab. 3.1: Hyper-parameters for BPDC network construction and learning.

Reservoir Size	300	
Connection	sparseness	init. Range
Input-Reservoir	0.2	0.1
Reservoir-Reservoir	0.02	0.02
Reservoir-Output	0.2	0.1
Learning	η	
IP	0.00015	
BPDC	0.015	

to a large extend from “elbow left” to “elbow right” configuration while moving through the untrained interpolation area.

Teaching the null-space constraints for these setups is done as described in Sect. 3.1 on the real robot system, while monitoring collisions in simulation: During the *CONFIGURATION* stage each of the training areas is explored in a 3D helix-like trajectory, that consists of 150 up to 300 points. The recorded training data reflects the respective desired elbow configurations (e.g. “elbow down” or “elbow left”) determined by the obstacles in the robot’s workspace. In particular, any collisions during kinesthetic teaching in all our setups is avoided.

The neural network approach utilized in this experiment is the backpropagation-decorrelation approach. After recording, the data is used to train the BPDC network as described in Sect. 3.2.1 with the parameters given in Table 3.1. Learning is organized in epochs and conducted until the MSE of the network output drops below a threshold of 0.2 but limited to a maximum number of $N_E = 1000$ epochs. The threshold was obtained manually from previous experiments and seems to constitute a reasonable lower bound. The hyper-parameters of the BPDC approach are based on a proposal by [27] and [100] for controlling a humanoid robot respecting specific constraints. After learning, the network is embedded into the system’s control architecture as described in Sect. 3.1.

For a systematic evaluation of the addressed questions (i) and (ii) the robot’s

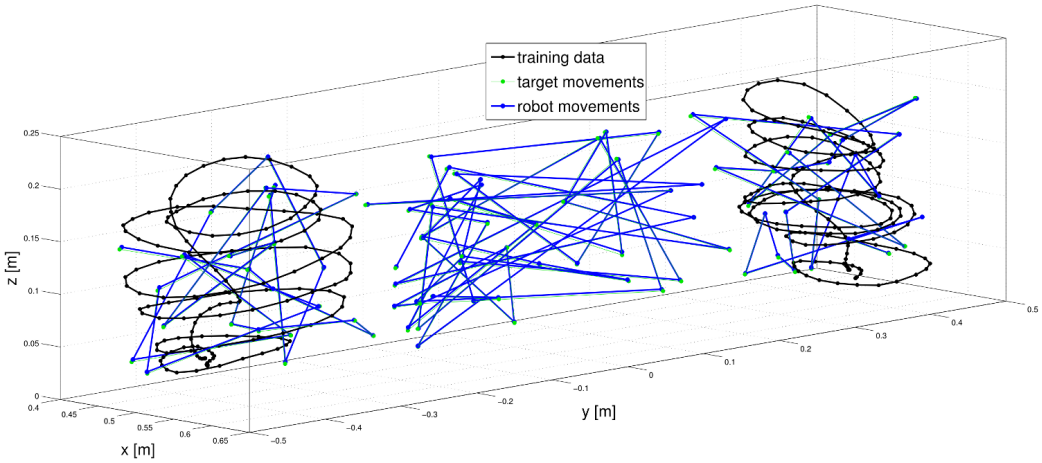


Fig. 3.7: Illustration of training data and evaluation trajectories of setup A in left and right training area as well as interpolation area.

workspace is partitioned into three areas as illustrated in Fig. 3.7. The left and right training areas are defined by axis-aligned bounding boxes around each of the two training data trajectories. The interpolation area is defined by the area between those boxes. In each of the setups (A-D), we generate straight 3D trajectories between 100 random points, 25 per training area and 50 for the interpolation area, and evaluate the task-space accuracy E_{task} as well as the number of collisions E_{coll} in the respective area. Note, that the evaluation is done on the real robot with the above mentioned simulation visualizing the internal model of the robot and checking for collisions with obstacles.

The results are summarized in Tab. 3.2. Concerning the task space accuracy, it is worth mentioning that the average error E_{task} of the entire system is about 2 mm. As hypothesized, these values are measured independent of the evaluated area and even of the evaluated setup, and hence independently of the taught constraints. The hybrid control scheme consisting of the learned mapping $\hat{\mathbf{y}}(\cdot, \omega)$ combined with the analytical, hierarchical controller ensures a reasonable and reliable task-space performance. Concerning the ability to learn the null-space resolution, the results show that the taught elbow configurations are learned and securely retrievable from the BPDC network. While performing robot movements in the training areas no collisions were detected in all four evaluation setups A-D. Furthermore, in the setups A, B and C the redundancy resolution, and hence the taught elbow configurations, is generalized by the learned network also to the interpolation area well enough, so that no collision was recorded either. The robot's behavior in that area is represented by the center robot arm in each of the setup visualizations in the Fig. 3.6. Interestingly, in setup D the robot collided during 16 of the performed 50 trajectories in the interpolation area.

In order to discuss these results, we have to distinguish the task difficulties of

Tab. 3.2: Experimental results of the evaluation of task space accuracy E_{task} and number of collisions E_{coll} in the obstacle setups A-D.

Setup	**Training**			**Evaluation**					
				task space error E_{task} [mm]			**collisions E_{coll}**		
	points	epochs	MSE	left	right	interp.	left	right	interp.
A	$177 + 195$	202	< 0.2	2.0 ± 0.6	1.6 ± 0.8	1.9 ± 0.5	0	0	0
B	$250 + 232$	237	< 0.2	1.8 ± 0.6	2.0 ± 0.7	1.9 ± 0.5	0	0	0
C	$125 + 155$	275	< 0.2	2.0 ± 0.6	1.6 ± 0.7	1.7 ± 0.6	0	0	0
D	$159 + 163$	1000	0.2017	1.9 ± 0.6	1.6 ± 0.7	1.6 ± 0.5	0	0	16

the designed workspace setups. The design of both setups A and B determines the desired elbow configurations in left and right training area to be very similar, and therefore generalization in between those areas is fairly easy. Setup C constitutes a problem, where the robot has to switch between two different elbow configurations, from "elbow down" to "elbow up" and the other way around while moving through the untrained area. The results related to this setup greatly show the generalization abilities of the learner embedded into the system's control architecture. The BPDC takes the "right guess" for a good, i.e. collision-free, redundancy resolution in between the two training areas even though there was no data provided. In setup D, the task difficulty is further increased due to stronger constraints induced by the narrow obstacle setup. The robot has to change its elbow configuration from "elbow left" and "elbow right" within the interpolation area. As a result, during 16 of the 50 performed trajectories collisions with the environment occurred. However, it is worth mentioning that still the BPDC network takes the "right guess" for resolving the manipulator's redundancy as can be observed from the visualized robot arm in the interpolation area of Fig. 3.6(d) but is only not very accurate in generalizing them. We point out that the distance between the two training areas is quite large with approx. 0.74 m and therefore expecting high accuracy in the redundancy resolution according to this very narrow obstacle setup is unrealistic. Learning in our approach is model-free and thus purely data-driven. Strong null-space constraints with the need for high accuracy in a workspace area have to be present in the training data. As a proof of concept to that argument we repeated training and evaluation of setup D, but now using three training areas: a left, right and an additional third one in the former interpolation area where the collisions occurred. The results of this setup D* show that the BPDC network benefits from the additional training area, as the number of collisions in the interpolation area was reduced from formerly 16 to 3, still by teaching the constraints within less than 5 minutes.

3.3.2 Number of Training Areas and the Reachable Workspace

The results presented above indicate that the selected redundancy resolution of the system can be improved in terms of collision avoidance by providing more training areas by the user. Naturally the question arises how the system's generalization

abilities develop with the number of training areas. It is one major goal of the *CONFIGURATION* stage to enable the robot to move in the predefined, confined workspace. The trained system shall reach at best all targets that are theoretically accessible without colliding with the environment. How many training areas are needed to enable the system to move collision-free in the workspace? In the following an experiment is conducted to answer this question. For that purpose a more valuable measurement is devised, namely the reachable workspace of the robot as described in detail below. The following results have been published partly in [115].

Two obstacle setups are used for this experiment which are similar to those in the last section as shown in Fig. 3.8. Setup I consists of a prolate box measuring $0.2 \times 0.2 \times 0.8\,\mathrm{m}$ placed beside the robot arm and a board mounted in the height of $z = 0.55\,\mathrm{m}$ partly constraining the height of the workspace. Setup II consisting of two boxes as the one in setup I which are placed side by side with the robot arm defining a very narrow gap left for the robot to move. Given one of these experimental setups, a human tutor conducts the kinesthetic teaching process according to the *CONFIGURATION* stage as described in Sect. 3.1. He provides training data in several areas of interest in the workspace. Again, after training the recorded data is used for training the neural network to learn the redundancy resolution. Here, the ELM approach is used as described in Sect. 3.2.1 with $\varepsilon = 1$, which was found to work reasonably but will be evaluated more systematically in Sect. 3.3.3. After learning, the evaluation phase starts for which the ELM is integrated into the control system of the robot and resolves its redundancy with a single unique solution, thereby implementing the inverse kinematic mapping from 3-D task-space to 7-D joint-space. For each obstacle setup (I or II), this process is conducted in several trials, each trial with a different number of training areas. For instance, Fig. 3.8 (left) shows setup I for which the teacher this time provided training data in only a single area of the workspace, whereas Fig. 3.8 (right) shows setup II for which data in 4 different training areas was recorded.

For an objective evaluation, we define a fixed set $T \subseteq \mathbb{R}^M$ of target points in task-space. This set is created by sampling a grid $G = [-0.9; 0] \times [-0.55; 0.55] \times [0.05; 0.8]$ equidistantly and removing all targets outside the workspace $W \subseteq \mathbb{R}^M$ of the LWR IV, i.e. $T = G \cap W$. These points[4], shown in Fig. 3.9 serve as basis of the evaluation of the learned redundancy resolution. In order to account for the obstacles in the setups I and II we need to remove all points from T that are not reachable for the robot. This obviously includes all targets located within the objects. For setup I this also includes points located above the board part of the obstacle as these are not accessible for the robot due to joint limits. This way, we create obstacle-dependent targets T_{I} and T_{II} for setup I and II, respectively. The generalization ability of the system is analyzed on the basis of these points by combining the measurements of task space accuracy E_{task} and collision monitoring

[4] Note that, as illustrated in Fig. 3.9 the origin of the world coordinate system is the mounting point of the robot's root.

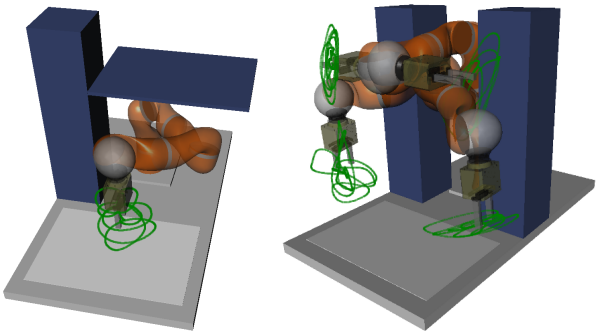


Fig. 3.8: Experimental setups. Left: scenario I with exemplary training data recorded in a single training area. Right: setup II with exemplary data recorded in 4 different areas of the workspace.

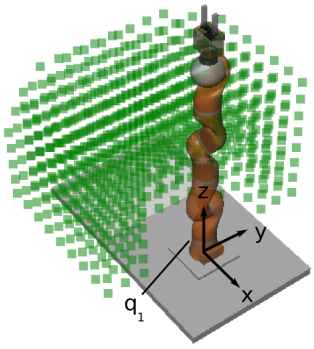


Fig. 3.9: Visualization of the obstacle-independent defined workspace T.

E_{coll}. During the evaluation phase the robot is commanded to reach for each of the targets with the learned ELM included in the inverse kinematics control of the robot. The reachable workspace analysis now evaluates the number of goals that were reached by the trained LWR IV with a precision of at least 0.01 m without colliding with the environmental obstacles. For suitability reasons, the evaluation phase is done in simulation only.

Tab. 3.3 shows the results of this analysis, displaying the number of training areas the teacher provided during the kinesthetic teaching process, how much of the defined reachable workspace T_{I} or T_{II} is already explored by that training data, and how much of the defined reachable workspace T_{I} or T_{II} is reachable for the trained robot system. As for setup I, it can be seen that already one training area provided by the human tutor as in Fig. 3.8 (left) makes 51.3% of the targets T_{I} accessible for the robot although the training data itself covers only ca. 5% of that space. In the left column of Fig. 3.10 the accessible workspace for that scenario is visualized. In that trial, where the tutor provided two training areas, the resulting workspace accessibility is not higher although more workspace is covered by the training data. The reason for that is, that the two areas selected by the teacher do not provide more implicit knowledge about the environmental scene than the single area in the trial before. This could be due to an inappropriate selection of training areas or simply that one or two training areas are not enough to cover more than approx. 50% of the workspace in that environmental scene. Having trained the system with data recorded in 4 selected areas enabled the robot to move accurately and collision-free in almost 90% of the accessible workspace while the training data covered only 23.7%. This result is also visualized in the center column of Fig. 3.10 showing that not only the free space in front of the robot is made accessible but also many targets behind the box part of obstacle and below the board part where

Tab. 3.3: Results of the reachable workspace analysis.

setup setup	training training areas	training data workspace exploration	reachable reachable workspace
I	1	4.9 %	51.3 %
	2	9.2 %	47.3 %
	4	23.7 %	89.7 %
	6	31 %	93 %
	12	36.5 %	97.1%
II	1	7.1 %	51.6 %
	2	9.7 %	64.8 %
	4	19.1 %	73.5 %
	6	31.6 %	80.8 %
	21	56.5 %	94.1 %

the LWR IV has to reach around the box or entirely fold itself below the board, respectively. This performance can be improved by providing training data in more than just 4 areas. By training in 6 different areas covering only 31% of the workspace the teacher enabled the robot to access 93% of the targets, and providing data in 12 selected areas made more than 97% of the targets in setup I reachable (see Fig. 3.10 right). The evaluation in setup II reveals a similar relation between the number of provided training data, the explored workspace by the training data and the resulting accessible workspace as indicated in Tab. 3.3 and Fig. 3.11. Providing training data in only 1 area enables the system to access more than 60% of the workspace. Even in this narrow scenario, 6 training areas are enough to make more than 80% of the workspace accessibility for the robot. Inaccessible were only those parts of the workspace where the manipulator has to move around the box obstacles and reach behind them. In order to access also this areas safely, more precise constraints need to be taught since all parts of the robot are very close to the obstacle. The last evaluation of setup II shows that this is possible by providing more training data (21 areas). As a result, the reachable workspace can be almost entirely covered ($> 94\%$).

The results of this experiment reveal the approach's capability to infer a valid, i.e. collision-free, redundancy resolution for a large part of the robot's workspace from only few training data. Hence, the taught null-space constraints are generalized well beyond the trained areas. In addition, the results reflect the scalability of the approach. With more training data the user systematically and effectively enhances the reachability of the system but no exhaustive sampling of the workspace is needed. However, even in the case that high accuracy is demanded for the manipulator's redundancy resolution to maximize the reachability, the results for 12 and 21 training areas in setup I and II, respectively, show that the proposed kinesthetic approach enables users to teach the respective constraints. Then, all relevant and sufficient areas of the workspace need to be seen for training to allow

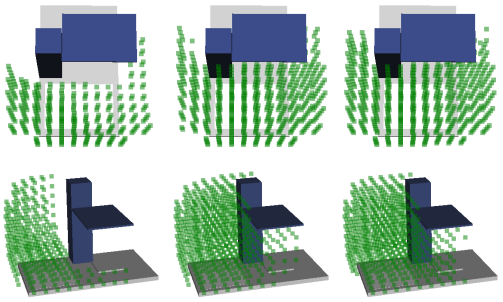


Fig. 3.10: The reachable workspace (green dots) after kinesthetic teaching of obstacle setup I. From left to right, the tutor provided training data only in 1, in 4 or in 12 areas of the workspace, respectively.

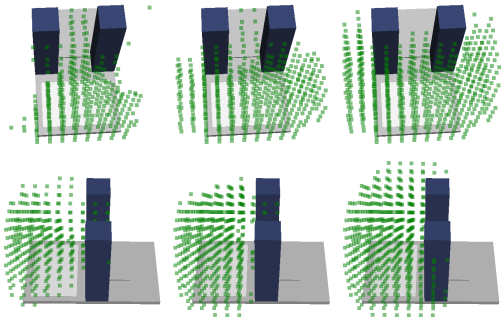


Fig. 3.11: The reachable workspace (green dots) after kinesthetic teaching of obstacle setup II. From left to right, the tutor provided training data only in 1, in 4 or in 21 areas of the workspace, respectively.

for generalization of the learned redundancy resolution to the full workspace. Still, it is worth noting that teaching this high amount of training data is manageable in less than 10 minutes.

3.3.3 Model Selection and the Number of Training Samples

As mentioned above and intensively discussed in [107, 108] machine learning algorithms suffer from the problem of model selection. This refers to the fact, that each algorithm relies on a set of hyper parameters that strongly influence the performance of the learned mapping. Qualitatively different results can be expected for different choices of these parameters, in particular with respect to the amount of training samples available.

Consider for instance the ELM approach. Here these parameters refer to the random distributions chosen for initializing the activation function parameters a_i, b_i and the input weights W^{in}, the dimensionality R of the hidden layer $\mathbf{h}$, as well as the regularization parameter ε. All have to be chosen properly to obtain good learning results. In [107] we analyzed this problem for the ELM approach and showed, that a key factor to effectively tailor the model complexity of the learned mapping is the regularization parameter ε which can prevent over-fitting of the training data but most be chosen carefully to avoid limiting the expressiveness of the mapping to much.

As for the proposed, specifically tailored variant of the LLM approach, only the learning rate η remains an hyper parameter to be tuned. The number K of models and the position $\mathbf{c}^{(k)}$ of the corresponding centers are determined by the user-defined number K of training areas and the initially chosen Cartesian position $\mathbf{x}_{\text{fixed}}^{(k)}$ of the robot during *APPROACHING*.

In the following both approaches are compared to each other with respect to these parameters (ε for ELM, η for LLM) and a varying number of training samples per area. This experiment is designed to answer the following questions:

1. To what extend do the utilized approaches depend on the selection of their hyper parameter, in particular with respect to a varying number of training samples?

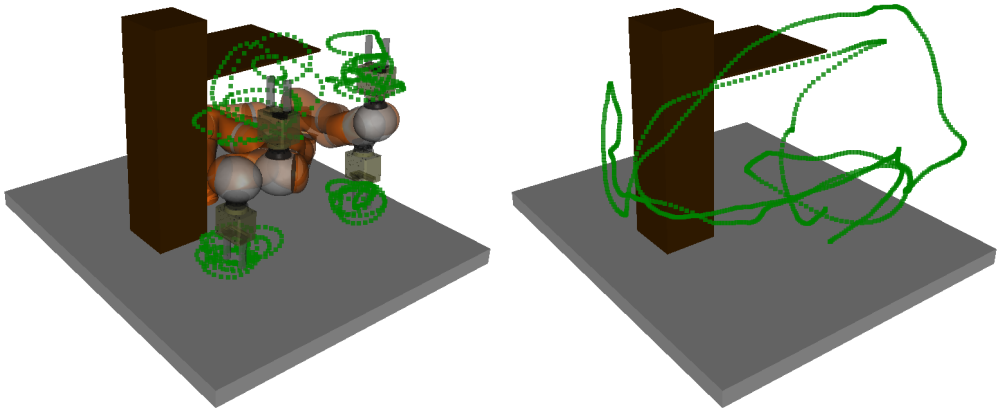
2. How well do both algorithms generalize the demonstrated constraints between training areas and even beyond, particularly compared to each other?

Note, that the BPDC algorithm is not longer considered in this section, since recurrent neural networks are best suited for temporal tasks such as time series prediction, not for static mappings such as considered in this thesis. The implementation and experiment presented in Sect. 3.3.1 therefore served as a proof-of-concept.

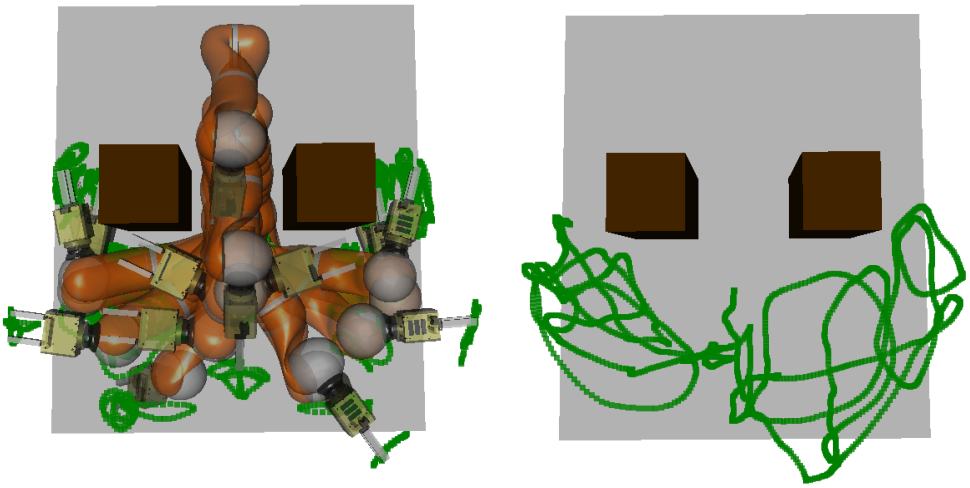
Experimental Setup

In order to answer the posed questions, both approaches are evaluated in two different setups, one with only very few training data and the other with more exhaustive sampling of the workspace. For each setup, a target trajectory was recorded on the real robot that serves as the basis for the evaluation. The first setup consists in the obstacle scenario I, which is the same scenario already investigated in the previous sections, with training data collected in only four different areas as shown in Fig. 3.12(a) together with the target trajectory. In contrast, the second scenario - obstacle setup II - represents a situation where the tutor demonstrated 25 training areas, see Fig. 3.12(b). Although the latter scenario typically relates to undesirable teaching effort, it is included in the analysis in order to investigate the generalization capabilities of both learning approaches also in such situations.

For both algorithms training in this scenario is conducted with varying number of training samples per training area, and with varying value of their respective learning parameter. The ELM approach is evaluated with $\varepsilon = 1$, $\varepsilon = 0.01$ and $\varepsilon = 0.0001$, and the LLM algorithm with $\eta = 1$, $\eta = 0.1$ and $\eta = 0.01$. The amount of training data per area is reduced successively by means of sub-sampling the originally recorded training data with a sampling rate of $s = 1$, $s = 5$ and $s = 80$. By this means, the effective number of training samples is reduced from all data over an intermediate amount of samples to only very few training samples (< 5) per training area (cf. Tab. 3.4). Hence, for each setup and each learning algorithm we obtain an 3×3 experiment design. In order to account for the random initialization of the ELM networks and the random presentation of the training samples during the learning epochs for the LLM approach, each evaluation is averaged over 10 network initializations. Throughout this analysis, the measurements E_{task} and E_{coll} will be used to investigate the generalization capabilities of the trained networks.



(a) Setup I with training data in four areas and target trajectory.



(b) Setup II with training data in 25 areas and target trajectory.

Fig. 3.12: Experimental setups used in this section with recorded training data (left) and the target trajectory that is used for evaluation (right).

		setup I (4 areas)			setup II (25 areas)		
sub-sampling s		1	5	80	1	5	80
avg. #data per area		226.5	45.5	3.2	210.3	42.5	3.1
ELM	$\varepsilon = 1$	7.0	0	2.0	0	2.5	2.5
	$\varepsilon = 0.01$	4.9	4.5	8.7	0	0	0
	$\varepsilon = 0.0001$	4.6	4.8	15.2	3.3	0	6
LLM	$\eta = 1$	0	0	0	4.6	4.7	2.6
	$\eta = 0.1$	0	0	0	0	0	0
	$\eta = 0.01$	0	0	0	1.7	1.6	1.6

Tab. 3.4: Evaluated number of collisions E_{coll}, each value averaged over 10 different network initializations.

Results Concerning Collision Avoidance

The first measurement investigated in this analysis is the number of collisions that might occurred during execution of the target trajectory. Tab. 3.4 displays the respective results.

Consider first the results for setup I. A very remarkable result is, that the proposed LLM algorithm resulted in completely collision-free movements along the target trajectory. That is, during the evaluation of none of the 90 tested networks collisions occurred. This is particularly interesting as the trajectory moves along the entire workspace even far beyond the training data, which is recorded only in the front of the robot. However, even in situations where the target trajectory moves behind the box obstacle, the robot could follow it without causing collisions with the environment. In contrast, most of the tested ELM networks had few collisions. Still, the number of collisions is remarkably low, and I report from visual inspection that these are caused only be slightly touching the obstacle. In order to get a more visual insight, Fig. 3.13 displays the utilized training data (green), the target trajectory (black dotted) and that positions $\mathbf{x}_{\text{curr}}$ of the robot, that caused the collisions (red). Hence, not the collision point between robot and obstacle themselves are plotted, but rather the position of the end-effector during that collision. As can be seen from this plot, also the ELM exhibits good performance in terms of collisions. In fact, for $\varepsilon = 1$ and $\varepsilon = 0.01$, the only end-effector positions that caused collisions are far away from the training data behind the box obstacles. This holds for all ELM networks with $\varepsilon = 1$ and $\varepsilon = 0.01$. Only training the networks with too less regularization $\varepsilon = 0.001$ and only few training data results in collisions when moving in the vicinity of the data. Surprisingly, both algorithms perform very good even with only rare training data per area. Note, that for instance when sub-sampling the original data with $s = 80$, each training area does not comprise more than 4 data points; the total amount of samples is only 12. This requires strong regularization abilities, as the result of the ELM for $\varepsilon = 0.001$ shows.

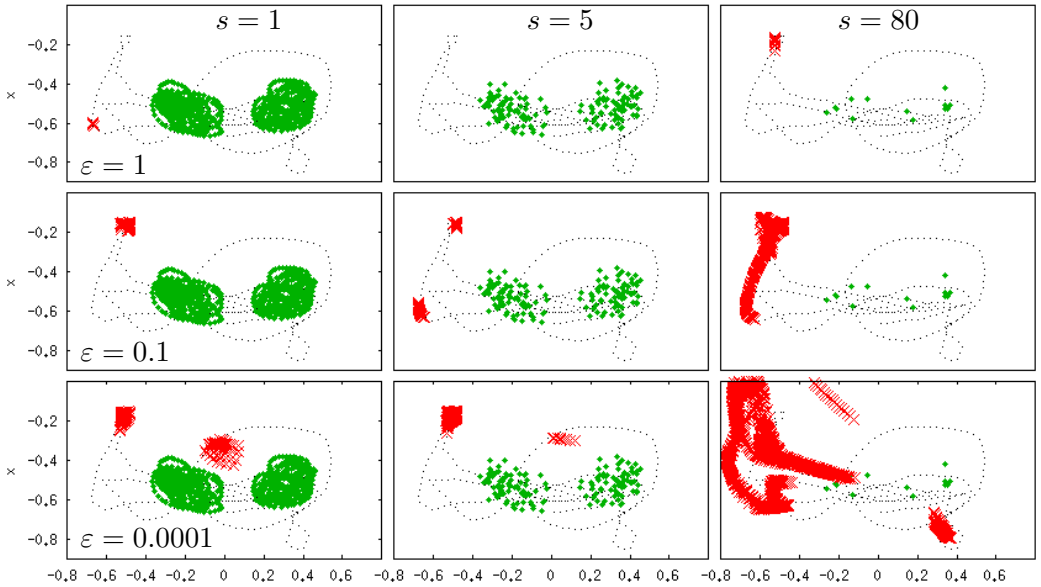


Fig. 3.13: Illustration of the encountered collisions during evaluation of the ELM networks in setup I, depending on regularization parameter ε and sub-sampling s of the training data. It plots the training data (green), the target trajectory (black dotted), and the robot positions $\mathbf{x}_{\text{curr}}$ which caused collisions (red).

Concerning the evaluation in setup II, the low values in Tab. 3.4 again indicate good generalization abilities in terms of collision avoidance. Setting $\varepsilon = 0.01$ and $\eta = 0.1$ results in collision-free execution of the target trajectory for all of the tested networks in both learning paradigms. In the other settings, only few collisions occurred. Fig. 3.14 displays these for the LLM networks, again in relation to the training data. However, again all encountered collisions relate to situations in which the robot only slightly touches the obstacle.

Results Concerning Task Space Accuracy

For the sake of brevity, I report the task space accuracy E_{task} only for the setup I, see Tab. 3.5. The results for setup II revealed qualitatively similar characteristics. Both network approaches show tracking errors higher than 5 cm, which is surprising at first glance. For static targets, the analysis in Sect. 3.3.2 attested an accuracy of < 1 cm for most parts of the workspace. Hence, the tracking errors seem to be caused by the dynamics of the recorded trajectories. In fact, I argue that this might be due to a combination of conservative joint velocity limits and high dynamic target trajectories. These have been recorded on the real robot using the *gravity compensation* controller, which allows the user to move all joints freely. As a result, in some areas of the workspace the trajectory might have been performed with a different redundancy resolution than encoded in the training data and with

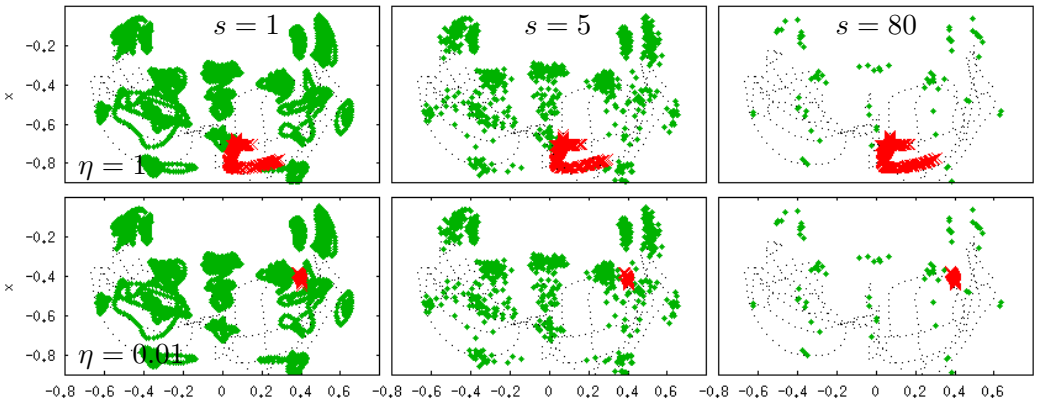


Fig. 3.14: Illustration of the encountered collisions during evaluation of the LLM networks in setup II for varying η and sub-sampling s of the training data.

		setup I (4 areas)		
sub-sampling s		1	5	80
ELM	$\varepsilon = 1$	0.08 ± 0.013	0.08 ± 0.014	0.07 ± 0.013
	$\varepsilon = 0.01$	0.07 ± 0.012	0.07 ± 0.012	0.11 ± 0.02
	$\varepsilon = 0.0001$	0.11 ± 0.018	0.09 ± 0.015	0.21 ± 0.2
LLM	$\eta = 1$	0.14 ± 0.014	0.09 ± 0.014	0.09 ± 0.014
	$\eta = 0.1$	0.09 ± 0.014	0.09 ± 0.014	0.09 ± 0.015
	$\eta = 0.01$	0.08 ± 0.014	0.08 ± 0.014	0.08 ± 0.013

Tab. 3.5: Task space accuracy in [m] during tracking of the target trajectory in setup I. Each value is averaged over 10 different network initializations.

higher joint velocities. The trained and embedded networks then generate joint angle trajectories that would allow to realize the desired Cartesian position only by violating the velocity limits, which is not permitted in the FlexIRob control architecture. Hence, the tracking errors are due to fast changes in the redundancy resolution which is necessary for complying to the environmental constraints. This is recognizable in Fig. 3.15, which shows the target trajectory in black and the executed trajectory by the robot in blue. Most of the time the robot follows the target motion accurately. Strong deviations can be observed only for z-values around 0.4 where the robot has to switch between the gripper- pointing-down and the gripper-pointing-up solution as encoded in the training data (cf. Fig. 3.12(a)).

However, despite the settings $s = 80$, $\varepsilon = 0.0001$ for the ELM and $s = 1$, $\eta = 1$ for the LLM, both approaches perform reliably and are able to track the target trajectory, although not with a high accuracy. Again, this is worth mentioning particularly with respect to the low number of training data that is provided in the $s = 80$ condition.

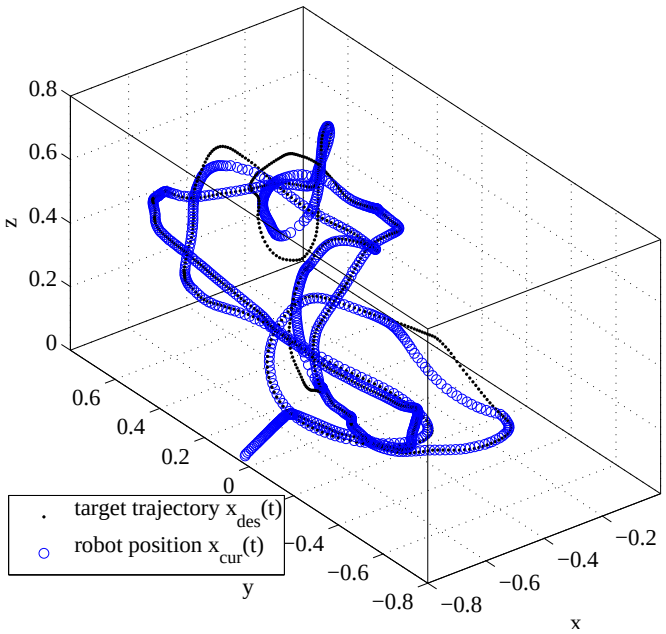


Fig. 3.15: Exemplary execution of a target trajectory with an embedded LLM ($\eta = 1$, $s = 5$) in setup I.

3.4 Discussion of Results

This chapter presented and discussed neural-network-based machine learning algorithms to efficiently learn, encode and generalize demonstrated redundancy resolutions. As discussed Sect. 3.2, this approach is motivated by research on learning inverse kinematics for humanoid robots. A first prototype is published in [29] but lacks a decent evaluation of the embedded, learned redundancy mapping. Hence, a major goal was to analyze different available learning methods.

The presented methods are purely data-driven and therefore do not require any model of the robot or environmental constraints. I mainly chose the presented ELM and the specifically tailored LLM approach since they alone span a broad range of learning characteristics. While the ELM approach is a fast and efficient regression learner which needs to process each training sample only once (even in an online-fashion [116] as shown in Chap. 6), the LLM algorithm is online capable but relies on the gradient descent technique. Hence, learning must be conducted in epochs, where training samples are presented randomly and repeatedly in order to follow the gradient. This might require high computational effort, and additionally a criterion when to stop learning. However, in contrast to the ELM, the LLM networks offer the appealing advantages of local learning methods. Hence, adapting the learned mapping has only local influence in one area or the surrounding, and does not change or deteriorate already trained constraints all over the workspace.

As a result, it allows to remove or change the learned mapping locally, e.g. to correct already demonstrated constraints or adapt the redundancy mapping to a changed environment. As for the ELM method this is not (straightforward) possible. Once trained with a particular training set, changing the learned mapping only locally or with respect to only few samples requires reprocessing of the entire data set.

Nevertheless, both approaches have been shown in this chapter to provide good generalization of the taught constraints. The learned mappings efficiently encode the learned redundancy resolutions and generalize the taught constraints even beyond the training data. As revealed in Sect. 3.3.1, the proposed approach permits to adapt a robot's redundancy resolution to a variety of different confined workspaces with only two or three training areas. Embedded into the robot's motion controller, the learned constraints are even generalized between training areas (interpolation). The analysis in Sect. 3.3.2 demonstrated that already very few training areas are sufficient to make large portions of the workspace accessible to the robot. And even though the *CONFIGURATION* as proposed in this thesis is rather designed for efficient teaching in only relevant areas of the workspace, it still allows to exhaustively sample the environmental constraints in few minutes. Similarly, the last section showed that generalization of the demonstrated constraints goes also beyond the training areas (extrapolation). Furthermore, it revealed the robustness of both approaches against parameter selection and the variety of presented training data. Properly chosen parameters ($\varepsilon = 0.01$ for ELM, $\eta = 0.1$ for LLM) allow learning of the demonstrated constraints even from less than 15 training samples in total.

Therefore, I conclude with respect to the posed hypothesis, that both ELM and LLM approach are valuable methods for the proposed interaction scheme.

Chapter 4

Teaching Redundancy Resolutions

As motivated in Chap. 1 one of the major goals of the work presented in this thesis is to enable non-expert users to program redundant robots in presence of environmental constraints. The approach presented in Sect. 2.3 derived a dedicated *CONFIGURATION* stage throughout which users demonstrate these constraints solely by means of kinesthetic teaching. The hypothesis is that

> *the users intuitively perceive the constraints and provide them implicitly to the robot by means of the recorded demonstrations.*

While the previous chapter focused on the associated machine learning problem, in this chapter I will elaborate on the hypothesis concerning the user's perspective. What kind of knowledge actually is conveyed in the demonstration data and transfered to the robot system? How is the user experience during the proposed structured user interaction? And does it enable non-experts to teach such environmental constraints successfully?

In order to answer these questions, in the first section the proposed interaction model concerning the *CONFIGURATION* phase is implemented on our interactive robot prototype FlexIRob [30] employing the KUKA Lightweight Robot IV as redundant, compliant manipulator for the physical human-robot interaction. Subsequently, I will shed light on the question what kind of knowledge is comprised in the training data and thus transferred to the robot system in the second section. It presents a visualization scheme of the taught, implicit constraints both in task and configuration space aiming to relate them to the physically present constraints in the robot's workspace. The third section presents results from the user study FlexIRob@Harting [22] in order to evaluate both teaching experience and teaching success of non-expert users utilizing the proposed interaction model during the *CONFIGURATION* stage. A decent analysis of the participants' training data will then investigate structural differences between the data provided by successful and not successful participants. Finally, the presented results are summarized and discussed.

4.1 Interaction Model

The implementation of the interaction model follows the workflow outlined in Sect. 2.3.1 (Fig. 2.5) using the described interaction triggers `on_affected` and `on_converged` to physically guide the robot to relevant areas of the workspace (*APPROACHING*) and then record training data for the learner (*RECORDING*).

The required *gravity compensation* controller π_q is implemented based on the low-level joint impedance controller of the LWR IV as illustrated in Fig. 4.1. The external forces f_{int} applied by the human operator to the LWR IV are measured by its torque sensors in each joint and generate motor torques according to an impedance based control scheme [25]. For constant $\mathbf{q}_{\text{cmd}}$ this inner control loop acts like a spring-damper system with customized spring and damping constants k and d for each joint allowing the user to physically deflect the manipulator from the commanded joint configuration $\mathbf{q}_{\text{cmd}}$. In the proposed implementation, this deflection is measured by a changed position $\mathbf{q}_{\text{curr}}$ in configuration space and fed to an additional damping term to prevent the robot from continued drifting after being moved. This interaction controller behaves similar to the originally provided gravity compensation controller provided by the LWR IV [25], but has the advantages (from a system integration point of view) of avoiding mode switches of the low-level control on the LWR IV. These typically require several seconds of a constant joint configuration and zero external torques to be permitted which is not desired for a seamless handling experience of the user. Furthermore, this allowed us to control stiffness, damping and maximal velocity to our needs regarding safety requirements. The resulting *gravity compensation* controller π_x allows users to freely move all joints of the robot according to the demonstrating-in-configuration-space strategy in order to guide the robot during the *APPROACHING* phase to relevant areas of the workspace with a desired initial joint configuration $\mathbf{q}_{\text{fixed}}$.

The utilized controller for *RECORDING* is the low-level impedance controller of the LWR IV. Hence, the required *compliant recording* π_{rec} is "implemented" by simply exploiting the active compliance of the LWR IV controlled in the joint impedance mode as shown in Fig. 4.2. This allows the user to record demonstration data $\mathcal{D}^{(k)} = (\mathbf{x}_l^{(k)}, \mathbf{q}_l^{(k)})_{l=1,\dots,L^{(k)}}$ of the desired redundancy resolution in a relevant training area by physically deflecting the robot from the commanded configuration $\mathbf{q}_{\text{cmd}} \equiv \mathbf{q}_{\text{fixed}}^{(k)}$. Since the impedance control acts like a spring-damper system [25], the recorded joint configurations vary only locally around the selected initial posture $\mathbf{q}_{\text{fixed}}^{(k)}$. Examples of such training data are shown in Fig. 4.3 in $K = 6$ training areas.

Since the proposed implementation of the *assisted gravity compensation* controller utilizes the intrinsic active compliance of the LWR IV, the ease of moving is determined by the stiffness and damping values $\mathbf{k}$ and $\mathbf{d}$ of the underlying joint impedance controller. For the prototype FlexIRob and the following evaluations these were carefully chosen as detailed in Chap. A to allow smooth interaction.

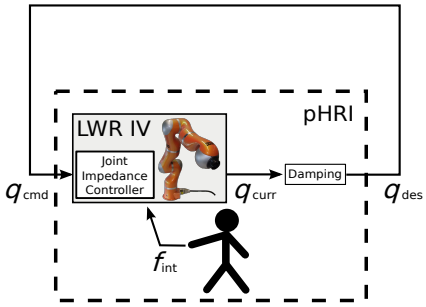


Fig. 4.1: Implementation of the required *gravity compensation* controller π_q according to Sect. 2.3.1. It utilizes the low-level joint impedance controller of the LWR IV and an additional damping term.

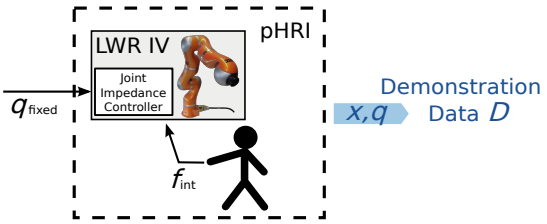


Fig. 4.2: Illustration of the proposed *compliant recording* controller π_{rec}. It refers to the LWR IV's joint impedance controller and allows physical deflection only locally for recording consistent training data as proposed in Sect. 2.3.1.

4.2 Teaching Implicit Constraints

The approach presented in this thesis is designed to enable intuitive interactive configuration of redundant robots by lay users. But instead of programming *explicit constraints*, I argue that this goal can be rephrased as the problem to transfer the user's *implicit knowledge* and understanding of an environmental scene through interaction into the robotic system. When teaching, the user knows where the workspace is confined, he or she can not move the robot into the static obstacles physically and has to care that neither part of the robot arm collides anywhere. These constraints are intuitively clear to the users who choose their training data accordingly. This section deals with the question what kind of knowledge actually is transferred from the human tutor to the robot system during the *CONFIGURATION*. While this question is rather difficult to answer in general, I will shed light on two aspects of it in the following. Sect. 4.2.1 gives an exemplary view on the implicit constraints that are imposed by the user demonstrations in the manipulator's joint space. However, since a general analysis of these constraints in joint-space is neither intuitive for interpretation nor simply to visualize, I displace this analysis to the Cartesian space in Sect. 4.2.2. A visualization of the implicitly modeled scene gives rise to constraints modeled in the robot's task space.

For both sections the data of the reachable workspace analysis from Sect. 3.3.2 is recycled and analyzed now from a different viewpoint. That is, this analysis utilizes again the obstacle setups I and II and is based on the set of target points T_{I} and T_{II} which per definition constitute the reachable workspace of the respective setup. After the human tutor has conducted the kinesthetic teaching process according to the *CONFIGURATION* stage, the robot is commanded to reach for each of the targets in T_{I} or T_{II} with the learned neural network included in the

inverse kinematics controller of the robot. Again, the ELM approach is utilized here as described in Sect. 3.2.1. As an illustration, Fig. 4.3 shows setup I for which the user recorded training data in six different areas of the workspace.

4.2.1 Constraints Modeling in Joint Space

During the kinesthetic teaching process as described in Sect. 2.3.1, the human tutor provides different redundancy resolutions of the inverse kinematics in different areas of the robot's workspace. By this means, implicit constraints are imposed on the robot's joint-space by means of the training data. The goal of this section is to give an exemplary insight in these constraints.

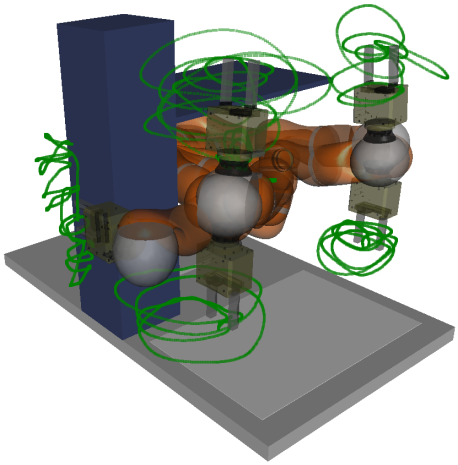


Fig. 4.3: Exemplary training data recorded in six areas of setup I.

Given the training data shown in Fig. 4.3, the trained robot is commanded to reach for each of the 635 targets in T_{I} and the resulting joint values $\mathbf{q}$ as well as task space positions $\mathbf{x}$ are recored. By visualizing the resulting distributions of joint values of the manipulator's joints, implicit constraints that are induced by the training data can be revealed. For instance, the very left scatter plot in Fig. 4.4 shows the distribution of the joint values of q_2 both during the evaluation (black) and during the training phase (grey). It clearly reveals, that the user by means of the training data imposed a strong constraint on that joint, namely that the values should not exceed a small neighborhood around $q_2 = \frac{\pi}{2}$. Taking a look at the obstacle setup I (cf. Fig. 4.3), this constraint is quite obvious. A value of $q_2 = 0$ refers to an upright position while $q_2 = \frac{\pi}{2}$ results in a position parallel to the table surface. Hence, values of $q_2 >> \frac{\pi}{2}$ would result in collisions with the table surface while movements with $q_2 << \frac{\pi}{2}$ would cause collisions with the upper part of the obstacle. The choice of $q_2 = \frac{\pi}{2}$ therefore minimizes the risk of collisions or, in other words, maximizes the distance between the robot's links and the environmental obstacles. The second plot in Fig. 4.4 visualizes another constraint related to that ceiling part of the obstacle. The values of q_5 are plotted against the height of the resulting end-effector positions. In order to reach at or higher than $z \approx 0.4\,\mathrm{m}$ without collisions, the robot needs to change its gripping pose from "grip from above" to "grip from underneath". Here the user has chosen joint q_5 to comply to this environmental constraint on the end-effectors's orientation, so that low positions $z < 0.4$ are realized with $q_5 \geq 0$ and high positions $z > 0.4$ with $q_5 \leq 0$. This behavior is reflected in the example data provided by the user in the

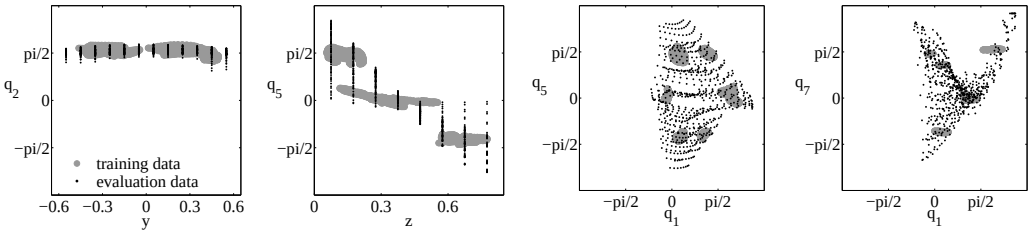


Fig. 4.4: Visualization of the implicitly taught joint space constraints for setup I by means of selected scatter plots. The system was trained with six training areas as visualized in Fig. 4.3 and then evaluated on the set of target points T_{I}.

training areas and is generalized by the system well to the rest of the obstacle-dependent workspace T_{I}. While these two plots are examples for rather tangible constraints imposed by the obstacles in the manipulator's workspace, the third and fourth scatter plot in Fig. 4.4 show more implicitly taught relations between different joints, that are more difficult to analyze and understand. These might even not be crucial for the collision avoidance, as for example since they relate to the very last joint q_7 which only rotates the the gripper around the gripping axis, but nevertheless have been selected by the user for the redundancy resolution and can be generalized well beyond the training areas.

A similar exemplary visualization for the system being trained in obstacle setup II is provided in Fig. 4.5. Again, a user recorded training data in six different training areas, and the system, i.e. the learned redundancy resolution, is evaluated on the set of target points T_{II}. The first scatter plot visualizes a very obvious environmental constraint on joint q_1. Since two pillars are placed besides the robot constraining the workspace for the link mounted on this joint, only a very narrow gap around $q_1 \approx 0$ is left for the robot to move collision-free. The second plot demonstrates how the user taught the robot to reach for targets left and right (y-axis) in the workspace while avoiding the two obstacles. The training data indicates that for end-effector positions with $y < 0$ joint values $q_3 > 0$ should be used and vice versa. The third and the fourth plot again visualize constraints purely in the joint space. The training data provided by the user reveal a clear, almost functional relation from joint q_3 on q_2 which can be approximated by $f(x) = |x|$ on $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and which is again well generalized by the trained system to the rest of the workspace T_{II}. Concerning the joints q_4 and q_6, the data exhibit no relation between those but rather strong constraints in its distribution. The values of both joints are constricted to a corridor around $q_4 \approx -\frac{\pi}{2}$ and $q_6 \approx \frac{\pi}{2}$.

The presented examples illustrate the joint space constraints that are modeled by the human tutor during the kinesthetic teaching process. By means of the training data the user is enabled to introduce obstacle dependent restrictions on and relations between the robot's joints to resolve the redundancy. It is worth noting, that the joint values of $q = \pm\frac{\pi}{2}$ and $q = 0$ seem to play an important role in the

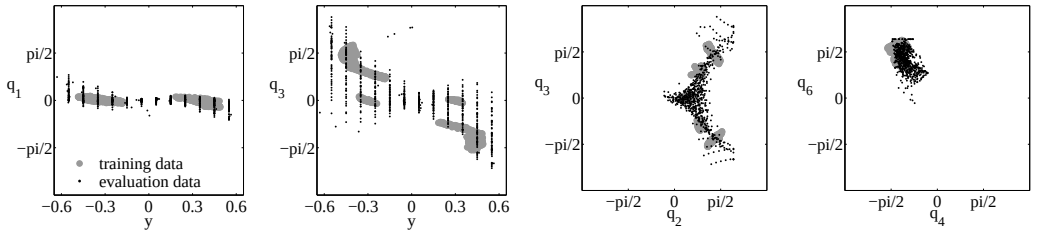


Fig. 4.5: Visualization of the implicitly taught joint space constraints for setup II by means of selected scatter plots. The system was trained with six training areas and then evaluated on the set of target points T_{II}.

description of the constraints exemplified above, particularly when regarding the provided training data. This is due to the geometry of the investigated obstacles, which are placed axis aligned to the robot's coordinate frame and consist of rectangular elements, as well as the kinematics of the LWR IV. Using these joint values seems to be a "natural" or straight-forward way for a safe, collision-free redundancy resolution. A robot expert configuring and programming the robot system according to explicit reward or cost functions like minimum distance between robot and obstacles probably would come up with a similar solution. However, the devised interaction and learning scheme proposed in this thesis enables non-expert users to program those constraints implicitly and in an intuitive way without having such a reward function explicitly in mind.

4.2.2 Implicit Scene Model in Task Space

While the former section focused on a visualization of the constraints in the joint space, this section visualizes the taught constraints in task space. The idea behind this is, that with the training data an implicit model of the task space obstacles is conveyed from the tutor to the robot system. When teaching, the user knows where the workspace is confined, he or she can not move the robot into the static obstacles physically and has to care that neither part of the robot arm collides anywhere. Instead of considering where the robot arm physically is when driven by a particular redundancy resolution, the following analysis rather asks where the robot is not. It checks which parts of the workspace are never used by the robot's links, joints or any other parts of its body when generalizing the learned solutions and moving to the relevant parts of the task space. This is in contrast to Sect. 3.3.2 where I answered the question to what extend the workspace was made accessible without colliding with the environment. This section rather provides a "dual" view on the data gathered by that evaluation.

The evaluation is conducted similar to the reachable workspace analysis from Sect. 3.3.2. That is, it utilizes again the obstacle setups I and II and is based on the set of target points T_{II} and T_{II} which per definition constitute the reachable workspace of the respective setup. Again, the human tutor conducts the kinesthetic

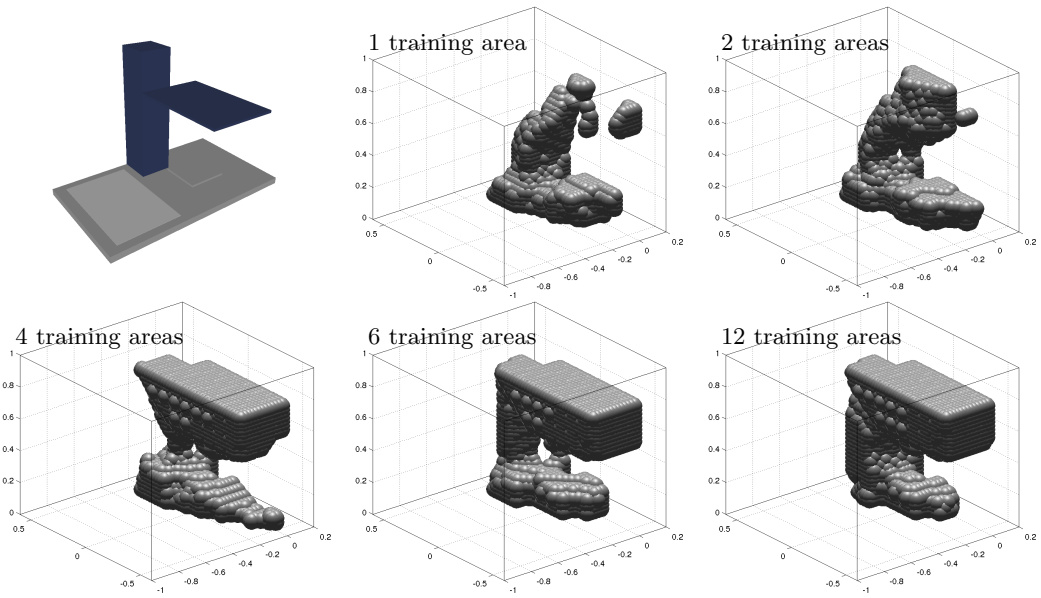


Fig. 4.6: Obstacle setup I and implicitly modeled scenes for varying number of training areas.

teaching process according to the *CONFIGURATION* stage. For each obstacle setup he repeats this procedure with a different number of training areas. As an illustration Fig. 4.3 shows setup I for which the user recorded training data in six different areas of the workspace. The neural network utilized to generlize the taught constraints from that data is the ELM approach as described in Sect. 3.2.1. During evaluation the robot is commanded to reach for each of the targets in T_{I} or T_{II} with the learned ELM included in the inverse kinematics control of the robot (cf. Sect. 4.1). For each target the resulting posture, i.e. the positions of all links and joints of the LWR IV, is stored. By this means, the targets together with the corresponding postures define a subspace of the Cartesian space which is termed the *maneuvering space* $\mathcal{M} \subseteq \mathbb{R}^3$, enclosing all points of the space where the robot was. By "inverting" this viewpoint, we visualize that parts of the workspace which are never used by the arm. From the set T of the defined obstacle-independent workspace of the LWR IV we subtract all points that are in or close to the maneuvering space:

$$S := \{\mathbf{x} \in T \ : \ d(\mathbf{x}, \mathcal{M}) > 0.1\}. \tag{4.1}$$

These parts comprise a kind of *implicit scene model*, which is encoded in the particular learned redundancy resolution. In particular, if the system was trained appropriately by the human tutor, these parts should enclose the static obstacles of the environmental scene.

The results for setup I are shown in Fig. 4.6 which visualizes the actual obstacle and the implicitly modeled scenes S obtained for different numbers of taught

training areas. As for the trial with only one training area, already a large part of the workspace does not lie in the maneuvering space of the robot. The visualized model, that already covers parts of the actual obstacle (the lower part of the box part of the obstacle) has been avoided by the robot system during the evaluation on the targets T_{I}. By providing training data in only one area of the workspace, the user conveyed the information to avoid that visualized parts of the workspace. However, since other parts of the actual obstacle still lie in the maneuvering space of the trained system, this result indicates that the provided training data does not comprise enough knowledge about the environmental scene to cover all obstacles. Given more training data, the implicit model S covers more and more of the environmental scene and models the obstacles more accurately. For instance, given four training areas which the user found to be relevant the board part of the obstacle is already entirely covered by the implicit scene. Only the middle of the box part of the obstacle is not covered. Hence, the training data resulting in this trained system comprised a better implicit model of the environmental scene. Finally, the results for the trial with 12 training areas show that the implicit scene model almost entirely envelopes the physical obstacle.

As for setup II similar results are obtained, see Fig. 4.7. Again, the first picture shows the obstacle in the simulated environment, whereas the second shows the implicit scene S after the tutor provided data in only one training area. With the given training data the user conveyed information about which parts of the workspace should be avoided. Since in this trial only training data with "elbow up" solutions was provided, only the lower parts of both box obstacles are covered by the implicit scene indicating that the upper parts are being touched when driving the robot with the learned redundancy resolution. In the next trial, the user taught the system in six different training areas which transferred enough information to already model one of the box obstacles precisely as the resulting implicit scene S reveals. However, as argued already in Sect. 3.3.2 these obstacles constitutes a challenging scenario requiring a precise inverse kinematics solution to reach for all of the targets T_{II}. Hence, more training data is needed for a precise model of the obstacles. Fig. 4.7 (right) shows the implicit scene after the teacher has recorded data in 21 areas of the workspace he found to be relevant, revealing an accurate model of both boxes. Whereas the left box is entirely covered by the implicit scene, only a small area on the right box still lies in the maneuvering space of the trained system.

The presented analysis shows what kind of implicit knowledge about the environmental scene in task space is transfered to the robot by means of the training data. By visualizing those areas of the workspace that are not used by the manipulator, a model of those parts in the scene to be avoided is revealed, which is implicitly conveyed with the training data. Providing only few or meaningless training data results in an inaccurate implicit scene model, whereas training in interesting areas shapes the implicit model such that more parts of the obstacle are covered. Two things are worth noting here. First, the presented interactive

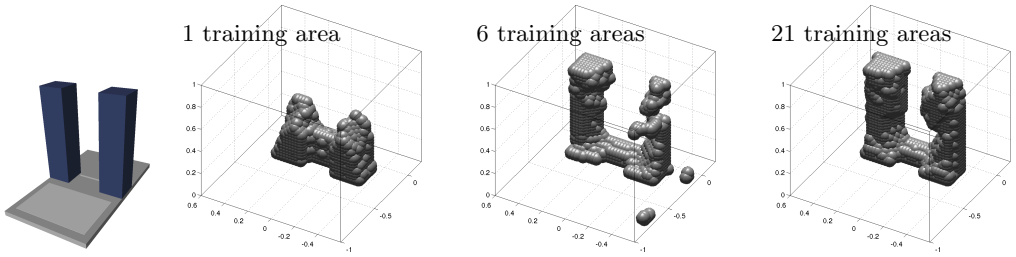


Fig. 4.7: Obstacle in simulated setup II and implicitly modeled scene for one, six and 21 trained areas (from left to right).

teaching procedure is designed for fast and intuitive (re-)configuration of a robot's redundancy resolution according to environmental constraints rather than precisely learning and modeling physical obstacles. The obstacles of setup II define a challenging scenario where the robot arm has only a very narrow gap of ca. 5 cm on each side for navigating the "elbow joint" q_3. However, the results of this analysis show that the approach effectively is capable of also carrying out that. By means of appropriate training data the teacher is able to provide a very accurate model of the environmental scene in both setups entirely enclosing the static objects in the workspace. Second, although the number of 21 training areas sounds a lot on first sight, the entire teaching procedure still requires less than ten minutes by a skilled person, with the term "skilled" refering to being properly instructed rather than a robot expert or an engineer. The kinesthetic teaching process as described in this thesis enables users to adapt the redundancy resolution without any explicitly modeling or writing any line of code.

4.3 FlexIRob@Harting: Teaching Redundancy Resolutions

While in the last sections, the presented results were obtained with an expert providing the training data, in the following I present results from the user study FlexIRob@Harting, a larger study about physical human-robot interaction with redundant manipulators in the context of industrial scenarios [22]. Hence, parts of the results presented in this section have been published in [22]. Throughout this study, 49 factory floor workers from the German company Harting [24], most of whom never had worked with robots before, were asked to teach the robot system redundancy resolutions in two training areas of a confined workspace similar to setup I from the previous sections as shown in Fig. 4.8. The study design and the experiment course for each participant is summarized in Chap. B. After the first subsection reports on the individual user experience during the interaction with the robot system, subsequently the success of that teaching procedure is evaluated and the users' provided training data is analyzed with respect to that success.

Fig. 4.8: Participant of the FlexIRob@Harting study during the *CONFIGURATION* phase teaching the LWR IV redundancy redundancy resolutions in the second training area of the confined workspace.

4.3.1 General User Experience

The interaction experience of each individual user during the user study was retrieved by means of a questionnaire he or she filled after finishing the entire experiment (cf. Chap. B). The questionnaire asked for the users' experience concerning the physical handling of the robot manipulator, the cognitive load they encountered, properties they attributed to the system such as reliability and intelligence, as well as the pleasantness of the interaction scheme.The results for this general experience with the robot during the experiment are shown in Fig. 4.9. A rating of 1 indicates yes/very much and 5 indicates no/not at all.

A very important result is that participants rated the robot as non-threatening, although, depending on which department of Harting they work in, they get safety instructions and are warned against the dangers of industrial robots. One reason for that might be that they rated the system as reliable and perceived the robot's intelligence to be high. Concerning the handling of the robot, the users rated it to be very easy and also self-explanatory. Here, the proposed interaction and teaching prototype takes advantage of the direct physical interaction between robot and human. Touching and manually guiding a robot is much less difficult than operating it by means of remote devices. These results also indicate that the structure of the teaching procedure was chosen appropriately, which otherwise would contradict with the rating of a self-explanatory system. Another very distinct and important

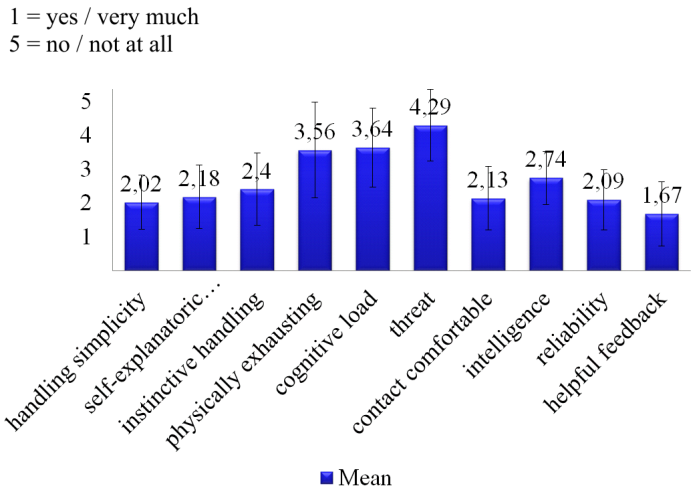


Fig. 4.9: Average ratings of the general user experience with the FlexIRob system.

rating is that the participants felt the acoustic and visual feedback given by our system was helpful to complete the task. This indicates that not only the structure of the teaching procedure was chosen appropriate for an intuitive interaction but that also the feedback about the current state of the system (*APPROACHING* or *RECORDING*) was appreciated by the users. Interestingly, concerning the cognitive load and the physical effort the users had to bring up during the interaction the results are not quite clear. Both ratings exhibit a mean value of ≈ 3.5, which means "more or less", and a high standard deviation across all users. As for the physical effort, the reason might be stiffness and damping values of the impedance control mode (cf. Chap. A for the specific values) that might be inappropriate for some (maybe less strong) users such that manipulating the joints was physically more exhausting for them as for others. As for the cognitive load, I hypothesize that even though the users had a short warm-up phase to familiarize with the handling of the robot before the experiment, that phase might have been not decent enough for them to get used to the robot's kinematic abilities (such as redundancy) and limitations (such joint limits). Therefore, finding appropriate training postures in both training areas might be difficult for some users. In order to investigate these effects, in the following different user behaviors or teaching experiences are illustrated exemplarily.

4.3.2 Illustration of Training Data for Different User Experiences

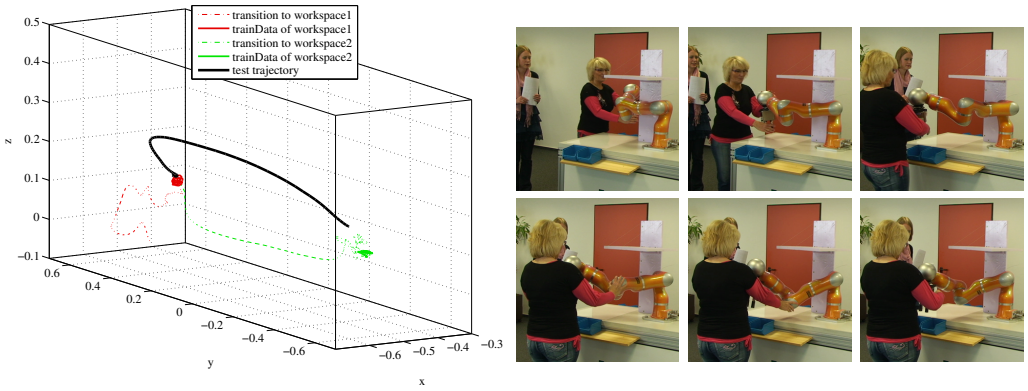
Throughout the user study the participants revealed a high variety of how they attempted to solve the task of teaching the redundancy resolutions in both training areas. In the following I illustrate this variety mainly by means of the recorded training data, the chosen postures of the robot arm in the two training areas and

the interaction forces induced by the users during the teaching process. Indicated by the high standard deviations in Fig. 4.9, the hypothesis is that these differences mostly are based on the cognitive load imposed on the participants during the task and the physical effort during the interaction.

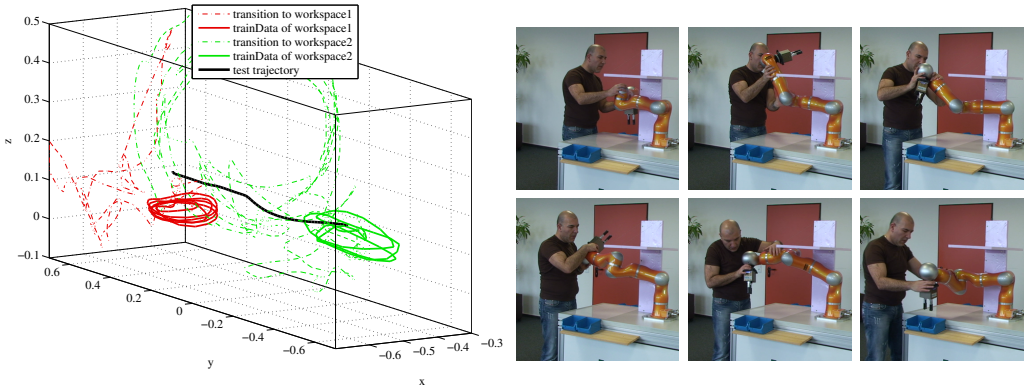
Regarding physical effort, these differences are well reflected in the recorded system data. As described in Sect. 4.1, during the *RECORDING* phase the LWR IV was controlled in *compliant recording*. For recording training data in a certain area of the workspace, the participants were encouraged to move the robot's end-effector in circular trajectories. Since the robot is commanded to stay in the user-defined posture, the participants had to exert force on it in order to perform these movements. These external forces, measured as estimated external torque in each joint, were recorded during the study and deliver an insight into the different user experiences. Whereas some participants did not hesitate to strongly push the robot from its commanded position and therefore induced torques up to 18 Nm in some joints, other participants only slightly touched the end-effector and induced only about 2 Nm as maximum joint torque. These different behaviors are also revealed by the task-space coordinates of the recorded training data. In Fig. 4.10 (left), the recorded system data of three participants during the *CONFIGURATION* phase are shown. As illustrated in these figures, the cubic expansion of the recorded training data (solid green and red lines) varies from data with almost no variance in Fig. 4.10(a) to data consisting of several circular movements with a diameter of up to 32 cm as in Fig. 4.10(b).

Regarding the cognitive load, the hypothesis is that the inter-subject differences mainly relate to the problem of handling the redundant robot arm. Whereas some participants had no problem manipulating the KUKA LWR IV's joints in order to guide it to a certain end-effector position without colliding with obstacles in the environment, others seemed to think a lot and play around with the robot until they found a solution to the task. An example to that gives Fig. 4.10 (right), where video snapshots show the participants guiding the robot arm from the first training area to the second. In Fig. 4.10(b), the participant tries several ways of guiding the robot from the first area to the second, thereby rotating the end-effector in several ways while experiencing the joint limits of the robot. In contrast, Fig. 4.10(c) shows a participant which seems to have no problem with the redundancy of the robot and quickly moves it to the second training area. These differences are also revealed by the participants' system data visualized in Fig. 4.10 (left), where the red and green dashed lines show how the participants moved the robot's end-effector while trying to find a good training posture in the corresponding training area. In order to quantify these differences between the participants, we measured the time needed to guide the robot to a certain training area. For the first training area, these times varied from a minimum of ca. 8 s up to a maximum time of 104 s; for the second training area, the minimum and maximum amount of time were 6 s and 45 s, respectively.

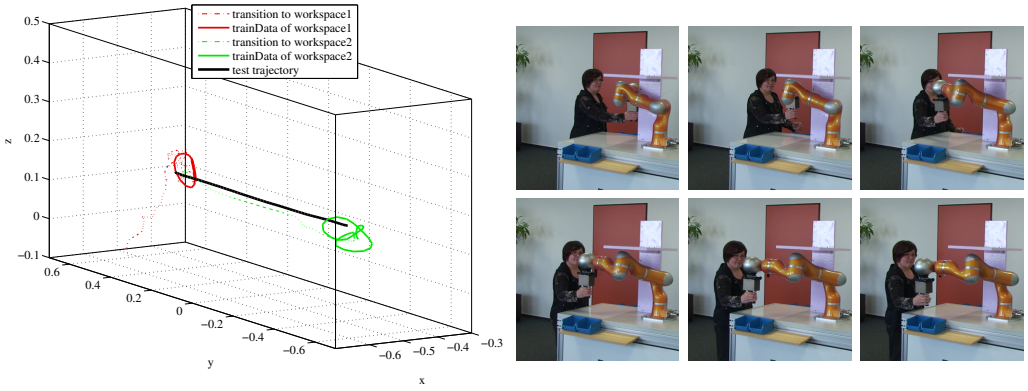
In particular, these results show that a too small variance in the training data



(a) Test person recording training data with only a low variance and guiding the robot arm through a kinematic singularity. As a result, the accuracy of the performed test trajectory (should be a straight line in-between the two areas of the workspace) is poor.



(b) Test person recording training data with a high variance but having problems guiding the robot arm from one area of the workspace to the other. Nevertheless, the performed test trajectory is more or less a straight line in between the two areas of the workspace.



(c) Test person providing good training data with an appropriate variance and a valid combination of postures in both areas of the workspace. As a result, the robot is able to perform the test trajectory between the training areas with a high degree of accuracy.

Fig. 4.10: Examples of different user behaviors during *CONFIGURATION*.

and badly chosen postures can lead to poor generalization performance in terms of bad task space accuracy or encountered collisions. In contrast, these illustrations also signal that some users intuitively provide suitable data after some instructions and the quick familiarization phase such that an accurate and collision-free test movement is performed by the trained robot system. In order to present a more detailed numerical analysis of this user experiment, the development of the user's training success over several trials is systematically investigated in the following.

4.3.3 Analysis of the Participants Teaching Success

As a result of the participants' varying behaviors and experiences as described above, the training success also varied strongly, which will be analyzed in the following. As outlined in Chap. B, each participant was asked to conduct the *CONFIGURATION* in three trials. In order to evaluate the teaching success after each trial, the trained system was commanded to move along a straight line in the work-space as a reference movement. The following metrics are then utilized to analyze the success of that trial.

Defining Metrics for Evaluation

According to [21], where the authors proposed coherent means for evaluating the success of human-robot interaction methods, I utilize measurements to access the *system effectiveness* and the *system efficiency* of the proposed approach. The *effectiveness* measures how well a predefined task is accomplished. The goal of the *CONFIGURATION* phase is to teach the robot arm certain inverse kinematic solutions in different areas of its workspace in order to enable it to move in a predefined part of the workspace while avoiding obstacles in the environment. The predefined part thereby was introduced to the users as the area consisting of both training areas and the space in between them. Thus, I identify two measures of task effectiveness: (a) accuracy of the performed test trajectory, which ideally consists of a straight trajectory between the training areas, and (b) the number of unintended contacts or collisions of the robot arm with its environment. Note, that these metrics directly relate to the measurements E_{task} and E_{coll} from Sect. 3.3 that have been utilized to evaluate the generalization abilities of the learning approaches and the entire system. *Efficiency* measures the time that is needed to complete a task. Here, I distinguish between the time that is spent on the kinesthetic teaching and the time needed for calculation of the estimated inverse kinematic model, i.e. time that is solely consumed by the learning algorithm.

General Teaching Success

Based on how well the trained system executed this reference trajectory, the results obtained after the third configuration trial with regard to effectiveness can be categorized as shown in Fig. 4.11:

- 3 participants were unable to teach the system to produce no collisions and additionally only managed to achieve low accuracy of the test trajectory.
- 18 participants achieved high accuracy of the test trajectory but with collisions.
- 17 participants managed to train the system to produce no collisions but with low accuracy of the test trajectory. For an example, see Fig. 4.10(a).
- 9 participants were able to teach the system such that high accuracy was achieved and no collisions occurred (see e.g. Fig. 4.10(c)).

In this analysis, the threshold of an appropriate task-space accuracy was set to 1 cm.

These results show that non-expert users are indeed able to train the robot according to environmental constraints as 26 participants successfully taught the robot to avoid the environmental obstacles. However, the results also raise the impression that most of the participants were able to do the configuration either accurately or collision-free. Concerning a statistical analysis, this effect is well reflected in the data by a correlation of $r = -0.58$, $p < 0.01$ between the task space accuracy of the performed test trajectory and the number of collisions that occurred. The reasons for that will be analyzed in Sect. 4.3.4. As a result, only nine participants trained the system such that both accuracy and collision avoidance of the trained system was acceptable.

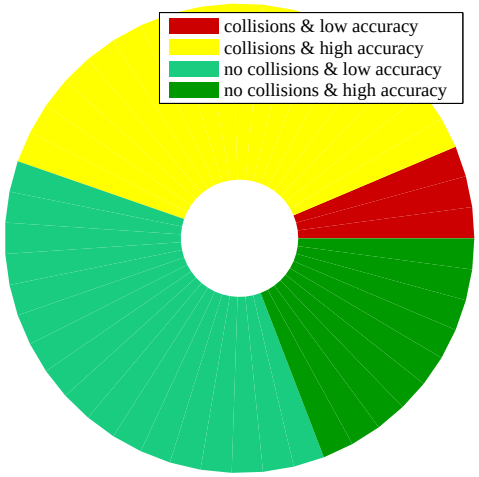
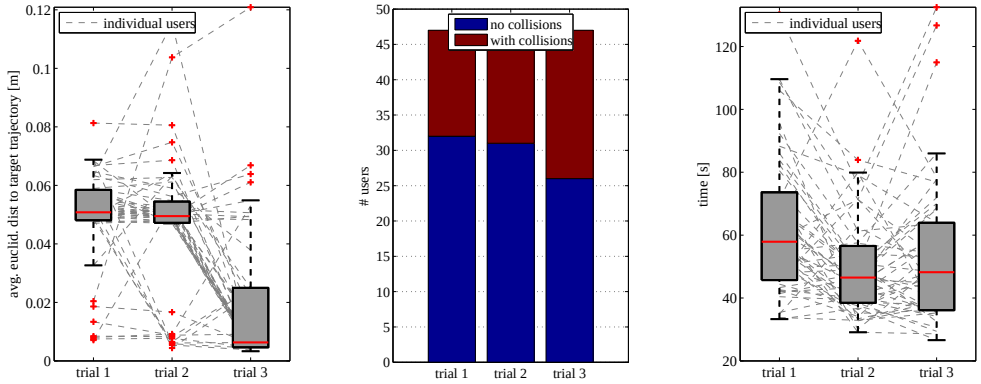


Fig. 4.11: General success of the configuration after the third trial.

Concerning the efficiency of the human-robot interaction, the time needed for a single configuration procedure splits into time that is spend on the interaction phases *APPROACHING* and *RECORDING* on the one hand, and the computation time required by the learning algorithm on the other hand. We report an average time of 54.1 ± 24.8 s for the former and 5.3 ± 1.0 s for the latter, which results in an average time requirement for a single entire configuration procedure of approximately one minute.

Development over Trials

In order to investigate the participants' teaching success over several trials and to find out how much instructions they need to successfully perform the teaching (in



(a) Development of participants' task-space errors.

(b) Number of participants with collisions during testing.

(c) Time needed by the participants for the configuration.

Fig. 4.12: Development of the participants' performance over three trials of the configuration task.

the sense of providing reasonable training data in both regions of the workspace), the *CONFIGURATION* was repeated in three trials with increasingly informative instructions, which were the same for all participants. The instructions were chosen in a way that a) with ongoing trials the possibilities for moving the robot from the first working area to the other were more and more restricted, and thus should help the participants to find proper redundancy resolutions, and b) the participants' thoughtfulness concerning the trained robot's ability to move in a straight line from the left to the right working area should be raised (cf. Chap. B).

Figure 4.12(a) shows the development of the trained system's task space error E_{task} while performing the reference trajectory of each individual user over the three trials. The corresponding box-plots indicate the distribution of errors at each trial. Interestingly, most of the users neither improved nor got worse from the first to the second trial, yielding a median Euclidean error of approximately 5 cm. However, in the third trial many users improved their performance significantly, such that the median user performance is below 1 cm. Concerning the participants' ability to train the system to avoid the environmental obstacles, we could not measure any significant difference between the first and the second trial (see Fig. 4.12(b)). But differences between the second and third trial indicate that the improvement in terms of task space accuracy comes at the cost of a slight gain in the number of participants with collisions. Again, this raises the hypothesis of a negative correlation between the ability to train the system accurately and to avoid collisions. As for the users' time needed for kinesthetic teaching, only a slight improvement was measurable from the first to the second trial (see Fig. 4.12(c)).

4.3.4 Quality of the Provided Training Data

The results in the previous sections showed that it was not possible for all participants of the study to teach the robot such that a reference trajectory could be performed accurately *and* without collisions. While the particular reasons remain unclear so far, they rather reveal a tendency for most of the trained systems to perform *either* with high accuracy *or* without collisions. With the following analysis I address this issue through a decent analysis of users' the training data that was provided by the participants during the *CONFIGURATION*.

Inconsistent Combinations of Redundancy Resolutions

The first hypothesis - and a reason for a possible failure of teaching the robot system appropriate redundancy resolutions - is, that users might chose redundancy resolutions or postures for the two training areas that are not compatible with each other with respect to the environmental or task constraints. Each chosen posture might be suitable for the respective training area and respect the environmental constraints there, but performing a movement between the areas would only be possible by not respecting the task constraints (straight line between areas) or the environmental constraints (avoid collisions with obstacles). An example for this situation is given in Fig. 4.13 where "elbow-right" and "elbow-left"-solutions have been selected by the participant for the left and right training area, respectively. A similar example was already shown in Fig. 4.10(a). However, while both solutions allow collision-free movements in or in the vicinity of the respective training area, no solution exists to the problem of moving the robot arm from the first to the second posture with the end-effector performing a straight line between them. In that sense, both redundancy solutions are incompatible with each other and as a result also the trained system violates the task constraint or the environmental constraints while performing the reference trajectory.

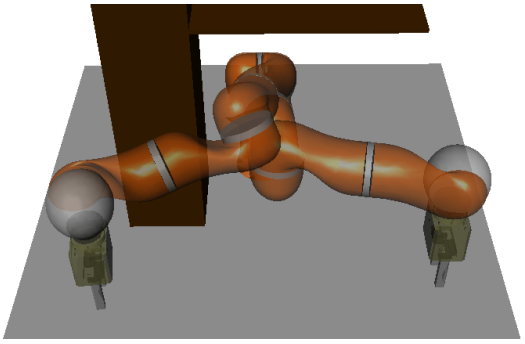
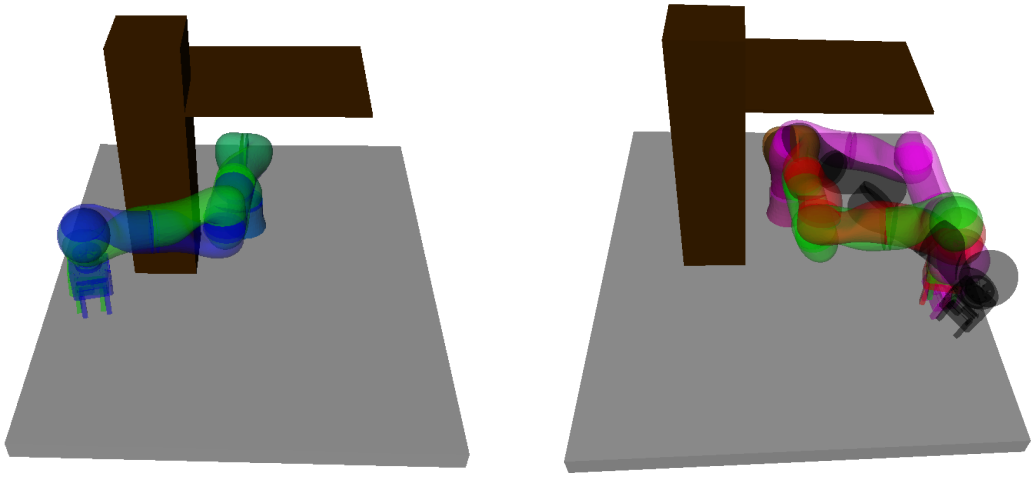


Fig. 4.13: Example for inappropriate combination of redundancy resolutions in left and right training areas.

In order to support the hypothesis, that the participants' teaching success relates to the combination of selected redundancy resolutions, the users' training data from the third *CONFIGURATION* trial have been clustered into different classes of redundancy resolutions. For each of the two training areas a separate cluster analysis is done based on the respective robot postures provided by the users in terms of the initial joint angles $\mathbf{q}^{\text{fixed}}$ recorded during the *RECORDING* stage. The method utilized for this analysis is the k-means algorithm [117] with



(a) Visualization of cluster centers L_2 (green) and L_3 (blue) in left training area.

(b) Visualization of cluster centers R_1 (red), R_2 (green), R_6 (magenta) and R_7 (black) in right training area.

Fig. 4.14: Exemplary results of the clustering of the participants' training postures $\mathbf{q}^{\text{fixed}}$ done separately in left and right training area. For the sake of clarity only selected cluster centers are displayed in the simulated environment.

a predefined k and initialized with cluster center candidates in each training area, respectively. Both the number k of clusters and the initial cluster centers were chosen after a visual inspection of the data and by incorporating expert knowledge about the kinematics of the KUKA Lightweight Robot IV. For detailed informations about the clustering procedure confer to Sect. B.4. The results of this analysis reveal that the training postures chosen by the participants in the left training area clearly can be clustered into three different classes which will be termed L_1, L_2 and L_3 in the following. Hence, the redundancy resolution selected by each of the participants in the left training area can be assigned to one of these clusters. As a result, 23 users are assigned to L_1, two participants refer to L_2 and L_3 consists of the remaining 22 users. The different clusters can be characterized as all realizing approximately the same end-effector position since, during the study, the designated training area was clear to the users (indicated by the space above some blue boxes, cf. Fig. 4.8). They may even look very similar to each other on first sight but constitute completely different redundancy resolutions as visualized in Fig. 4.14(a) which shows the cluster centers L_2 (green robot) and L_3 (blue robot) in the simulated environment. Although the visual impression is very similar, the displayed postures differ strongly in q_3, q_4, q_5 and q_6. The training data of the right training area is distributed over more clusters. In total, seven clusters $R_1, \ldots, R_7$ have been identified with cluster members ranging from only one user up to 16 participants. Fig. 4.14(b) visualizes the cluster centers of selected classes. Again,

the color-coding of the shown robots relates to the particular clusters R_1 (red), R_2 (green), R_6 (magenta) and R_7 (black), respectively. For more detailed informations about the identified clusters as well as for a visual inspection of the clusters' data distributions in 2-D using multi-dimensional scaling [118] please confer to Sect. B.4.

Having the users' training data clustered into dedicated classes, their teaching success can be analyzed with respect to what combination L_i-R_j of redundancy resolutions in left and right training area they had chosen. Fig. 4.15 shows the participants' teaching success in terms of the task space accuracy E_{task} of the reference trajectory depending on the chosen combinations of redundancy resolutions. For instance, users whose training data can be assigned to the combination L_3-R_1 all missed to teach the system a high accuracy along the reference trajectory. From the corresponding boxplot a median task space error of approximately $E_{\text{task}} \approx 0.055\,\text{m}$ can be deduced. In that sense, this combination of redundancy resolutions seems to be inappropriate. With the same argumentation, the redundancy resolutions of the combinations L_1-R_3, L_3-R_7, L_2-R_2 and L_3-R_3 seems to be incompatible with each other, since none of these participants achieved high accuracy of the trained system. An example for the last combination L_3-R_3 was already shown in Fig. 4.13. In contrast, all the users who selected their training data according to L_1-R_1 achieved a high accuracy of below 1 cm of the trained system. As for the combinations L_1-R_6 and L_3-R_4 some users achieved good results but some also missed to enable the robot accurately following the reference trajectory. The same effect is observable concerning the relation between combination of redundancy resolutions and the number of collisions E_{coll} that occurred during the reference trajectory after teaching. Fig. 4.16 shows the respective results. Here, teaching success refers to a value of zero collisions. Again, some combinations, namely L_1-R_7, L_3-R_7, L_1-R_1 and L_3-R_3, can be recognized as inappropriate since none of the respective users achieved a teaching success of zero collisions. As for the other combinations of redundancy resolutions in left and right training area at least some users managed to teach the system successfully in terms of collision avoidance. Hence, these combinations in principle are compatible with each other.

	# users	consistent
L_1-R_1	7	-
L_1-R_3	3	-
L_1-R_6	12	✓
L_1-R_7	1	-
L_2-R_2	1	-
L_2-R_5	1	✓
L_3-R_1	4	-
L_3-R_3	1	-
L_3-R_4	16	✓
L_3-R_7	1	-

Tab. 4.1: Identified (in-)consistent combinations of trained redundancy resolutions in left and right training area.

Combining the results for both measurements, this analysis reveals that the teaching success of the participants clearly depends on which redundancy resolutions they combine in the left and right training area. Only the

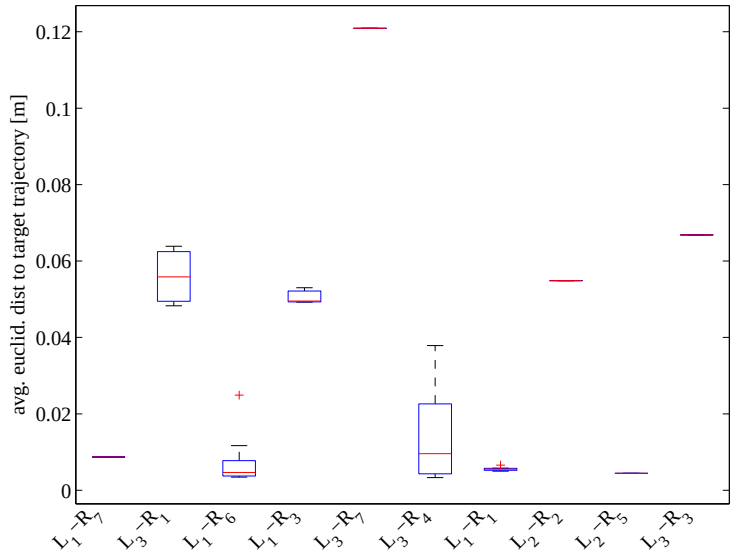


Fig. 4.15: Teaching success in terms of the average error depending on which cluster the participants' training data can be assigned to in each workspace. E.g. L_1-R_6 displays a boxplot of E_{task} for those users whose data can be assigned to cluster L_1 in the left training area and to R_6 in the right training area.

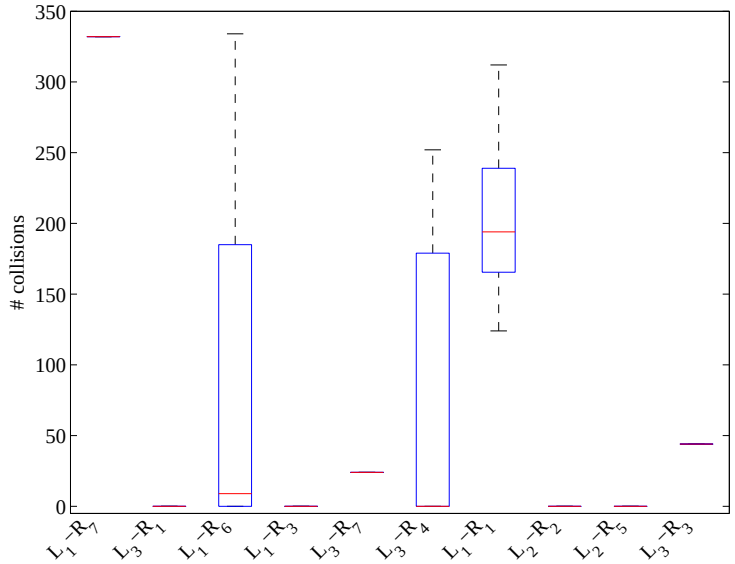


Fig. 4.16: Teaching success in terms of number of collisions depending on which cluster the participants' training data can be assigned to in each workspace. E.g. L_1-R_6 displays a boxplot of E_{coll} for those users whose data can be assigned to cluster L_1 in the left training area and to R_6 in the right training area.

combinations L_1-R_6, L_3-R_4 and L_2-R_5 allowed users a successful teaching concerning *both* measurements, and the 9 participants that managed to do so obviously belong to one of these combinations. An example for such a consistent combination from L_3-R_4 is displayed in Fig. 4.17. Users, whose training data can be categorized into one of the other combinations, selected inconsistent training data. Inconsistency here refers to the effect that the robot's kinematics do not allow a movement from one posture to the other while respecting the task and the environmental constraints. Although the users who selected inconsistent redundancy combinations are in the minority as shown in Tab. 4.1, the reasons why users provided this training data and how to account for this behavior need to be discussed further, which is done at the end of this chapter and in Chap. 6.

Joint Limits of the LWR IV

The fact that a majority of 29 study participants selected consistent combinations (cf. Tab. 4.1) but only 9 users actually managed to successfully teach the system in terms of both measurements indicates that finding the consistent combinations of redundancy resolutions does not guarantee teaching success. Therefore, a qualitative difference between the users' data even within the consistent redundancy combinations is hypothesized. From a visual inspection of the data the hypothesis arises that during the teaching the users are not aware of the robot's joint limits. As a result some users provided training data very close to these limits in both training areas and some recorded data even slightly beyond the limits manifested in the system's software layers[5]. Fig. 4.17 shows an example of a participant's recorded training postures that are consistent with each other (combination L_3-R_4) but comprise joint values of $q_6^{\text{fixed}} \approx 119°$ for the left and $q_6^{\text{fixed}} \approx 113°$ for the right training area, respectively, while the limits for this joint are specified as $\pm 115°$. However, the learning methods analyzed in this thesis are completely data-driven and model-free which is desired for the reasons discussed in Sect. 2.3. Therefore, the learned mapping will generalize also slightly beyond the LWR IV's

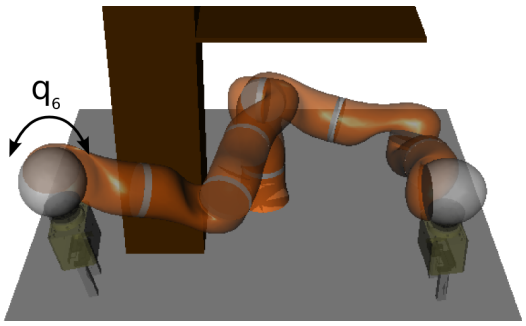


Fig. 4.17: Example for a consistent combination of redundancy resolutions (L_3-R_4) in left and right training area but with data very close to the LWR IV's joint limits.

[5] This was possible due to the compliance of the LWR IV. As described in Chap. A the joint limits defined within our control architecture are slightly more conservative than the LWR IV's actual hardware limits. Although our low-level software layer rejects any joint commands exceeding these limits, it does not prevent the users to go beyond them during the physical human-robot interaction and record data that slightly exceed these limits

joint limits when provided with such data, but the control architecture rejects such joint commands exceeding the limits. The resulting joint commands sent to the robot then exhibit inaccuracies, i.e. the task space error E_{task} increases. This effect is statistically significant as a correlation analysis between the error E_{task} and the distance $d(q_6^{\text{fixed}}, q_6^{\text{limit}})$ of the users' recorded joint values to the respective joint limit reveals. For the left training area this correlation is represented by $\text{r} = -0.71$ and for the right area by $\text{r} = -0.69$, both being highly significant with $\text{p} < 0.005$.

Distribution of Data per Training Area

While the previous analysis focused on the initial postures $\mathbf{q}^{\text{fixed}}$ the participants selected for the left and right training area, respectively, a third hypothesis relates to the distribution of the training data within each area. However, during a correlation analysis none of these parameters showed up to be statistically relevant for neither the task space error E_{task} nor the number of collisions E_{coll}. Therefore, this hypothesis is regarded as rejected. This is in line with the results obtained in Sect. 3.3.3, revealing that the utilized ELM ($\varepsilon = 1$) approach is robust against the choice of number of training data per area.

4.4 Discussion and Related Work

The purpose of this chapter was to investigate the proposed interaction scheme from the user point of view. One purpose of the user study was to validate the design of the structured interaction and the developed interaction controllers.

Concerning the general user experience with the robot system, the results show that the system was generally appreciated by the study participants as the handling of the robot was rated to be easy and self-explanatory and the feedback to be helpful. The ratings concerning threat and reliability of the system also serve as good indicators for the general acceptance of such collaborative systems in future manufacturing systems. However, some users rated the cognitive load they had during the interaction to be high. Regarding this result in the context of the analysis of the user's teaching success in the previous section, I conclude that finding appropriate postures in left and right training area might be difficult for some users. Still, the majority of the participants had no problem in selecting consistent training postures, more than half of the users managed to teach the system to avoid collisions with the obstacles and nine participants configured the system successfully in all regards. Furthermore it is noteworthy, that the results of the study must be regarded with the background that most of the user had no or only very few experiences with robots and only had an introductory warm-up phase of approximately 2 to 5 minutes. Thus, the results clearly show that the proposed interaction approach enables non-expert users teach null-space constraints to resolve the robot's redundancy. The constraints are modeled implicitly in the robot's joint space but also convey information about an implicit scene model in tasks space as

the experiments in Sect. 4.2 revealed. Hence, by means of the training data the human tutor transfers his or her knowledge about the environmental constraints to the system's motion controller, while the accuracy of the conveyed implicit model can be shaped by the number of training areas shown to the system.

The findings of the analysis in section Sect. 4.3.4 can be utilized to devise concrete improvements for the interaction scheme and the system prototype. As the system's feedback during the interaction already was rated helpful by the participants, I propose to enhance it with online information about the current state of the system. This includes online feedback about the quality of training data and the current learning state of the system such as force feedback about the training areas and the selected training postures. This proposed variant of the interaction model for the *CONFIGURATION* stage is developed in Chap. 6. Also raising the users' awareness of the robot's joint limits would help to improve the teaching success as well as the interaction experience.

To the best of my knowledge, only few work addressed the feasibility of programming-by-demonstration methods from a user-centered point of view. In [15] and [50] the authors investigate kinesthetic teaching from the user's perspective, by comparing standard trajectory-based kinesthetic teaching to their key-frame-based approach. Key-frames are recorded as sparse points in the state space of the robot. They found, that each of both demonstration paradigms has its advantages, which is slightly different to the results obtained in our studies presented in Sect. 1.2.1 and Chap. 5. The results even reveal, that recording key-frames took longer than providing trajectories. Furthermore, relying only on key-frames neglects the information about the timing of a task. In addition, the approach is based only on the demonstrating-in-configuration-space paradigm as discussed in Sect. 2.2.1.

Chapter 5

Assisted Programming in Task Space

The goal of this chapter is to evaluate the proposed interaction scheme on the next level, i.e. during the *PROGRAMMING* phase which contributes to the idea of reducing the teaching complexity for non-expert users. After having configured the robot in the previous *CONFIGURATION* stage, the learned redundancy resolution can be included into the robot's control architecture in order to support the human teacher during the interaction. This idea allows the user to freely interact with the manipulator's end-effector in the demonstrating-in-task-space strategy (cf. Sect. 2.2.2) while the joints are controlled by the robot to resolve the redundancy. As a result, the tutor can focus on teaching the temporal and positional aspects of the actual task by neglecting the task-independent constraints. Fig. 5.1 shows a shop-floor worker from the Harting company [24] during that interaction stage teaching a specific task trajectory to the 7-DoF KUKA Lightweight Robot IV. The hypothesis of this chapter is therefore that

> *the proposed interactive assisted programming reduces the complexity of kinesthetic teaching of redundant robots.*

In order to test this hypothesis, in the following the implementation of the proposed interaction model for the *PROGRAMMING* stage is introduced. The subsequent evaluation is based on the results obtained in the user study FlexIRob@Harting [22] and analyzes the interaction scheme both from the human perspective and the viewpoint of teaching success.

5.1 Interaction Model

The implementation of the interaction model follows the workflow outlined in Sect. 2.3.3 (Fig. 2.8) using the described interaction triggers `on_affected` and `on_converged` to physically guide the robot to the starting position of a trajectory demonstration as well as starting and stopping the recording.

The *assisted gravity compensation* controller is implemented based on the low-level joint impedance controller [25] provided by the KUKA Lightweight Robot IV as illustrated in Fig. 5.2. As already described in Sect. 4.1, external forces f_{int}

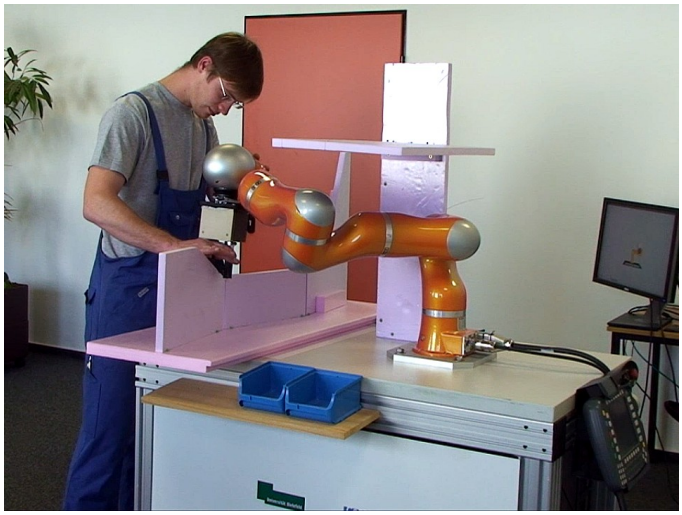


Fig. 5.1: A shop-floor worker from Harting [24] interacting with the FlexIRob system during the *PROGRAMMING* stage. He teaches a 3-D trajectory in a confined workspace while already being assisted by the robot's control system to avoid the static obstacles in the scene [22].

applied by a human operator are measured by the torque sensors and generate motor torques according to the impedance-based control scheme in [25]. This inner control loop acts like a spring-damper system allowing the user to physically deflect the manipulator from the commanded joint configuration $\mathbf{q}_{\text{cmd}}$. In the proposed implementation this deflection is measured by a changed position $\mathbf{x}_{\text{curr}}$ in task space and fed to an additional damping term to prevent the robot from continued drifting after being moved. Hence, the low-level task space controller π_x allowing interaction according to the demonstrating-in-task-space paradigm as discussed in Sect. 2.2.2 is a combination of the LWR IV's joint impedance controller and the added damping term in Cartesian space. In the outer control loop, the resulting desired task space position $\mathbf{x}$ is then set as target for the hierarchical controller employing the learned redundancy resolution $\mathbf{q}_c = \mathbf{q}_c(\mathbf{x}_{\text{des}})$ to calculate a joint configuration $\mathbf{q}_{\text{cmd}}$. The latter simultaneously realizes the desired end-effector position $\mathbf{x}_{\text{des}}$ and a self-motion in the manipulator's null-space to comply to the user-taught environmental constraints. The hierarchical controller is realized as a closed-loop controller, incorporating feedback about the current joint position $\mathbf{q}_{\text{fdb}}$.

Again, the main motion characteristics of the physical interaction, e.g. the ease of moving, are determined by the stiffness and damping values $\mathbf{k}$ and $\mathbf{d}$ of the underlying joint impedance controller. For the system prototype FlexIRob and the evaluation in the next section these were carefully chosen as detailed in Chap. A to allow smooth interaction.

It is worth noting that the specific implementation of the *assisted gravity compensation* controller is not uniquely determined by the utilized robot plat-

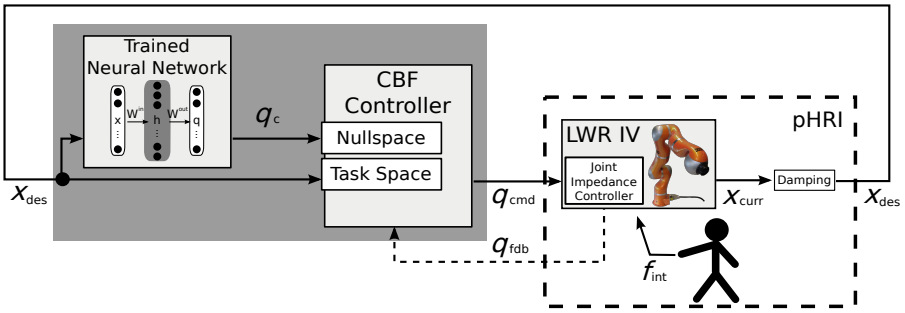


Fig. 5.2: Implementation of the proposed *assisted gravity compensation* controller concept from Sect. 2.3.3. The task space interaction controller π_x is implemented based on the low-level joint impedance controller of the KUKA Lightweight Robot IV and an additional damping term.

form. Depending on the platform-specific compliance features, sensors and available low-level interfaces, different solutions might exist to implement the required task space controller π_x. To demonstrate that, Fig. 5.3 shows an alternate implementation of π_x required to allow interaction according to the demonstrating-in-task-space strategy. Here, the LWR IV only uses its low-level joint position controller instead of the impedance controller. The external interaction forces f_{int} are sensed by the torque sensors and based on a precise dynamics model of the manipulator transformed to estimated external forces $\hat{f}_{\text{int}}$ at the end-effector. This functionality is provided by Kuka's fast research interface (FRI) [119]. A simple external Cartesian admittance controller is added calculating the desired end-effector position $\mathbf{x}_{\text{des}}$. However, although this latter implementation was shown to work in a short proof-of-concept scenario, it was not yet evaluated in a decent user study and is therefore not further considered in this chapter.

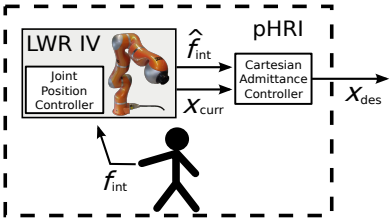


Fig. 5.3: Alternate implementation of π_x employing an external admittance controller.

On the one hand this demonstrates the feasibility of the proposed generic approach. Making no assumption on the low-level interfaces of the specific robot platform used, the approach can be applied to a variety of different manipulators. On the other hand, it points out that the haptic experience (admittance, jerk, stiffness) of the physical interaction is mainly determined by the interfaces and low-level controllers which are available to implement the interaction controller π_x.

5.2 FlexIRob@Harting: Assisted Programming Eases Teaching in Confined Spaces

In the following, the proposed *assisted gravity compensation* controller as implemented in the previous section is evaluated both from the teacher's perspective and in terms of teaching success. As the goal of this thesis is to enable lay users to intuitive teaching of redundant robots in confined spaces, the central research question regarded in this section is:

> *Does the derived* assisted gravity compensation *controller enable non-expert users trajectory-based kinesthetic teaching of redundant robots in presence of environmental constraints?*

In order to answer this question, in the following I present results from the user study FlexIRob@Harting [22]. The 49 participants most of whom never had worked with robots before were asked to teach a particular task trajectory to the 7-Dof KUKA Lightweight Robot IV. As already described in Sect. 1.2.1, they were asked to perform an adapted version of the wire loop game, namely guiding the end-effector along a styrofoam parcours but without getting the robot into contact with styrofoam obstacles that are placed in the robot's workspace as shown in Fig. 5.1. This teach-in scenario is motivated by typical industrial tasks such as gluing or welding often requiring that a robot arm is able to follow a demonstrated trajectory. Hence, the interaction model throughout this experiment is restricted to record only one demonstration which is regarded as a simple teach-in of a task trajectory.

To test the hypothesis that the *assisted gravity compensation* mode reduces the teaching complexity, the study participants have been divided into two groups:

Group A These particicpants were interacting with the robot supported by the *assisted gravity compensation* mode with a pre-trained redundancy resolution (ELM, cf. Sect. 3.2.1 and Chap. B) embedded to the hierarchical control architecture. Hence, the physical interaction followed the demonstrating-in-task-space paradigm; they only needed to guide the end-effector.

Group N As a control group, the other participants were using the *gravity compensation* mode as described Sect. 4.1. Thus, they were interacting with the LWR IV using to the demonstrating-in-configuration-space strategy throughout which they needed to take care of all joints and the end-effector simultaneously in order to avoid collisions with the obstacles while also performing the task programming.

In the following, the teaching success and user experience of the participants is evaluated. For the detailed study design and the experiment course for each participant please refer to Chap. B.

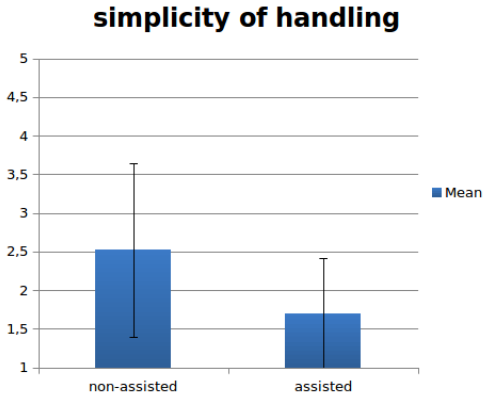


Fig. 5.4: Average rating of simplicity of handling the LWR IV during the wire loop game.

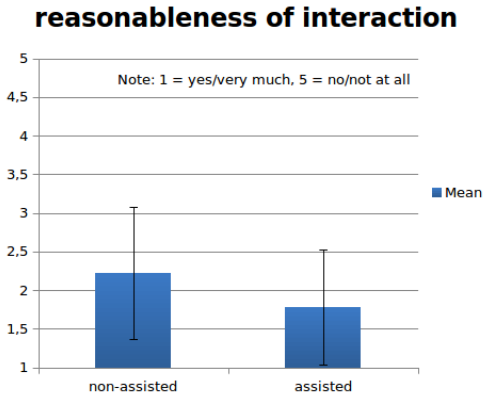


Fig. 5.5: Average rating of how reasonable the interaction with the LWR IV was during the wire loop game.

5.2.1 Analysis of the Teaching Experience

As described in Chap. B, after the experiment each participant was asked to answer a questionnaire concerning the interaction experience with the FlexIRob prototype during the wire loop game. These questions asked the participant to rate the *simplicity* and *pleasantness* of handling the robot as well as the *reasonableness* of the interaction mode. As the main purpose of the derived *assisted gravity compensation* controller is to simplify the teaching procedure, the hypothesize is that participants of group A, being assisted by the robot, would find the task programming easier and more pleasant and would rate the interaction mode more reasonable than those of group N, i.e. without assistance.

In order test these hypotheses, in [22] an analysis of variance (ANOVA) was computed with condition (group A vs. group N) as the independent variable and the variables concerning the simplicity and pleasantness of the handling and the reasonableness of the interaction mode as dependent variables. I briefly summarize the important aspects here. For the sake of readability, only means and standard deviations are described in the text. For other relevant values see Tab. 5.1. As expected, the ANOVA revealed a highly significant effect on the simplicity of handling. Participants using the *assisted gravity compensation* controller found the handling significantly easier ($M = 1.70$, $SD = 0.71$) than did participants of the control group N ($M = 2.52$, $SD = 1.12$), see Fig. 5.4. A marginally significant effect was also found for the interaction mode. Participants in group A rated it significantly more reasonable ($M = 1.78$, $SD = 0.74$) than those in group N ($M = 2.22$, $SD = 0.85$), see Fig. 5.5. Thereby, low values indicate yes/very much, and high values indicate no/not at all.

The ANOVA was again computed with covariates that could have influenced

Tab. 5.1: Main effects regarding the condition (group A vs. group N).

	F	df	p	η^2
effect on **simplicity of handling** (without covariates)	8.59	1.42	< 0.01	0.17
effect on **easonableness of interaction** (without covariates)	3.34	1.42	< 0.10	0.07
effect on **simplicity of handling** (with covariates)	3.17	1.37	< 0.10	0.08
effect on **instinctiveness of handling** (without covariates)	2.83	1.42	< 0.10	0.06

the effects. Covariates were identified correlating all control variables with the dependent variables. Covariates included in the analysis were the length of time the participants had worked at the company, their experience with computers, their spatial vision, and their spatial imagination, as these variables significantly correlated with one or several of the dependent variables. Including these covariates, the results showed a marginally significant effect for the simplicity of handling. Again, participants using the *assisted gravity compensation* controller rated the handling significantly less difficult than those of the control group N.

In [22], we reported also another very interesting effect. As we were also interested in any differences concerning the general experience with our robot, an ANOVA was conducted with condition (group A vs. group N) as the independent variable and all variables rating the general experience with the robot as dependent variables. The general experience here relates to the ratings that we assessed from the participants not specifically for the wire loop game but for the general experience throughout their interaction with the FlexIRob system (cf. Chap. B for details). A marginally significant effect on the instinctiveness of handling could be found, indicating that the non-assisted group (M = 2.71, SD = 1.01) rated the handling less instinctive than did the assisted group (M = 2.17, SD = 1.11). This effect is very interesting as participants were instructed to rate the wire-loop game separately. It indicates that the positive effect of the interaction mode during the wire loop game was strong enough to influence the whole experience with the robot also during the other parts of the study. However, this effect vanished when computing the ANOVA a second time with covariates that could have influenced the effects. These included gender, computer experience, spatial vision, spatial imagination, prior experience with robots, number of robots known and the department at HARTING the participants worked at.

5.2.2 Analysis of the Teaching Success

Like for the analysis of the *CONFIGURATION* in Sect. 4.3.3, the quantitative success of the teach-in is accessed with three measurements, evaluating effectiveness and efficiency of the physical human-robot interaction [21]:

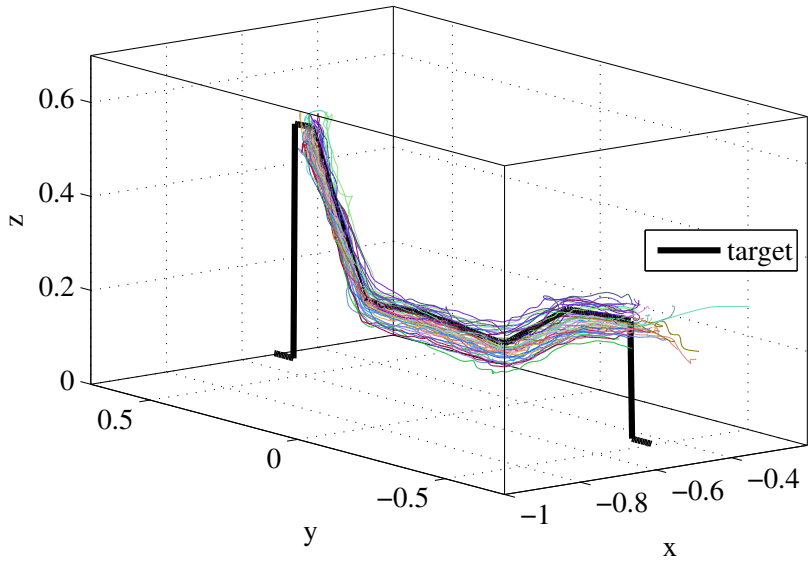
Task space accuracy As the teaching task relates to a specific desired end-effector trajectory, effectiveness can be assessed by means of the task space accuracy of the recorded demonstration.

Monitoring collisions Effectiveness also relates to the abidance of environmental constraints and is accessed by counting unintended contacts, i.e. collisions E_{coll}, of the manipulator with the obstacles. This is done automatically in the utilized simulation software of the FlexIRob prototype. For details on the used software please refer to Chap. A

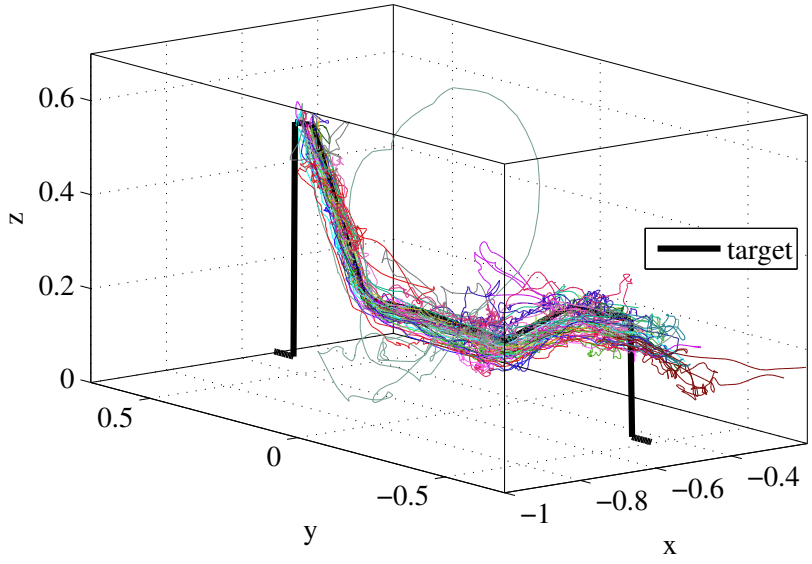
Efficiency is measured by the time needed for the teach-in.

Fig. 5.6 shows the teach-in trajectories that have been recorded by the participants of both groups. It clearly reveals that the trajectories of group A (see Fig. 5.6(a)) are smooth, very similar to each other and close to the target, that was represented by the styrofoam parcours during the study and is plotted in Fig. 5.6 as black line. In contrast, participants of group N (see Fig. 5.6(b)) recorded trajectories that are jerky, deviate a lot amongst each other and in some cases exhibit strong error to the target trajectory. I explain the observed high errors as follows: Due to the fact that this group was not assisted in finding a valid joint configurations, some users started the wire loop game game with a very unsuitable posture (e.g. "elbow right"). At the beginning of the parcours, this posture is not problematic and results in a correct end-effector position. However, in the further course of the parcours the environmental constraints (confined workspace) require a different redundancy resolution (e.g. "elbow left"). Now, these users have to change the robot's joint configuration (e.g. from "elbow right" to "elbow left") while performing the wire-loop game which, for some users, produces the observed high task space deviations.

In order to evaluate the difference between the two conditions in terms of task-space accuracy systematically with reasonable measures, two metrics are used. First, the maximum Cartesian deviation of a user's trajectory from the target trajectory is measured. The results are shown in Fig. 5.7(a). They unveil that assisted participants stay significant closer to the target trajectory than the non-assisted and do not deviate a lot amongst each other, namely 4.9 ± 1.4 compared to 12.2 ± 11.1 centimeters on average. Second, the geometrical shape of the teach-in trajectory is compared with the target by means of a Procrustes analysis [120]. Since the users were not explicitly encouraged to perform a specific timing during the teach-in, the trajectories are normalized in time, i.e. the velocity profile is removed by re-sampling the data with constant velocity and equal number of points. For comparison, the user trajectory and target trajectory are optimally superimposed by means of translation and rotation (I omit scaling in this analysis). Finally, the Procrustes difference is calculated as the average Euclidean distance between the points of both trajectories. The results in Fig. 5.7(b) show the significant higher geometrical matching with the target trajectory for assisted users. The average Procrustes distance for group N measures $15.9 \pm 1.3\,\text{cm}$, which means that even after optimal translation and rotation the geometrical shapes of teach-in trajectory and target trajectory on average differ by 16 centimeters per point. In contrast, the Procrustes difference for group A measures only $1.8 \pm 0.7\,\text{cm}$.

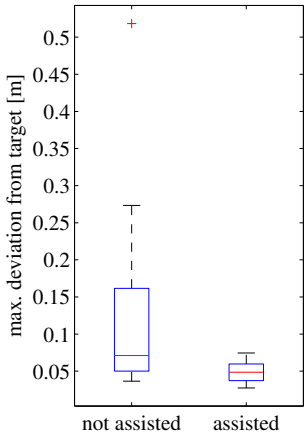


(a) Group A (assisted)

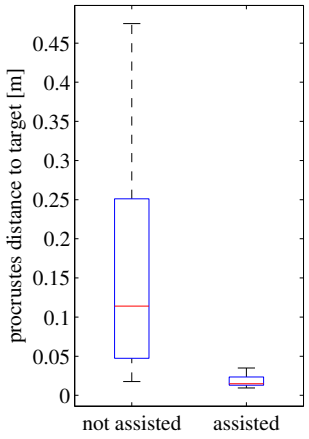


(b) Group N (non-assisted)

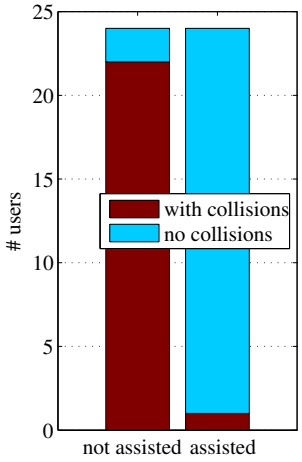
Fig. 5.6: Recorded demonstrations of the participants during the wire loop game. The target trajectory relating to the the styrofoam parcours is plotted as black line. Group A used the proposed *assisted gravity compensation* controller and participants in group B the *gravity compensation* from Sect. 4.1 without any assistance for redundancy resolution.



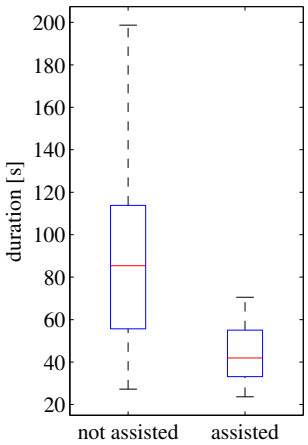
(a) Maximal deviation of the participants' teach-in trajectories from the target trajectory.



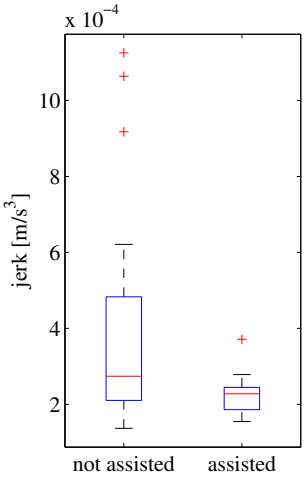
(b) Procrustes difference of the users' teach-in trajectories to the target trajectory.



(c) Number of participants with and without collisions.



(d) Time needed by the participants for the teach-in.



(e) Non-smoothness of the participants' teach-in trajectories in terms of jerk.

Fig. 5.7: Evaluated metrics for accessing the participants' teaching success during the wire loop game relating to system effectiveness (a-c), efficiency (d) and additional metrics (e).

As for collisions with environmental obstacles[6], 22 of 24 non-assisted participants induced collisions during the teach-in. In contrast, only one of 24 participants in group A induced unintended contacts between the robot arm and the environment (see Fig. 5.7(c)).

System efficiency is measured by the time needed by the participants for the teach-in task. The results are displayed in Fig. 5.7(d). Again, the data reveal a significant advantage of the assisted users. Whereas participants of group N needed 93.4 ± 44.5 seconds on average, participants of group A required 44.9 ± 13.9 seconds, i.e. only half the time for the same task. As a side effect of the improved system effectiveness and efficiency, the trajectories recorded by participants of group A are much smoother in terms of jerk calculated as the second derivative of the trajectories' velocity profile with respect to time, see Fig. 5.7(e).

All of the presented differences between participants of group A and group N have been reported to be statistically significant [22, 23].

5.3 Discussion of Results

The purpose of this chapter was to implement the proposed *assisted gravity compensation* on a real robotic system and evaluate it with non-expert users in an industrial-like setting both from the user point of view and the teaching success.

According to the implementation, I point out that the proposed concepts do not rely on any specific requirements regarding the compliance features, low-level controllers or interfaces of the utilized robot platform. In contrast it only uses them to implement the outer control loop as discussed above according to the specific available sensors and controllers. Hence, the presented concept of assisted kinesthetic teaching can be applied to a variety of currently available industrial, humanoid or other redundant, compliant robots such as [17]. However, the feasibility of the resulting approach must always be evaluated with respect to a specific platform.

As for this evaluation, I presented results from a large user study with non-expert users where the proposed *assisted gravity compensation* was implemented on a KUKA Lightweight Robot IV. During the study users with no robotic expertise performed a practically relevant teach-in procedure. The results reveal not only a high system effectiveness and efficiency of the human-robot interaction in terms of task space accuracy measurements, collision avoidance and time to completion, which all were significantly improved using the *assisted gravity compensation* compared to a control group. Collisions with the environment were almost completely eliminated[7]. They also attest a positive effect on the teaching experience from the

[6] Collisions are automatically detected in our simulation software where the robot and the environmental obstacle are accurately modeled.

[7] The single remaining collision can be attributed to an interaction failure but is not related to the *assisted gravity compensation* concept as a visual inspection of the participants teach-in procedure shows.

user's point of view. During the study we found significant effects indicating a perceived simplification of the teach-in procedure and an increased reasonableness of the physical human-robot interaction. While these results completely confirmed the expectations, no significant effect (positive or negative) could be found for the *assisted gravity compensation* on the pleasantness of handling. However, I report that on average all users rated the pleasantness to be good ($M = 2.02$, $SD = 1.0$). This can be interpreted such that already the used demonstrating-in-configuration-space controller *gravity compensation*, as implemented in Sect. 4.1 and completely relying on the low-level compliance properties of the LWR IV, is very pleasant for physical guidance of the robot. Hence, adding the outer control loop with the hierarchical controller does not disturb the pleasantness of handling.

Summarized, the proposed *assisted gravity compensation* controller helped to simplify teaching of the redundant LWR IV in a confined workspace which was a major goal of the derived concepts in this thesis.

Chapter 6

Assistance Blending for Teaching Redundancy Resolutions

The previous chapters demonstrate that the process of teaching and learning redundancy resolutions is a valuable approach to efficiently transfer, encode and generalize the human's implicit knowledge about environmental constraints. The proposed *assisted gravity compensation* control scheme even allows to efficiently exploit this knowledge already in the subsequent human-robot interaction for assisted programming of tasks. However, as indicated by the analysis of the users' provided demonstrations in Sect. 4.3.4, treating the programming-by-demonstration paradigm as a one way channel during the *CONFIGURATION* stage reduces the desired effect of intuitive and efficient teaching. The tutor might not be aware of the robot system's current capabilities (learned redundancy resolution) and limitations (e.g. joint limits). As a result, in the FlexIRob@Harting study some users failed to successfully configure the robot according to the constraints imposed by the environment.

The goal of this chapter is therefore to derive a new interaction model for the *CONFIGURATION* stage that allows the tutor to "experience" the robot's current state of knowledge via a shared *assistance blending* control mode. It is inspired by research on policy-blending for shared human-robot control in teleoperation scenarios [121] and intended to close the interaction loop between tutor and learner [122, 123] by providing feedback about the robot's current state of knowledge. The idea is to (a) use the robot's *gravity compensation* π_q controller where no information about demonstrated constraints is available, (b) the *assisted gravity compensation* controller when moving in the vicinity of already taught constraints, and (c) continuously blend between both control schemes. This shared, physical human-robot interaction provides haptic feedback about where in the workspace the robot already has been trained, and how. The underliyng assumption is that

> *feedback about the system's current state of knowledge helps the user to provide more suitable demonstrations and further reduces the interaction complexity.*

Within this chapter, the adapted interaction workflow during the *CONFIGU-*

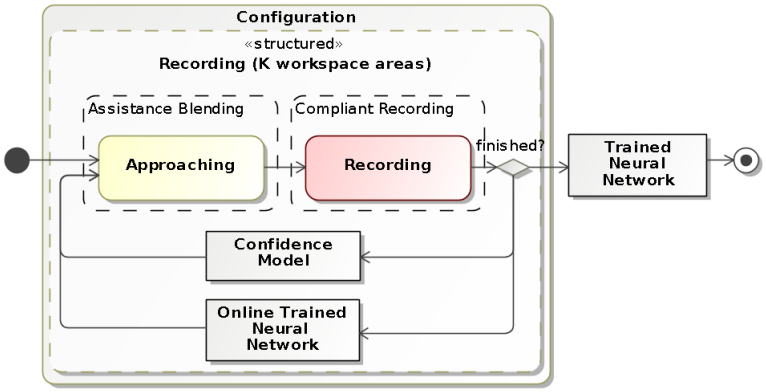


Fig. 6.1: Illustration of the proposed adoptions to the interaction workflow for kinesthetic teaching of redundancy resolutions during the *CONFIGURATION* stage. During *APPROACHING* the proposed *assistance blending* controller is utilized, embedding the online trained neural network and a confidence model in the control architecture.

RATION stage is described and the necessary concepts to implement the proposed *assistance blending* interaction controller are presented. These include algorithms for learning redundancy resolutions online, and estimating a confidence signal from the training data for blending between controllers. Please note, that the proposed concept is not yet validated in a user study. Therefore, a decent evaluation is missing in this chapter. However, the derived concepts have been implemented and tested to be working in our system prototype FlexIRob. The chapter closes with a short discussion of the proposed concepts relating them to findings from research with socially guided human-robot interaction.

6.1 Interaction Model

The changed interaction workflow of the *CONFIGURATION* stage is depicted in Fig. 6.1. As usual, the user guides the robot to relevant workspaces (*gravity compensation*) and records demonstration data (*compliant recording*). Again, the interaction triggers used to switch between these stages are `on_affected` and `on_converged`.

The difference to the interaction model introduced in Sect. 2.3.1 is that during *compliant recording* online learning algorithms are utilized to directly infer a redundancy mapping $\mathbf{q}_c(\cdot)$ from the demonstration data. An adapted variant of the ELM (cf. Sect. 3.2.1) is used in this chapter as briefly reviewed in the following section. This can be exploited in subsequent *APPROACHING* phases to assist the user in redundancy resolution according to the ideas discussed in Chap. 5. However, generalization abilities of the learned redundancy mapping $\mathbf{q}_c(\cdot)$ during

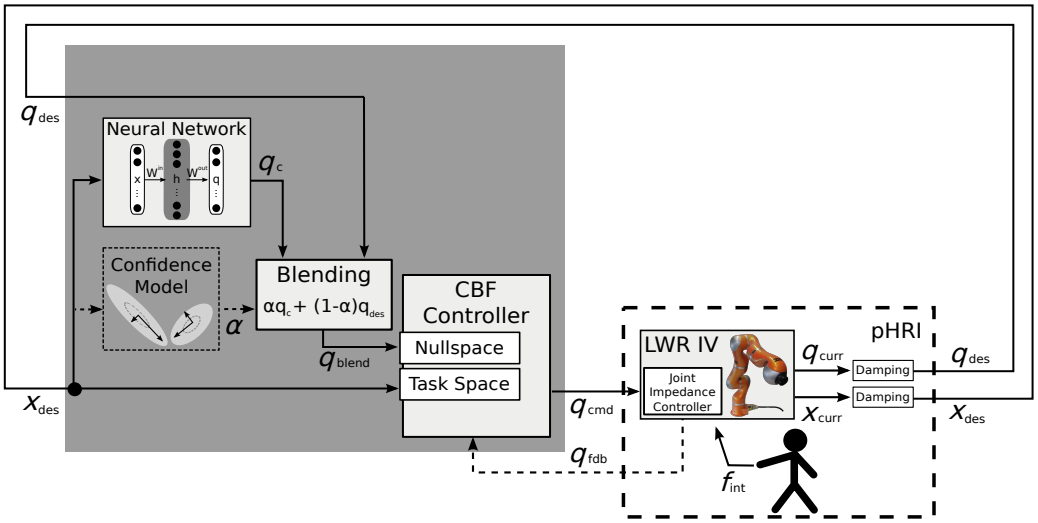


Fig. 6.2: Proposed *assistance blending* controller.

the *CONFIGURATION* stage will be limited, since it is based only on the so far provided demonstrations. A confidence model C is required for the robot to infer when to rely on the learned mapping and when not [121]. In this work, a confidence model is a mapping

$$C : \mathbb{R}^M \mapsto [0, 1], \qquad \alpha = C(\mathbf{x}) \tag{6.1}$$

based on the position data $\mathbf{x}_{\text{curr}}$ of the robot during the interaction and results in a confidence value $\alpha \in [0, 1]$, where higher values relate to higher confidence. Methods to learn such a confidence model are proposed in Sect. 6.3.

Both confidence model $C(\cdot)$ and learned redundancy mapping $\mathbf{q}_c(\cdot)$ are then embedded in subsequent *APPROACHING* stages by means of the *assistance blending* control scheme shown in Fig. 6.2. It is implemented on the LWR IV's joint impedance mode [5] as already explained in Sect. 4.1. By physically deflecting the manipulator from the commanded joint configuration $\mathbf{q}_{\text{cmd}}$, the user generates a desired change in both the robot's configuration $\mathbf{q}_{\text{des}}$ and $\mathbf{x}_{\text{des}}$. The damping again is added to prevent the robot from continued drifting after being moved. Based on the task space position $\mathbf{x}_{\text{des}}$ the control system calculates a proposed redundancy resolution $\mathbf{q}_c$ by means of the embedded neural network, and retrieves a confidence value $\alpha = C(\mathbf{x}_{\text{des}})$ from the learned confidence model. A blending component then mediates between the user intended $\mathbf{q}_{\text{des}}$ and the learned null-space constraint $\mathbf{q}_c$:

$$\mathbf{q}_{\text{blend}} = \alpha\, \mathbf{q}_c + (1 - \alpha)\, \mathbf{q}_{\text{des}}. \tag{6.2}$$

The proposed control scheme therefore dynamically blends between the demonstrating-in-configuration-space and the demonstrating-in-task-space strategy depending on the robot's confidence model. Consider for instance the extreme

case of the first *APPROACHING* stage ($k = 1$) when no training area was demonstrated by the user yet. The confidence of the robot should be $C \equiv 0$ for all positions in the workspace. As a result, the null-space constraint passed to the hierarchical controller $\mathbf{q}_{\text{blend}} = \mathbf{q}_{\text{des}}$ relates to the damped current position $\mathbf{q}_{\text{curr}}$ of the robot. Hence, effectively the *gravity compensation* controller as described in Sect. 4.1 is used during the interaction. The other extreme case describes the (undesired) situation where the user demonstrated redundancy resolutions exhaustively throughout the workspace such that the robot's confidence is $C \approxeq 1$. As a result, effectively the *assisted gravity compensation* control scheme is used all over the workspace as described in Sect. 5.1, reducing the interaction effort for the user. For all intermediate situations, the proposed *assistance blending* controller mediates between *gravity compensation* and *assisted gravity compensation* based on the robot's confidence model, allowing the robot to assist the user in workspace areas with high confidence and the user to freely interact with all joints in low-confidence areas.

6.2 Online Sequential Extreme Learning Machine

In order to implement the proposed interaction scheme, a learning algorithm is required that allows to process the data $\mathcal{D}^{(k)} = (\mathbf{x}_l^{(k)}, \mathbf{q}_l^{(k)})_{l=1,\dots,L^{(k)}}$ online during the *RECORDING* stage to allow the exploitation of the learned mapping directly in the subsequent *APPROACHING* phase.

In [116] the authors propose an online sequential learning algorithm for single-hidden-layer feedforward networks, such as the ELM discussed in Sect. 3.2.1. In fact, the algorithm uses the ideas of ELM [112] and adapts them to an online-capable variant of it (OS-ELM). In the following, the necessary adaptions are briefly revised, as the OS-ELM algorithm is used in the remainder of this chapter.

We consider the same, three-layered feed-forward network architecture as depicted in Fig. 3.4 with fixed, randomly initialized input weights W^{in}, an input, hidden and output layer $\mathbf{x} \in \mathbb{R}^M$, $\mathbf{h} \in \mathbb{R}^R$, $\mathbf{y} \in \mathbb{R}^N$, respectively, as well as parameterized activation functions $f_i(x) = (1 + \exp(-a_i x - b_i))^{-1}$ with fixed, randomized parameters a_i, b_i drawn from a uniform distribution. During execution of inputs $\mathbf{x}$, the network's state is governed by the update equation

$$\hat{\mathbf{y}}(\mathbf{x}) = \mathsf{W}^{\text{out}}\,\mathbf{h}(\mathbf{x}) \qquad \text{with} \qquad \mathbf{h}(\mathbf{x}) = \mathbf{f}(\mathsf{W}^{\text{in}}\,\mathbf{x}). \tag{6.3}$$

Like in the ELM approach, the only learning parameters adapted during training are the output weights W^{out}. However, in this section they are adapted online [116]. For a given training sample $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$ the output weights W^{out} are adapted according to

$$\mathsf{W}^{\text{out}}_{k+1} = \mathsf{W}^{\text{out}}_{k} + (\mathbf{y}_{k+1} - \mathsf{W}^{\text{out}}_{k}\,\mathbf{h}_{k+1})\,\mathbf{h}_{k+1}^{\mathsf{T}}\,\mathsf{P}_{k+1}, \tag{6.4}$$

$$\mathsf{P}_{k+1} = \mathsf{P}_k - \frac{\mathsf{P}_k\,\mathbf{h}_{k+1}\,\mathbf{h}_{k+1}^{\mathsf{T}}\,\mathsf{P}_k}{1 + \mathbf{h}_{k+1}\,\mathsf{P}_k\,\mathbf{h}_{k+1}^{\mathsf{T}}} \tag{6.5}$$

where $\mathbf{h}_{k+1} = \mathbf{f}(\mathsf{W}^{\text{in}}\,\mathbf{x}_{k+1})$. These recursive formulas are derived from matrix algebra and are similar to the well known recursive least squares algorithm [101].

This approach combines the advantages of the ELM approach concerning efficient computation with online learning capabilities. However, as already mentioned in Sect. 3.2.1 and intensively analyzed in [107], model selection is an issue also for ELM approaches. Therefore, again a regularization parameter ε is introduced that prevents large values for W^{out} (cf. Eq. (3.12)) and thus over-fitting. This is reflected in the the initialization of P and W^{out}, which I propose to choose according to $\mathsf{W}_0^{\text{out}} = \mathsf{0}_{N\times R}$ and $\mathsf{P}_0 = (\varepsilon\,\mathbb{I}_{R\times R})^{-1}$, where $\mathbb{I}$ refers to the identity and $\mathsf{0}$ to a zero matrix, respectively. Note that this implementation allows online learning with the first data sample, which is different to the work in [116], where an initial batch learning phase is required.

6.3 Learning a Confidence Model from Training Data

The confidence model $C(\cdot)$ proposed in this chapter relies on the distribution of the provided demonstration data in task space. The idea is to generate high confidence $\alpha \approx 1$ when close to the provided training data and low confidence $\alpha \approx 0$ far away from any training data.

The implementation proposed in this section utilizes a set of K gaussian-like responsibility functions

$$g^{(k)}(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^{\mathsf{T}}\Sigma_k^{-1}(\mathbf{x}-\mu_k)}, \tag{6.6}$$

one for each training area. The corresponding mean μ_k and covariance matrix Σ_k are estimated from the training data $(\mathbf{x}_l^{(k)})_{l=1,\ldots,L^{(k)}}$ provided by the user during *RECORDING* in the k-th area:

$$\mu_k := \frac{1}{L^{(k)}}\sum_{l=1}^{L^{(k)}}\mathbf{x}_l^{(k)} \tag{6.7}$$

$$\Sigma_k := \frac{d}{L^{(k)}-1}\sum_{l=1}^{L^{(k)}}(\mathbf{x}_l^{(k)}-\mu_k)(\mathbf{x}_l^{(k)}-\mu_k)^{\mathsf{T}}. \tag{6.8}$$

Once these parameters are estimated, $g^{(k)}$ is added to C and a confidence value can be retrieved by querying all of these responsibility functions and returning the maximum value, i.e according to:

$$C(\mathbf{x}) = \max_{k=1,\ldots,K} g^{(k)}(\mathbf{x}). \tag{6.9}$$

As described in Sect. 6.1, during the *APPROACHING* stage the current task space position $\mathbf{x}_{\text{curr}}$ of the robot during the interaction with the user is used to query the confidence. As each of the functions $g^{(k)}$ implements a mapping from task space

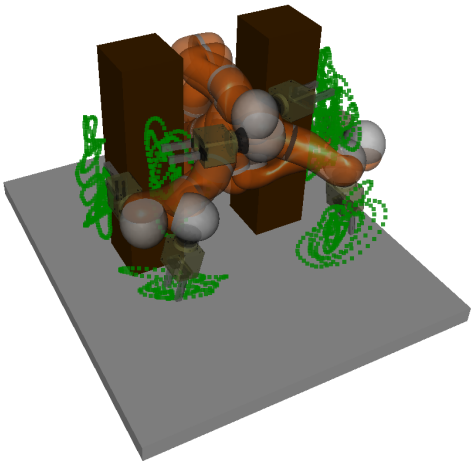
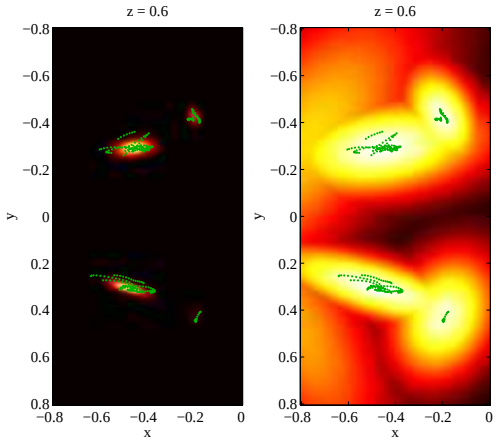


Fig. 6.3: Exemplary demonstration data $(\mathbf{x}_l^{(k)})_{l=1,\dots,L^{(k)}}$ recorded during *RECORDING* in $K = 6$ training areas for the construction of the confidence model C.



(a) $d = 1$ (b) $d = 50$

Fig. 6.4: Estimated confidence model C in the robot's workspace visualized for a fixed z-value and different parameters d.

to $[0, 1]$, it also holds that $\alpha = C(\mathbf{x}_{\text{curr}}) \in [0, 1]$ for any task space position $\mathbf{x}_{\text{curr}}$ of the robot during the interaction with the user.

Note, that the proposed confidence model C introduces a new parameter d to account for the fact that estimating gaussian distributions from non-gaussian data yields undesirable results. This effect is illustrated in Fig. 6.4, where the confidence model is shown based on the recorded data in Fig. 6.3. Setting the parameter to $d = 1$ relates to the sample-covariance estimator, which results in poor confidence even very close to the training data. In contrast, the parameter must not be set too high, as e.g. $d = 50$ generates high confidence even in areas far away from the training data. This might prohibit the user to intervene with corrective demonstrations in areas where the so-far learned redundancy mapping $\mathbf{q}_c(\cdot)$ yields bad results yet.

For accurate estimation of data distributions typically more advanced methods are used [124]. However, as these typically require high computational effort, such methods can be disregarded in the context of this thesis. Instead, the parameter d is utilized. In short tests with the system prototype FlexIRob, good results are obtained for any value of $10 \leq d \leq 25$, as e.g. shown in Fig. 6.5 for $d = 15$. Depending on the variance of the demonstrations, the confidence model C generates high confidence values within the demonstration data, but also in between training areas ($z = 0.1$, $z = 0.3$) and even extrapolates slightly beyond them ($z = 0.9$).

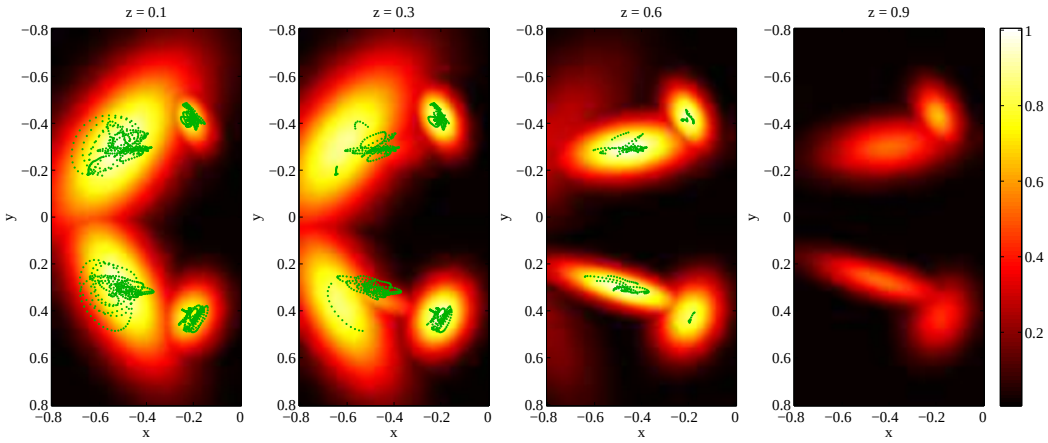


Fig. 6.5: Illustration of the learned confidence model C from the training data shown in Fig. 6.3. C is visualized in different heights $z = (0.1, 0.3, 0.6, 0.9)$ of the robot's workspace for the parameter $d = 15$.

6.4 Discussion and Related Work

This chapter presents an adapted interaction model for the *CONFIGURATION* stage, that accounts for the results obtained in the FlexIRob@Harting study as reported in Sect. 4.3.4. The proposed *assistance blending* controller dynamically and smoothly switches between the demonstrating-in-configuration strategy, which is the only strategy available when lacking knowledge about environmental constraints, and the demonstrating-in-task-space paradigm, which has been proven to reduce the interaction complexity on the user's side (cf. Sect. 5.2). For that purpose, a confidence model for the robot is constructed incrementally during the physical human-robot interaction that - in combination with online learning of redundancy resolutions - allows it to assist the user already during subsequent *APPROACHING* phases.

This strategy strongly relates to shared human-robot control approaches typically employed for teleoperation of mobile and/or dexterous manipulators [121, 125] e.g. for medical surgery [49, 126] or remote controlled humanoid robots, even in space [127]. The typical problem addressed in that line of research is the physical separation of robot and human operator, sometimes across very long distances. Aiming at an increased autonomy to not rely only on the human control input, these systems typically estimate the human intent and then used to arbitrate the humans (missing) input appropriately. In [121] the authors provide an interesting and exhaustive overview about existing methods for such policy blending in teleoperation.

In contrast, the approach presented in this chapter is employed in direct physical human-robot interaction. Arbitration of the users control inputs $\mathbf{q}_{\text{des}}$ in order to comply to already demonstrated constraints gives kinesthetic feedback about

where the robot already was instructed, and how. On the one hand, this reduces teaching complexity which is a major goal of the work in this thesis, since the user does not need to remember where the robot already was trained or to infer that from a two-dimensional projection on a computer screen. On the other hand, the feedback provides information about the current state of knowledge concerning the learned constraints. Since the confidence model allows the robot to evaluate the learned redundancy mapping also between and beyond the demonstrated areas, the user can test generalization already within the interaction, allowing to evaluate where demonstrations still are needed and where not.

I hypothesize, that this effectively shapes the teaching process according to ideas from research on socially guided demonstration learning. In contrast to traditional programming-by-demonstration paradigms, where interaction has been regarded as a one way channel (e.g. as such in [128]), the authors in [122] and [123] clearly state that both teaching and learning are intimately coupled. The instructor has to maintain a good mental model of the learner's internal state to provide appropriate *scaffolding*, i.e. structuring the learning process and guiding the learner's exploration accordingly. "[T]ransparency of the internal state of the machine could greatly improve the learning experience. By revealing what is known and what is unclear, the machine can guide the teaching process. To be most effective, the machine should use cues that will be intuitive for the human partner [...]" [122]. The work in [123] elaborates on this topic with a study about movement imitation learning on a humanoid robot showing that feedback about the robot's learning progress shapes the human's tutoring behavior. In the human-robot interaction scenario regarded in this thesis the communication between learner and teacher is based mainly on the direct physical interaction between the robot and the human tutor. Therefore, providing force feedback about the currently learned redundancy resolutions seems a natural and intuitive way to make the learning process transparent to the teacher.

However, decent evaluations of this hypothesis remain for future work and can not be tackled in the context of this thesis. Still, it is worth to be noted that the proposed approach is implemented and tested to be working on the real system prototype FlexIRob employing the KUKA Lightweight Robot IV.

Chapter 7

Autonomous Path Planning in Confined Spaces

One key aspect of the work presented in this thesis is to enable humans to transfer their implicit knowledge about environmental constraints to the robot's motion control system. The motivation behind that approach is to incrementally increase the robot's autonomy in order to reduce the interaction effort on the user's side. As analyzed in Sect. 4.2, this knowledge is conveyed in the demonstration data provided during the *CONFIGURATION* stage in form of constraints in the robot's configuration space, but also relates to present restrictions in the workspace. The experiment in Sect. 3.3.2 revealed, that it contains information about the obstacle-free task space encoded in the neural learner which generalizes it beyond the training areas. Embedded into the motion control architecture, it permits to decreases the interaction complexity for the user during the *PROGRAMMING* phase by providing assistance in avoiding the environmental constraints as shown in Sect. 5.2. Following this idea one step further, one might ask the question whether the provided data can be used to further increase the robot's autonomy, from an assistant teaching device to a semi-autonomously moving system. The hypothesis investigated in this chapter is that

> *the demonstration data of the* CONFIGURATION *stage already conveys sufficient information for the robot to autonomously plan and execute safe paths between the relevant working areas avoiding environmental constraints.*

In the following, a path planning approach is presented to address this hypothesis.

Notably, despite the investigation of the posed question, this approach is very interesting from an application-point-of-view. In typical industrial manufacturing or human-robot collaboration scenarios there will also be tasks like picking and placing objects where the specific task space trajectory between start and goal might not be important as long as it complies to the environmental constraints. For such applications it is desirable that the robot can move autonomously from start to goal. However, in the context of this thesis, state-of-the art planning methods where one can "take geometric descriptions of the robot and its static or dynamic environment for granted" [129] can not be applied. Since the proposed, purely data-driven approach presented in this thesis enables lay users to provide

such kind of "descriptions", it might be particularly interesting in applications where no information about constraints in the robot's workspace is given. This includes for instances frequently changed cluttered workspaces or scenarios where visual servoing of the robot and the constraints is not feasible.

This approach comprises several challenges since data gathered during the *CONFIGURATION* stage is typically sparse and is presented incrementally and correlated along the demonstrated trajectories. It must be processed online and in real-time to obtain an any-time planning capability. Furthermore, the manipulator's redundancy needs to be taken into account to avoid unintended switching of redundancy resolutions that can result in collisions with obstacles. To meet these requirements the instantaneous topological map (ITM,[130]) is utilized, the only state-of-the-art algorithm that can incrementally map correlated data. On that algorithmic basis, novel elements are introduced to cope with the sparse user data on the one hand and to account for the manipulator's redundancy on the other hand. Bootstrapping heuristics will improve the connectivity of the resulting navigation graph and a so-called joint space criterion will prevent unintended redundancy switches. For navigation in that graph standard algorithms to find the shortest path between start and goal position are utilized. The presented approach was introduced in [131] and published in [132].

The remainder of this chapter is organized as follows. First the resulting adaptations of the interaction model are described. Subsequently, the ITM as algorithmic basis for the construction of the navigation graph is briefly introduced and discussed, followed by the development of the necessary bootstrapping heuristics for an improved connectivity and the joint space criterion. Based upon that implementation, results of the proposed path planning method are presented for different workspaces. Finally the chapter closes with a short discussion of the results and related work.

7.1 Interaction Model

As illustrated in Fig. 7.1, the interaction model utilized in this chapter employs the *CONFIGURATION* stage as introduced in Sect. 2.3 utilizing the same interaction workflow (cf. Fig. 2.5) and the same interaction triggers `on_affected` and `on_converged`. Hence, with respect to the the user's viewpoint the interaction during the *CONFIGURATION* phase remains unchanged.

Concerning the subsequent course of interaction, the previous chapters relied on learning a task representation from demonstrations during the *PROGRAMMING* stage, which is logical in the tradition of classical kinesthetic teaching or programming-by-demonstration approaches. However, in this chapter a dedicated *PROGRAMMING* phase is not longer regarded to be necessary as the task is considered to be only goal-oriented and not manner-oriented [123]. Instead the user simply defines goal positions in task-space, e.g. by providing Cartesian coordinates

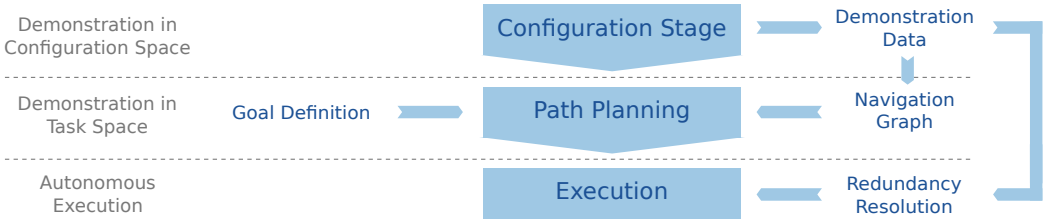


Fig. 7.1: Schematic view on the changed high-level interaction course (cf. Fig. 2.3 for comparison). The demonstration data gathered during the *CONFIGURATION* stage is exploited for both inference of a redundancy mapping and constructing a navigation graph in task space. After definition of a goal in task space, the latter is utilized for planning a safe task space trajectory, and the latter to execute this according to the demonstrated environmental constraints.

or by means of other, more suitable interfaces [8].

These are passed to the path planner for generating safe task space trajectories through the free workspace. During *EXECUTION* a path is executed by mapping the planned task space positions to configuration space by means of the learned redundancy resolution $\mathbf{q}_c(\cdot)$ embedded in the hierarchical controller as described in Sect. 2.3.2.

As for the system's perspective on the *CONFIGURATION* stage, the interaction data is not only used to infer a generalized redundancy mapping $\mathbf{q}_c(\cdot)$ for the hierarchical control architecture. It is also exploited to construct a topological roadmap G estimating the reachable workspace for the robot. In fact, as the data is gathered through the physical human-robot-interaction, the corresponding Cartesian positions $\mathbf{x}_{\text{curr}}$ "experienced" by the robot represent free areas of the workspace. Note, that the terms free or save here not only refer to obstacle-free areas but generally to areas that intended or desired by the user for the robot to be used. As usual in path planning, the roadmap is implemented as a respective navigation graph G with nodes representing free locations, and the edges specifying allowed movements between them [133]. In this work G is an undirected graph. However, in contrast to standard methods such as rapidly-exploring random trees (RRT, [134]) or probabilistic roadmaps [135], in this work the nodes of the graph are not located in the robot's configuration space but rather in task space. Information about the corresponding topology in configuration space are encoded in the learned redundancy mapping $\mathbf{q}_c(\cdot)$ but also in the connectivity of G.

[8] As the proposed approach is not evaluated in a larger user study, the implementation of proper interfaces for goal definition is not considered here. However, depending on the application at hand goal definition can be implemented straight forward using basic monoscopic vision algorithms e.g. by tracking simple markers placed by the user. Other possibilities include programming, memorizing and labeling goals by means of the *assisted gravity compensation* controller utilized in the *PROGRAMMING* stage which already serves as an intuitive interaction controller as evaluated in Sect. 5.2.

Concerning the input data to the algorithm, in contrast to learning the redundancy mapping $\hat{\mathbf{q}}_c(\cdot)$ all position data $(\mathbf{x}_{\text{curr}}, \mathbf{q}_{\text{curr}})$ of the robot during the *CONFIGURATION* stage is employed, not only the chunks $\mathcal{D}^{(k)} = (\mathbf{x}_l^{(k)}, \mathbf{q}_l^{(k)})_{l=1,\ldots,L^{(k)}}$ of the *RECORDING* phase (cf. Sect. 3.1). This allows the proposed algorithm to infer safe transitions between the demonstrated relevant areas. Construction of the navigation graph G is conducted online during the interaction as described in the further course of this chapter and is designed to serve life-long learning capabilities. That is, learning is not required to stop e.g. to avoid over-fitting or topological defects [136], and the resulting graph can be exploited for path planning any time.

7.2 Review: Instantaneous Topological Maps (ITM)

With respect to the interaction model used to gather input data during the *CONFIGURATION* stage, a number of requirements have to be met for constructing the navigation graph G. The data to be processed is unknown a priori, presented to the learner incrementally as a continuous data stream with no specific end of training, and contains highly correlated samples. With respect to applicability, randomizing or shuffling of the data is not desired as this prevents real-time capability and exploitation of the so-far learned graph for path planning at any time. Furthermore, the algorithm has to be able to adapt to arbitrary topologies since prior knowledge about the environmental constraints is not accessible. This topology has to be estimated from relatively sparse input data as the user guides the robot only to relevant parts of the workspace, but does not even try to sample all regions systematically. Therefore, data is clustered in some regions and distributed very inhomogeneously within these (cf. Fig. 7.2 and Fig. 7.3).

To the best of my knowledge, the only algorithm that meets these requirements to incrementally learn arbitrary topologies from correlated stimuli is the instantaneous topological map (ITM, [130]), which was proposed to model data typically produced by exploratory movements in robotics. As for the proposed path planning approach it serves as algorithmic basis which is augmented towards the requirements in this chapter. The ITM shares with the Growing Neural Gas algorithm (GNG) [137] the ability to learn practically any topology due to its adaptive and incrementally generated number of nodes and edges, but it can - unlike the GNG - deal with correlated input stimuli [130]. By construction, it is a local and online learning algorithm and does not require repeated presentation of input data in order to iteratively converge to a correct representation of the input data. Though its instantaneous mapping capability, it produces reliable results as soon as data is provided.

The ITM algorithm works on a set of nodes i, represented by corresponding weight vectors (or prototypes) $\omega_i \in \mathbb{R}^M$ in input space, and processes input stimuli $\xi \in \mathbb{R}^M$. Throughout this chapter I use $M = 3$ for the Cartesian task space or $M = 6$ when including position and orientation. The nodes are connected by a set

Algorithm 7.1 ITM algorithm

Require: $\xi \in \mathbb{R}^3$, $\eta \geq 0$, $e_{\max} > 0$

1) Matching
$n \leftarrow \arg\min_i D_\xi(\xi, \omega_i)$
$s \leftarrow \arg\min_{i, i \neq n} D_\xi(\xi, \omega_i)$

2) Adapt winner node
$\omega_n \leftarrow \omega_n + \eta \cdot (\xi - \omega_n)$

3) Creation/Deletion of edges
if n and s not connected **then**
  create new edge (n, s)
end if
for all $c \in \mathcal{N}(n)$ **do**
  if $(\omega_n - \omega_s) \cdot (\omega_c - \omega_s) < 0$ **then**
    remove edge (n, c)
  end if
end for
delete unconnected nodes

4) Creation/Deletion of nodes
if $(\omega_n - \xi) \cdot (\omega_s - \xi) > 0$ **and** $D_\xi(\xi, \omega_n) > e_{\max}$ **then**
  create new node r with stimulus $\omega_r = \xi$
  create new edge (r, n)
end if
if $D_\xi(\omega_n, \omega_s) < \frac{1}{2} e_{\max}$ **then**
  remove s
end if

of undirected edges, which define a local neighborhood $\mathcal{N}(i)$ for each node i. The algorithm starts with two connected nodes, for example the first two input stimuli. After that, for each stimulus $\xi \in \mathbb{R}^M$ a set of four rules is applied as detailed in Alg. 7.1:

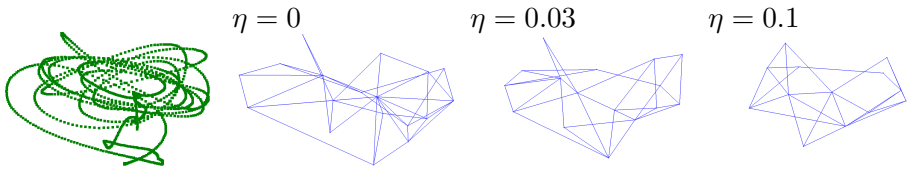
1. Matching of ξ to the nearest and second-nearest nodes n and s according to some distance metric $D_\xi(\cdot, \cdot)$.

2. Adaptation of the winner node towards ξ.

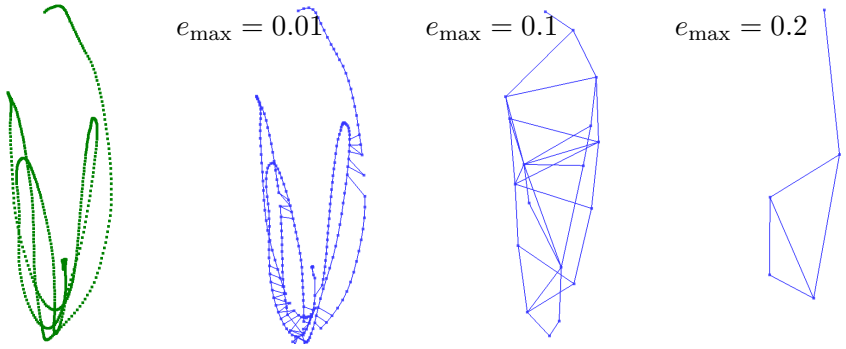
3. Creation and deletion of edges using a Thales sphere criterion.

4. Creation of nodes in unoccupied areas and deletion of dispensable ones.

The algorithm is based on two parameters, namely the adaptation rate η and the maximum quantization error $e_{\max}$, and a distance metric D_ξ in input space, which need to be specified. In this contribution, the Euclidean metric is used.



(a) Illustration for different η values.



(b) Illustration for different $e_{\max}$ values.

Fig. 7.2: Illustration of the influence of the ITM's learning parameters on the constructed navigation graph G for exemplary provided training data shown left. Note, that the topological defects [136] are only due that fact that the three-dimensional data is projected to two-dimensional space

The adaptation rate η is the ITM's equivalent to the learning rate of other topological networks, but here rather acts as smoothing parameter that slowly arranges the nodes to have approximately uniform distances [130]. In this work it is set to $\eta = 0$ for two reasons. First, the clear goal of the ITM learning is to use the constructed map for navigation, i.e. nodes and edges shall represent free task space. The adaptation would cause distortion of the nodes such that (for large η) or long learning it is no longer guaranteed that they are *free*. Second, as shown in Fig. 7.2(a) the nature of the input data gathered during *RECORDING* is typically circular and locally centered around the initially chosen $\mathbf{x}_{\text{fixed}}$ due to the utilized *compliant recording* controller π_{rec} (cf. Sect. 2.3.1 and Sect. 4.1). Adapting the nodes' position during processing of that data, would cause them to be *attracted* to the center $\mathbf{x}_{\text{fixed}}$ of a training area, thereby reducing the overall coverage of area by the learned graph significantly. Fig. 7.2(a) illustrates this effect by showing generated graphs for different η values on the right from the training data displayed on the left.

The second parameter, the maximum quantization error $e_{\max}$, determines the resolution or the density of the resulting topological map. Fig. 7.2(b) demonstrates the influence of this parameter. The resulting graph for $e_{\max}$=0.01 is constructed

along the trajectory of input stimuli with very few connections between the circular lines. This can be described as over-fitting. In contrast, $e_{\max} = 0.2$ is too large to create enough nodes to sufficiently learn the input area and therefore results in an under-fitting behavior.

7.3 Hybrid ITM with Connectivity Bootstrapping

In the following a number of enhancements and modifications are proposed to tailor the ITM towards practical use in the described human-robot interaction scenario.

7.3.1 Validation of Edges using a Joint Space Criterion

The ITM creates a navigation graph G in the input space (here task space) which can be used for navigation purposes. This is sufficient in cases where the robot performs only simple movements in unconstrained free space. But in context of redundant robots, planning in task space inherently is connected to the joint space. The manipulator's redundancy enables the robot to move in confined workspaces, i.e. grab around obstacles and navigate through narrow spaces. During the *CONFIGURATION* stage the user can fully exploit this feature, e.g. approaching the same workspace area in different configurations. As a result, pairs of task-space positions may occur during a teaching session, which the ITM considers as nearest neighbors, but which correspond to large distances in joint space. Fig. 7.3 illustrates this situation with a training session (training data on the left[9]) where the robot was moved twice or more to same areas but with different redundancy resolutions, e.g. behind the box obstacle on the right. The center figure shows two task-space positions referring to a close distance in task space but a large change in the robot's joint configuration. The plain ITM algorithm would connect these positions adding an edge to the navigation graph. But moving along these edges would cause the robot to change the redundancy resolution such that it collides with the obstacle.

This *hybrid* nature of the path planning problem is adressed by enhancing the ITM structure to a hybrid representation comprising both task space and joint space. Each node i of the hybrid ITM is represented by a corresponding weight vector $\omega_i \in \mathbb{R}^M$ in task space *and* a corresponding vector $\mathbf{q}_i \in \mathbb{R}^N$ in joint space. Validation of potential connections (edges) between these nodes can now be enhanced by means of an additional distance metric D_θ evaluating task-space edge candidates (n, s) in joint space using the corresponding joint values $\mathbf{q}_n$, $\mathbf{q}_s$ used by the robot (and the user) when approached n and s. In this work a simple threshold of $D_\theta(\mathbf{q}_n, \mathbf{q}_s) < d_{\theta\,\max}$ is used limiting the maximum distance in joint space for *valid* edges. Again, for D_θ the Euclidean metric is used. Finding a proper value for the threshold $d_{\theta\,\max}$ obviously depends on the specific robot

[9] The robot is positioned directly between the two box obstacles but is not visualized to prevent obscuring the data.

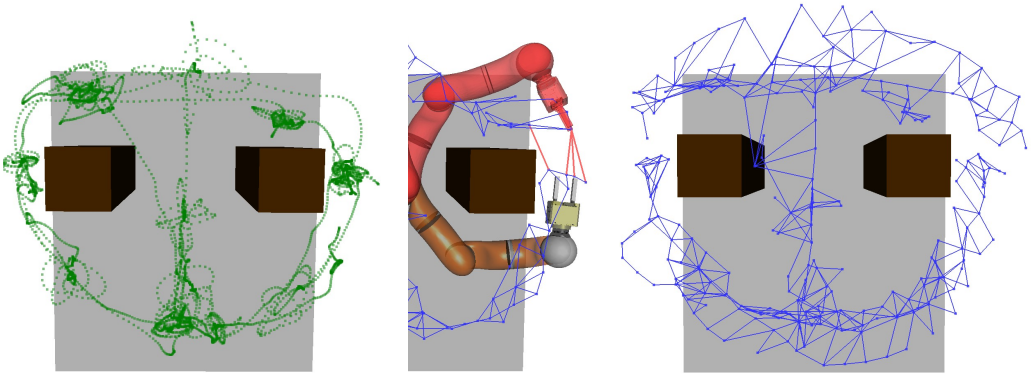


Fig. 7.3: Left: Recorded data from a training session during which the robot has been moved twice or more to same areas but with different redundancy resolutions. Center: Marked edge candidates between such areas that the standard ITM algorithm would connect but that can be detected as invalid using the joint space criterion. Right: The ITM learned from the training data with the joint space criterion included in the learning.

platform used (e.g. number of degrees of freedom, kinematic structure). During experiments suitable values were found between $d_{\theta\,\max} = 2.5$ and $d_{\theta\,\max} = 3.0$ for the KUKA Lightweight Robot IV. However, further experiments with other obstacles showed that the choice of $d_{\theta\,\max}$ seems to be rather scenario independent [131]. The necessary changes to the ITM algorithm are reflected in Alg. 7.2.

As shown in Fig. 7.3 (center), using the joint space criterion, edge candidates can be detected as invalid and are not added to the topological map. Fig. 7.3 (right) shows the learned graph from the training data shown on the left, which does not comprise any invalid connections.

7.3.2 Bootstrapping Data for Improved Connectivity

For the reasons discussed in Sect. 2.3.1, the data gathered through the *CONFIGURATION* phase *per design* is typically sparse in task space as depicted in Fig. 7.3 and Fig. 7.5. Learning a topological map of the free task space solely from this data results in a very sparse navigation graph G which is not suited well for navigation. The generated paths may be unnecessary long and detoured in many cases due to holes in the map and less connections between training areas. In graph theory, hole detection is a well known problem and several approaches such as [138] solve this problem. Detected holes can then be closed by creating edges between the corresponding nodes. However, these approaches are computationally expensive, require global processing and thus violate the prerequisite of an online-training capable method as discussed at the beginning of Sect. 7.2.

Therefore, the bootstrapping paradigm is utilized in the following to create

additional data samples from the set of acquired input data to be processed by the ITM algorithm. This allows to increase the connectivity in the learned topological map while preserving the *natural structure* of the ITM, in contrast to altering the graph structure a posteriori. The data pool is increased by two heuristics presented in the following.

Local Bootstrapping

The target of the local bootstrapping is to close small holes in the graph and increase local connectivity. During the ITM learning, for each newly created node r the local neighborhood

$$\mathcal{N}_\delta(r) = \{\text{node } c \,|\, \exists\, path(c \to r) \text{ of length } \leq \delta\}, \tag{7.1}$$

is utilized containing all nodes connected to node r through at most δ edges. For each $c \in \mathcal{N}_\delta(r)$, additional training samples are generated according to

$$\xi_c := \omega_c + \frac{1}{2} \cdot (\omega_r - \omega_c), \tag{7.2}$$

and the ITM is trained with them (omitting the bootstrapping steps). Fig. 7.4 illustrates this heuristic, showing the newly created node r and the local bootstrapping method for $\delta = 3$ creating new training samples to be processed by the ITM depicted as orange marks.

This introduces a new parameter δ of neighborhood depth. Note, that only values of $\delta > 1$ will make sense for this heuristic. The theoretical maximum value is defined by the longest path possible in the network. However, since computation time increases exponentially as a function of δ (assuming an approximately uniform valence for all nodes) a practical range of δ values is rather small. Furthermore, higher δ values will cause the topological map to converge to its convex hull and produce nodes far away from explored areas, which may be harmful in most scenarios. Since this heuristic is intended to enhance the data pool only locally, sample creation ξ_c far away from actual training data is prevented. The maximum distance for nodes $c \in \mathcal{N}_\delta(r)$ generating new sample stimuli ξ_c is limited to $D_\xi(r, c) < 3 \cdot e_{\max}$, which proved to be a suitable value, as this provides a span of exactly $e_{\max}$ for new nodes to be created.

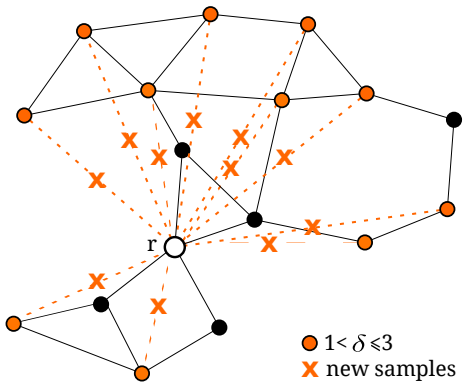


Fig. 7.4: Illustration of local bootstrapping heuristic.

Tab. 7.1 shows statistics about the results for a training session depicted in Fig. 7.5 (a) where the ITM was trained with $e_{\max}$=0.10. It shows the number of

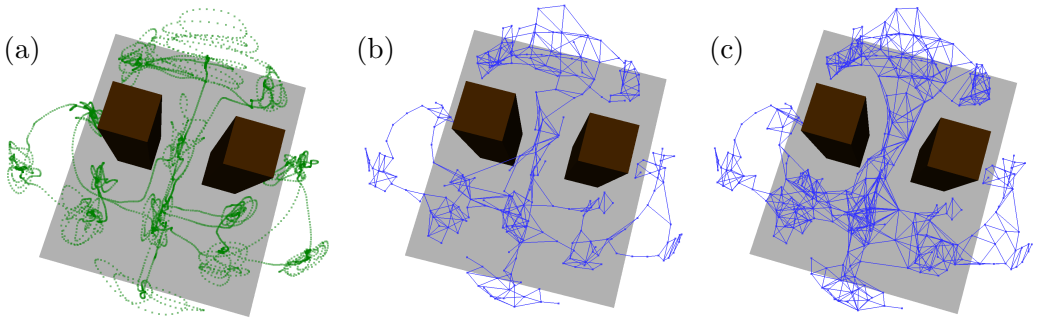


Fig. 7.5: (a) Training data recorded for evaluation of the local bootstrapping heuristic. (b,c) Generated navigation graphs G by ITM algorithm without and with local boostrapping ($\delta = 4$).

δ	Nodes	Edges	max ν	$\varnothing\,\nu$	max ϵ	$\varnothing\,\epsilon$
0	175	330	10	3.76	n.a.	n.a.
2	177	388	13	4.37	0.026	0.0208
3	191	509	14	5.32	0.053	0.0266
4	203	596	14	5.86	0.111	0.0399
5	211	653	15	6.18	0.128	0.0445

Tab. 7.1: Graph statistics of training with different δ values.

nodes and edges, the maximum and average valence ν of the constructed graph's nodes and an approximation of the error ϵ in meters caused by the heuristic. To measure this error, the minimum distance to any sample of the training data is calculated for each node created during the bootstrapping. From those values the maximum and average distances are calculated. Choosing δ=2 has practically no effect on node generation since bootstrapped samples are too close to existing nodes. However, there is a noticeable increase in the number of edges improving the local connectivity of the graph. For δ=3 the number of nodes increases by 9%, and the number of edges increases by 78%, improving the local connectivity significantly whilst the maximum and average estimated error induced by the heuristic stays acceptably small. I argue, that moving the robot in close vicinity of the obstacles of approx. 2 cm to 5 cm in human-robot interaction during the configuration is not desired and intuitively avoided by the teacher. Increasing δ results in further improved connectivity, as visualized in Fig. 7.5 (c) and (d), but comes with the cost of an increased maximum error of up to 13 cm which might be undesirable in some narrow setups. However, in the tested scenarios none of the chosen values for δ caused collisions between the robot and the environment.

Global Bootstrapping

This heuristic tackles the problem of weakly connected clusters, where the term clusters here loosely refers to the training areas chosen by the user during the kinesthetic teaching and covered by the constructed navigation graph G. Even if the distance between clusters is small, they are only connected if the robot has been moved between them during training. Sometimes this forces the path planner to traverse along other clusters to reach an adjacent one, even though a direct movement should be possible. This heuristic searches for potential cluster pairs and tries to connect them. In contrast to the local bootstrapping method which utilizes the local neighborhood $\mathcal{N}_\delta(r)$ of a node r as discussed above, for finding global connections the mere task-space distance between them is the main criterion to find possible connections.

During ITM learning, for each newly created node r a nearest neighbor search in task space selects all nodes

$$\mathcal{N}_{d_{C\,\max}}(r) := \{\text{node } c \,|\, D_\xi(r, c) < d_{C\,\max}\} \tag{7.3}$$

within a sphere with the radius $d_{C\,\max}$ around r. From this, all nodes of the δ-neighborhood $\mathcal{N}_\delta(r)$ are removed, since this heuristic tries to find *global* connections and not possible shortcuts in the *local* neighborhood. Finally - and for the same reasons discussed in Sect. 7.3.1 - the edge candidates between r and the remaining nodes $\mathcal{N}_{d_{C\,\max}}(r) \setminus \mathcal{N}_\delta(r)$ are validated using the introduced joint space criterion, i.e. for all valid nodes

$$c \in \mathcal{N}_{d_{C\,\max}}(r) \setminus \mathcal{N}_\delta(r) \quad \text{with} \quad D_\theta(\mathbf{q}_n, \mathbf{q}_s) < d_{\theta\,\max} \tag{7.4}$$

additional training samples according to Eq. (7.2) are generated.

An illustration of this heuristic is given in Fig. 7.6 where the ITM was trained again with the same data as shown in Fig. 7.5. The left figure shows the heuristic according to Eq. (7.4) used as a posteriori analysis marking a number of node pairs as valid to create new data samples at places where otherwise a larger movement along the graph is required. For example the training area at the very bottom is connected to the rest of the graph only by a single connection. The right figure shows the result when the heuristic is included during training. For each connection marked on the left a sample is generated and processed by the ITM to create new connections. Concerning the example of the bottom training, this adds the possibility to directly move to the training areas on either side. This experiment was done using a distance threshold of $d_{C\,\max} = 0.25$ which was found by means of a short analysis of different user's training data in [131]. As discussed there, this parameter is scenario specific relating to the constraints in the robot's workspace. For environments with few large obstacles a larger value can be used whereas for scenarios with small or thin obstacles a lower value is preferred.

The final algorithm of the ITM learning enhanced with the joint space criterion developed in Sect. 7.3.1 and the bootstrapping heuristics described in Sect. 7.3.2

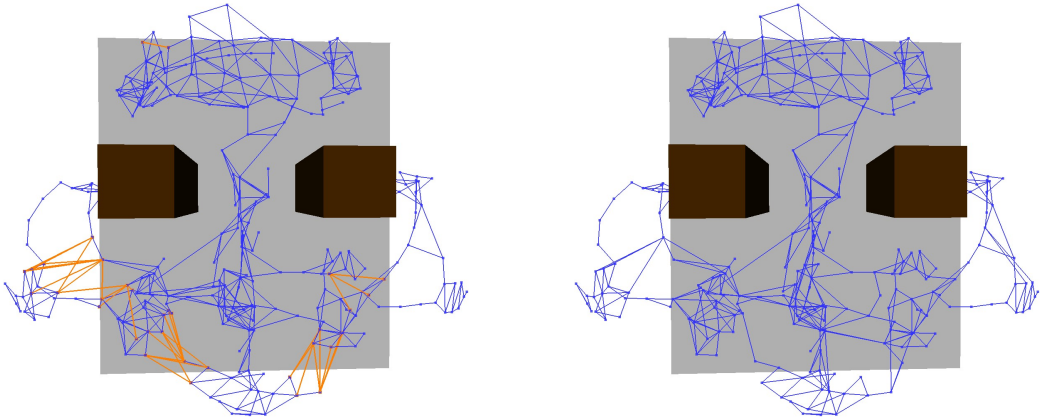


Fig. 7.6: Results of the global bootstrapping for $d_{C\,\max} = 0.25$ and $d_{\theta\,\max} = 2.5$. Left: A posteriori application of the global bootstrapping to a graph learned with the plain ITM marks compatible node pairs in orange. Right: Result of the topological map learning with the bootstrapping integrated in the learning.

is presented in Alg. 7.2. For the reasons discussed in Sect. 7.2 the adaption of the winner node (step 2) in Alg. 7.1) is omitted here. Also, removing of edges and nodes is omitted since these steps only have effect if adaption of the winner node is used.

7.4 Results

Given the current position of the robot $\mathbf{x}_{\text{curr}}$ and a goal $\mathbf{x}_{\text{goal}}$, the graph G generated by the enhanced ITM learning is used for path planning in task space. In this work the well known A* algorithm [139] is used, a state-of-the-art extension of the very popular Dijkstra algorithm [140] using a heuristic search while guaranteeing to still find the shortest path from the start to goal if it exists. However, in principle any algorithm capable of planning shortest paths in undirected graphs can be used. The result of the path planning algorithm is a sequence of nodes n_i with their task-space representations ω_{n_i}, which function as way-points in this work for a subsequently conducted spline interpolation. The generated, smooth task space trajectory is then executed by means of the *hierarchical control* concept utilizing the embedded learned redundancy mapping $\mathbf{q}_c(\cdot)$ as described in Sect. 2.3.2.

The presented method has been implemented and tested in simulation and on the real LWR IV using the software abstractions described in Chap. A. The presented topological map learning fulfills the real-time requirement without any problems: Using an Intel® Xeon® E5530 Quadcore CPU with 2.40GHz computation times below 1 ms on average for processing a single input sample are achieved in graphs of 300 nodes and more using $\delta = 4$.

Algorithm 7.2 Hybrid ITM with connectivity bootstrapping

Require: $\xi \in \mathbb{R}^M$, $\mathbf{q}_\xi \in \mathbb{R}^N$, $e_{\max} > 0$, $\delta > 1$, $d_{\theta\,\max} > 0$

1) Matching
$n \leftarrow \arg\min_i D_\xi(\xi, \omega_i)$
$s \leftarrow \arg\min_{i, i \neq n} D_\xi(\xi, \omega_i)$

2) Creation of new edge
if n and s not connected **and** $D_\theta(\mathbf{q}_n, \mathbf{q}_s) < d_{\theta\,\max}$ **then**
create new edge (n, s)
end if

if $(\omega_n - \xi) \cdot (\omega_s - \xi) > 0$ **and** $D_\xi(\xi, \omega_n) > e_{\max}$ **then**

3) and 4) Creation of new node and edge
create new node r with stimulus $\omega_r = \xi$ and $\mathbf{q}_r = \mathbf{q}_\xi$
create new edge (r, n)

5) Local Bootstrapping
for all $c \in \mathcal{N}_\delta(r)$ **do**
if $D_\xi(r, c) < 3 \cdot e_{\max}$ **then**
repeat 1) - 4) for $\xi = \omega_c + \frac{1}{2} \cdot (\omega_r - \omega_c)$ and $\mathbf{q}_\xi = \mathbf{q}_r$
end if
end for

6) Global Bootstrapping
for all $c \in \mathcal{N}_{d_{C\,\max}}(r) \setminus \mathcal{N}_\delta(r)$ **do**
if $D_\theta(\mathbf{q}_r, \mathbf{q}_c) < d_{\theta\,\max}$ **then**
repeat 1) - 4) for $\xi = \omega_c + \frac{1}{2} \cdot (\omega_r - \omega_c)$ and $\mathbf{q}_\xi = \mathbf{q}_r$
end if
end for
end if

Fig. 7.7 shows the result of the hybrid ITM algorithm trained with the interaction data during the *CONFIGURATION* stage as shown in Fig. 7.5. The visualized resulting navigation graph G demonstrates that the heuristics presented in this contribution greatly enhance the connectivity within the graph compared to the results of the plain ITM algorithm learning shown in Fig. 7.5. The local bootstrapping mainly enhances the connectivity within the training clusters whereas the global bootstrapping creates shortcuts between them drastically reducing the lengths of the resulting paths. Due to the introduced joint space criterion still all bootstrapped connections remain valid in terms of collision-freeness of the generated paths. The figure also displays examples of these planned and interpolated paths to random goal positions $\mathbf{x}_{\text{goal}}$ within that graph. Utilizing the embedded, learned redundancy mapping $\mathbf{q}_c(\cdot)$ the system transforms the resulting collision-

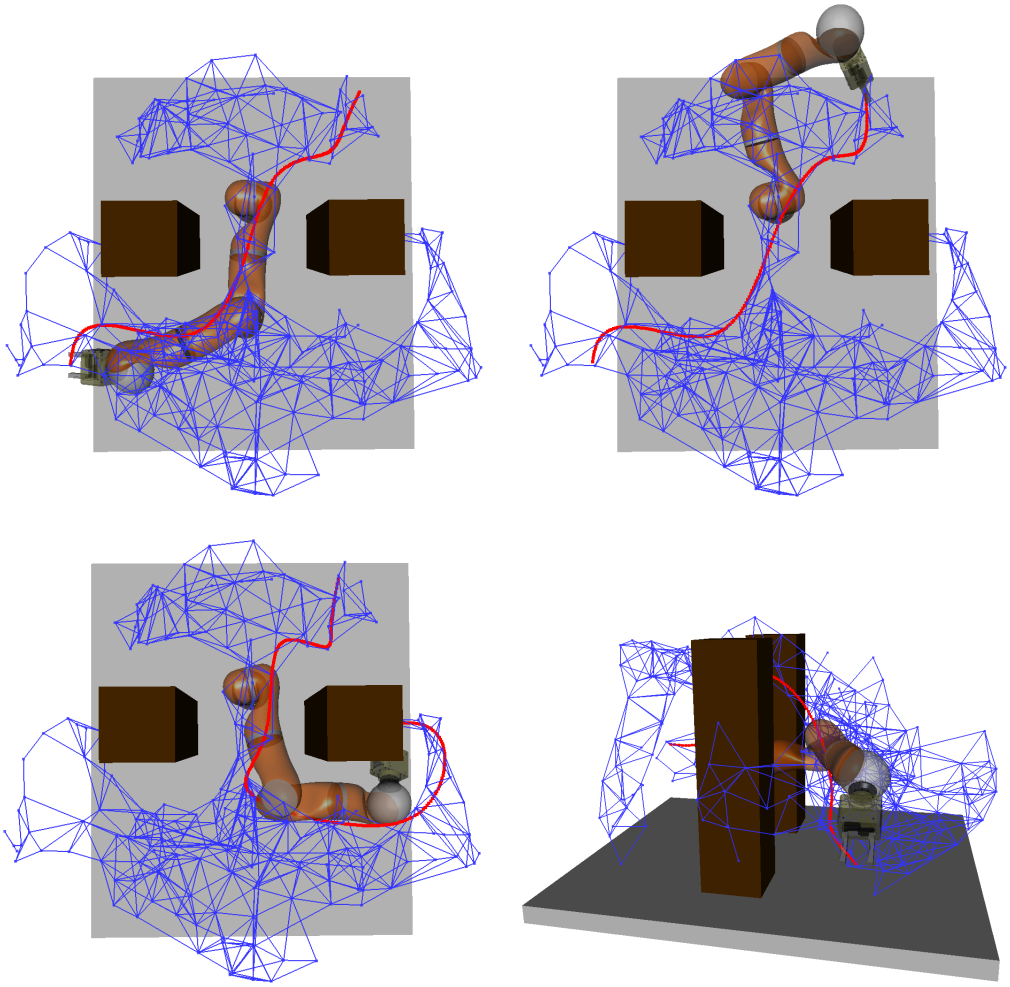


Fig. 7.7: The navigation graph G generated from the interaction data during the *CONFIGURATION* stage as shown in Fig. 7.5, and exemplary planned paths executed by means of the *hierarchical control* architecture embedding the learned redundancy mapping $\mathbf{q}_c(\cdot)$.

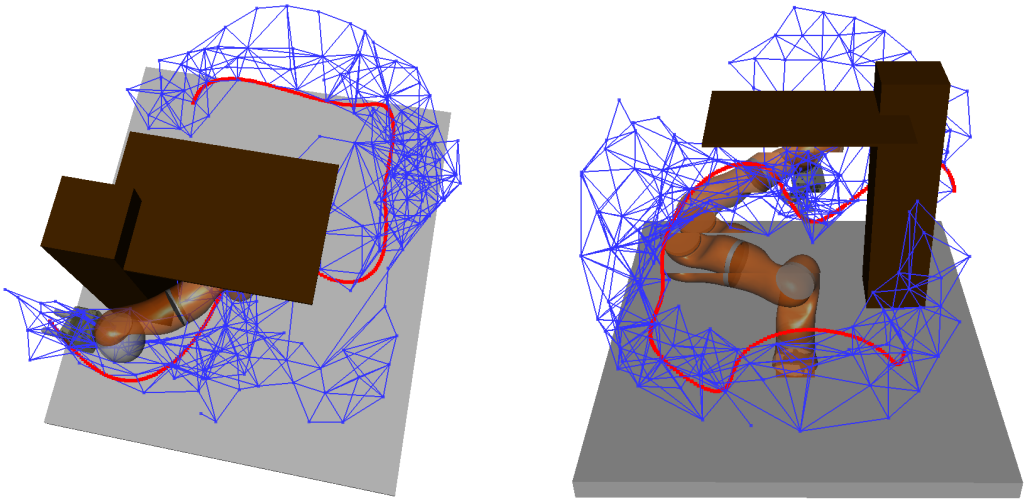


Fig. 7.8: Another illustrative result of the generated graph G and exemplary planned paths in a different obstacle setup.

free trajectory in task space into a collision-free joint space trajectory performed by the robot as indicated in Fig. 7.7. The combination of G and $\mathbf{q}_c(\cdot)$ enables the robot to autonomously navigate in the estimated free task-space without inducing any collisions with the static environment.

The example in Fig. 7.8 shows a different confined workspace. The training data was provided by a user in 10 different training areas within a few minutes. As can be seen, the generated graph already approximates the accessible workspace of the robot very well without introducing any explicit geometric description of the robot or its environmental constraints to the motion generation system. Again, the generated paths and corresponding joint-space motions are completely collision-free.

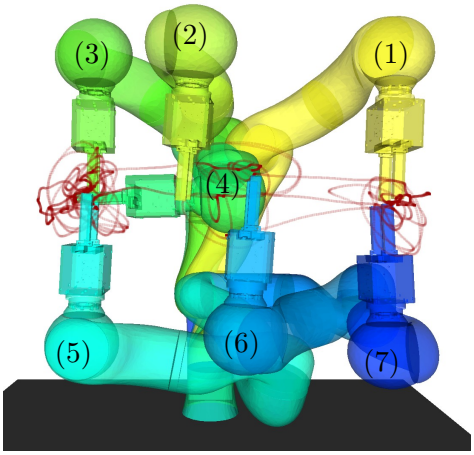
Extension to $\mathbb{R}^6$

Please note, that due to brevity and comprehensibility task-space planning in this work so far was reduced to $\mathbb{R}^3$. However, the proposed method also scales to the full six-dimensional task space and extending the presented ideas to incorporate also the rotational component of the task space is straightforward. Inputs to the hybrid ITM algorithm then relate to $\xi = (\xi_T, \xi_O)$, where the orientation ξ_O is represented with quaternions and ξ_T refers to translational coordinates. A new distance metric

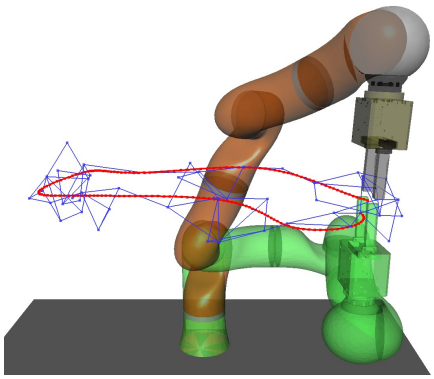
$$D_\xi(\xi, \tilde{\xi}) = (1 - \alpha)\ D_T(\xi_T, \tilde{\xi}_T) + \alpha\ D_O(\xi_O, \tilde{\xi}_O) \tag{7.5}$$

with a rotational component

$$D_O(\xi_O, \tilde{\xi}_O) = cos^{-1}(2 \cdot \langle \xi_O, \tilde{\xi}_O \rangle^2 - 1) \tag{7.6}$$



(a) Recorded training data and corresponding robot configuration in seven reelveant areas of the combined translational-rotational task space.



(b) Constructed navigation graph G and exemplary planned path from position (1) to same Cartesian position but with different end-effector orientation.

Fig. 7.9: Illustration of the hybrid ITM approach and path planning in combined translational-rotational task space $\mathbb{R}^6$.

computing the angle between two quaternions is utilized that mediates between translational and rotational space. D_T is the metric used in translational space and remains the Euclidean distance. In the experiments conducted in [131] a value of $\alpha = 0.1$ turned out to mediate well between both spaces while preserving the structural properties such as average edge length of the learned graph as in the translation-only case. Fig. 7.9 shows the result of the planning approach through safe regions in combined translational-orientational space in a proof-of-concept scenario. Seven relevant areas (1)-(7) have been selected to be demonstrated by the user during the *CONFIGURATION* stage, indicated by the recorded training data (red) and the corresponding configurations. The constructed navigation graph G represents the learned topology of the data. In order to test that topology, an exemplary path from the pose (ξ_T, ξ_O) referring to the training data in area (1) is planned to the same translational position ξ_T but with the orientation pointing downwards. Since from the training data a safe transition to the goal pose is only guaranteed by also changing the position, the path first proceeds to the left-hand side, where a safe transition between the orientations is guaranteed, and then back, instead of directly changing the orientation. This experiment indicates, that the implicit constraints provided by the user in task space are well reflected in the topology of the learned navigation graph.

7.5 Discussion and Related Work

This chapter presents a model-free path planning approach for redundant robots in confined spaces. It is purely data-driven based only on the robot's proprioceptive data gathered during the physical human-robot interaction in the *CONFIGURATION* stage. As this data is typically very sparse, specifically tailored heuristics are presented that bootstrap additional training data to increase connectivity of the resulting navigation graph. By this means, a topological map of the free, reachable workspace is constructed incrementally already during the interaction, that can be used at any-time to move autonomously within and between the relevant areas of the workspace. The idea behind that approach is taken from traditional planning and control for mobile robots that incrementally build up a world model when moving and interacting in the real world [141]. Similar to those approaches - and in contrast to traditional planning algorithms for rigid manipulators - planning is reduced to task space in this work and hence not subject to the curse of dimensionality when dealing with redundant robots since the number of degrees of freedom is irrelevant. This is possible by exploiting the learned redundancy mapping and by the presented dual internal representation of the topological map.

Please note, that for the sake of brevity the experiments conducted in this chapter rather serve as a proof of concept than a systematic evaluation of the proposed approach across different sets of learning parameters or different application scenarios. As for the former, I refer to a more decent evaluation in [131] that further elaborates on the introduced parameters e.g. with respect to average path length and curvature. Concerning applicability to other manipulators or more complicated scenarios, I refer to the experiments of chapter Chap. 8 where the developed path planning algorithm is used for a 9-Dof simulated manipulator in a more complicated scenario.

The proposed approach is hybrid in a sense that it connects research fields of learning from demonstrations, motion planning for redundant robots, and efficient path planning in task space. Other hybrid approaches such as the RRT-based planner proposed in [142] uses demonstrations to reproduce a path satisfying statistical constraints imposed by a human teacher while avoiding environmental obstacles. The same principles have been the basis for a framework called DGMP (demonstration-guided motion planning) using a clever combination of advanced techniques for probabilistic road maps and RRTs to reduce computation time as well as incorporating user demonstrations gathered from kinesthetic teaching [61]. However, both approaches plan in configuration space and therefore still suffer from high computational complexity. The method developed in [143] is based on dynamic wave expansion in task space and combines demonstrated trajectories with a full model of the environment. The method is easy to implement but plans only in task space for a mobile point robot, is not easily extendible to redundant manipulators and uses a model that is not available in our scenario. Further hybrid approaches such as the BiSpace planning method [144] combine planning in con-

figuration and task space by simultaneously exploring two RRTs, one in workspace and one in configuration space and thereby improving the planning performance for complex high-dimensional spaces. Behnisch *et al.* [145] reduce computation efforts by shifting the planning problem to a high-level representation in task space. The approach is divided into two steps: global planning using RRTs in task space and local, distance-based obstacle avoidance utilizing potential functions in configuration space. Still, all of the approaches require a model of the robot and the environment in order plan a collision-free path which are not available in the scenario discussed in this thesis.

In contrast, the proposed approach is completely model-free and allows the user to transfer his/her implicit knowledge about environmental constraints to the robot rather than programming them explicitly. They are encoded in both the learned redundancy mapping and the topology of the constructed navigation graph as the experiment in Fig. 7.9 particularly demonstrated. Since for this purpose the same data is utilized, that is already gathered through the *CONFIGURATION* stage, no additional effort for the tutor or change to the already validated interaction scheme is necessary. The added functionality is transparent to the user. In addition, the proposed approach supports online and life-long learning. It is already included during the *CONFIGURATION* and can serve as feedback to the user which areas are already covered by the graph and which remain to be taught. As long as the environmental constraints do not change, the map of the free task space is valid and can be extended as well as exploited any time for planning collision-free, autonomous motions for the robot.

Therefore, the presented method contributes to the overarching goal of the thesis and proves the hypothesis raised at the beginning of this chapter. By means of intuitive interaction interfaces the robot is instructed incrementally according to the user's knowledge and needs, in order to increase it's degree of autonomy from purely human-operated to a more self-reliant system reducing the interaction effort on the user's side. Only by learning from the demonstration data provided in the *CONFIGURATION* phase the robot is able to safely and autonomously move between relevant areas of its workspace in presence of environmental constraints.

Chapter 8

Incremental Teaching of a Simulated 9-DoF manipulator

Throughout this thesis, the derived concepts and structured user interaction for incremental teaching and efficient exploitation of implicit constraints have been implemented and tested decently on the 7-DoF KUKA Lightweight Robot IV. Although the presented implementation was rather straightforward utilizing the LWR IV's compliance features, I hypothesize that the former does not depend on the latter. As already briefly discussed in Sect. 5.1, the required interaction controllers π_q, π_{rec} and π_x can be implemented according to the offered sensors, interfaces and low-level controllers of the used robot platform and not impose specific requirements on these. Therefore, I hypothesize that

> *the proposed concepts of incremental teaching and learning of environmental constraints scale to both manipulators with more than seven DoF and across different implementations of the interaction model.*

As a result, the approach in this work can be applied to a variety of robots and applications.

However, a thorough evaluation of this hypothesis would require to systematically implement the required interaction structure and controllers on several, structurally different robot platforms each tested in user studies with non-experts. This is out of the scope of this thesis. In contrast, I approach this question in a proof-of-concept paradigm with an overly exaggerated example.

The experiment conducted in this chapter is therefore based on a *simulated manipulator* with *nine degrees of freedom* that is taught by a user in a *complicated, confined workspace* only by *tele-operation* via a *remote control keyboard and a computer screen*. This tests the scalability of the proposed approach in three important aspects. First, as discussed in Sect. 2.2 the interaction with a simulated robot via tele-operation from the human perspective constitutes a completely *different interaction interface* rendering the human-robot interaction less direct and less intuitive. Therefore, the required interaction controllers for the incremental teaching procedure must be implementable on a plain keyboard-based remote controller. Second, as the dimensionality of the configuration space increases but the task space is kept to be three dimensional the overall redundancy of the manipulator significantly increases to comprise a six dimensional null-space. Hence,

the scenario requires *enhanced generalization capabilities* of the utilized learning algorithm to still learn a valid redundancy mapping from on only few demonstrations. Particularly with regard to the complicated interaction interface, the number of required training areas should not increase drastically to not deteriorate the efficiency of the *CONFIGURATION*. Last, teaching redundancy resolutions is conducted in a challenging workspace comprising narrow gaps and other strong environmental constraints imposing *higher demands in terms of accuracy* on the learned redundancy resolution and the path planner.

In the following, I describe the implementation of the interaction model for the simulated arm according to Sect. 2.3 and briefly outline the experimental setup. Subsequently, the results of this experiment are presented, ensued by a short discussion of these.

8.1 Interaction Model

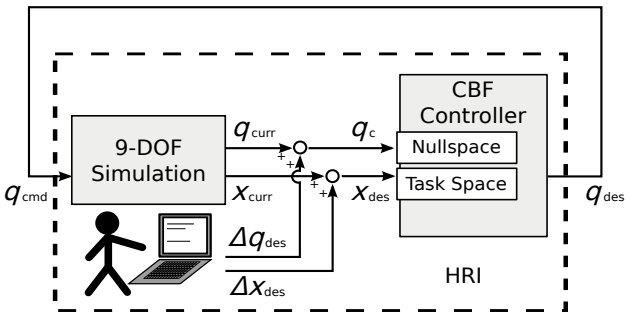
The implementation of the interaction model for this simulated robot arm again follows the workflow of the *CONFIGURATION* as outlined in Sect. 2.3.1 (Fig. 2.5).

However, a central difference between the chapter at hand and the previous ones is that the user cannot physically interact with the robot. The robot is simulated by a simple visualization on a computer screen as shown in Fig. 8.2(b). Interaction between the user and simulation is realized by simple key bindings on a remote control keyboard. Moving the robot's joints and/or the end-effector is implemented by means of (+/-)-buttons for each dimension respectively. The commanded joint values $\mathbf{q}_{\text{cmd}}$ are smoothed by the simulation by means of a second order filter clamping maximal velocity and acceleration, and checked for not exceeding the joint limits of the manipulator. Despite that, no other low-level control or interface is implemented.

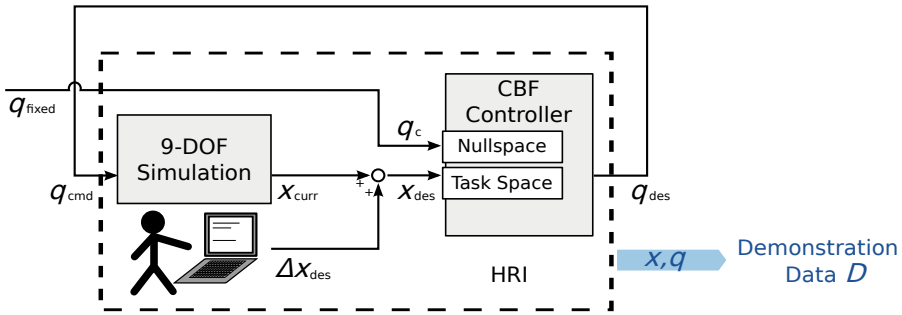
In fact, throughout the entire experiment the keyboard is the only input interface for the user, including guiding the robot to relevant workspace, recording training data for the learner, and for configuring the redundancy resolutions. Hence, also the previously described interaction triggers `on_affected` and `on_converged` (Sect. 2.3) to switch between the *APPROACHING* and the *RECORDING* stage are realized by simple key bindings.

Obviously, operating the nine degrees of freedom purely with this very basic interface, where each desired change of the robot's configuration would require to use the respective buttons either simultaneously or one after another, is not suitable. Therefore, already the required interaction controllers π_q and π_{rec} for the *CONFIGURATION* are implemented by means of a null-space controller as described in the following. They are designed to mimic the interaction with a real robot where users typically mostly move the robot's end-effector and adapt the joint configuration only temporally in presence of environmental constraints.

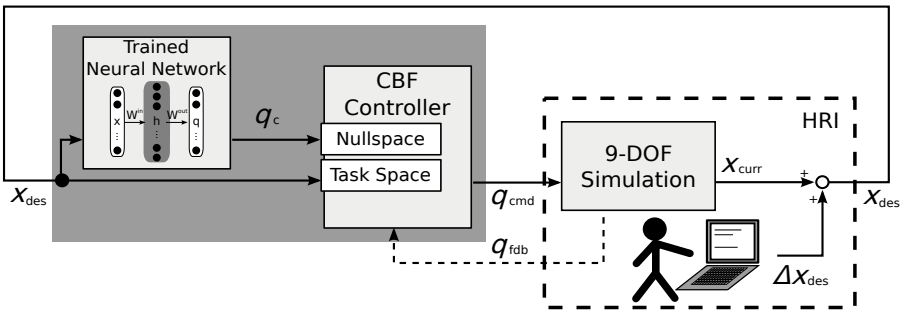
The required *gravity compensation* controller π_q is implemented based on the null-space controller as shown in Fig. 8.1(a). Via the remote control keyboard the



(a) *gravity compensation* controller π_q



(b) *compliant recording* controller π_{rec}



(c) *assisted gravity compensation*

Fig. 8.1: Implementation of the required interaction controllers for the simulated, tele-operated 9-DoF arm according to the concepts described in Sect. 2.3. Human-robot interaction is realized by means of a remote control keyboard (teleoperation).

user requests desired changes $\Delta\mathbf{x}_{\text{des}}$ in the robot's end-effector position or adapts the currently selected redundancy resolution by $\Delta\mathbf{q}_{\text{des}}$. The resulting commands in task space $\mathbf{x}_{\text{des}}$ and for the redundancy resolution $\mathbf{q}_c$ are then fused by the null-space controller to generate a desired joint configuration $\mathbf{q}_{\text{des}}$ which is set as command for the simulated robot. This interaction scheme allows the user during the *APPROACHING* stage to control the robot's end-effector and adapt the selected redundancy resolution on demand in order to guide it to relevant areas of the workspace with a desired initial joint configuration $\mathbf{q}_{\text{fixed}}$ while complying to constraints in the robot's workspace.

For the *RECORDING* stage, the required *compliant recording* controller π_{rec} is implemented similarly, but with the difference that the fixed joint configuration $\mathbf{q}_{\text{fixed}}^{(k)}$ is set as temporally fixed null-space constraint to the hierarchical controller, see Fig. 8.1(b). Training data $\mathcal{D}^{(k)} = (\mathbf{x}_l^{(k)}, \mathbf{q}_l^{(k)})_{l=1,\ldots,L^{(k)}}$ with $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{q} \in \mathbb{R}^9$ is recorded by the user through moving the robot's end-effector in that local area via the remote control keyboard. This control scheme can be interpreted as setting a temporal home posture for the simulated manipulator. Examples of such training data are shown in Fig. 8.3 in $K = 8$ training areas.

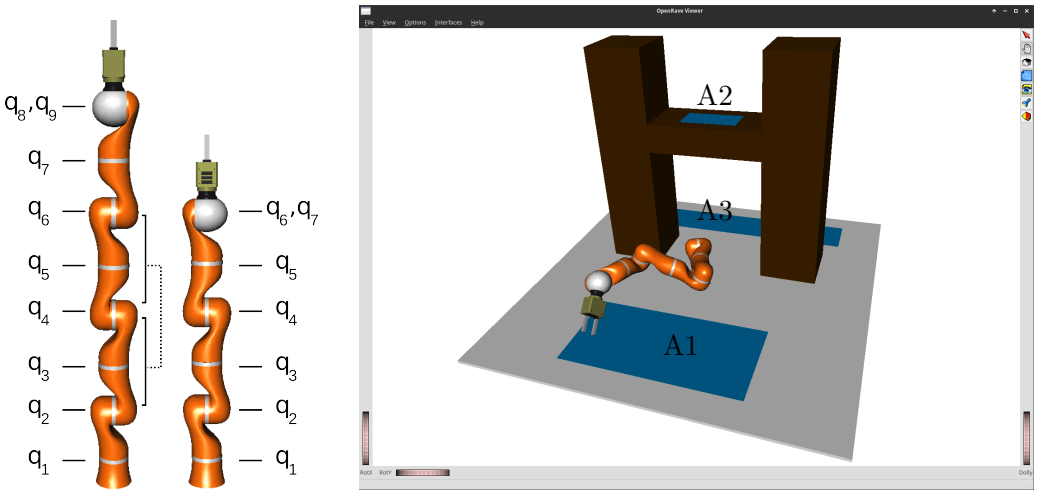
Learning the redundancy mapping $\mathbf{q}_c(\cdot) \equiv \hat{\mathbf{y}}(\cdot, \omega)$ from the demonstration data is conducted online with the OS-ELM algorithm [116] as discussed in Sect. 6.2. This allows to embed and test the so-far learned redundancy resolution $\mathbf{q}_c(\cdot)$ by means of the *assisted gravity compensation* controller already during the *CONFIGURATION* stage directly after each *RECORDING* phase, similar to the interaction model in Chap. 6. Note, that despite increasing the dimensionality of the learners output layer to $\mathbf{y} \in \mathbb{R}^9$ no other changes in the implementation of the learning component is required. In particular, the utilized hyper parameters such as ε or R are exactly the same as for the setup discussed in the previous chapters. The same holds for the graph learning component described in Chap. 7

As for the *assisted gravity compensation* controller, the implementation is illustrated in Fig. 8.1(c). The user operates the robot's end-effector through commanding desired Cartesian displacements $\Delta\mathbf{x}_{\text{des}}$ of the current positions $\mathbf{x}_{\text{curr}}$ by the remote control keyboard. Hence, the required task-space controller π_x (cf. Sect. 2.3.3) is implemented via the keyboard-based tele-operation.

8.2 Experimental Setup

Throughout this experiment an augmented simulation model of the LWR IV is used as shown in Fig. 8.2(a). It is created by simply duplicating q_3 and q_4 of the LWR IV's original simulation model together with the associated links and inserting them between the joints q_4 and q_5. As a result, the simulated manipulator comprises nine degrees of freedom.

The confined workspace used in this experiment is shown in Fig. 8.2(b) comprising three relevant working areas indicated as blue faces and the manipulator



(a) (b)

Fig. 8.2: Experimental setup used in this chapter: (a) Illustration of the simulated 9-DoF manipulator in comparison to the simulation model of the original KUKA Lightweight Robot IV. The duplicated links and joints are indicated. (b) Simulation environment and confined workspace consisting of different obstacles and three relevant working areas A1, A2 and A3 with the robot being mounted at the center of the plane.

mounted in the center. For this study, the latter is supposed to work within these areas and navigate between them autonomously. The large area displayed in the front (A1) is freely accessible. Since no constraining obstacles are placed beyond, many different redundancy resolutions are suitable to work in this area. For reaching a second area on top of the obstacle (A2) the robot must enfold itself very much, requiring the user to teach a very specific redundancy resolution. The third area (A3) is challenging in two concerns. First, only a very narrow space can be used to navigate from other areas to this one requiring the robot to fold itself in order to fit through the passage. Second, within this working area the lower part of the robot's body must stay in a certain narrow region to not collide with the obstacles, while the end-effector is required to move freely from left to right in that area behind the obstacles.

During the experiment, the implemented interaction controllers from the previous section are used to demonstrate redundancy resolutions in selected parts of the workspace according the proposed *CONFIGURATION* procedure. According to the implemented interaction model, after each training area the currently learned redundancy resolution is tested manually in interaction by means of the *assisted gravity compensation* controller and additional training areas are selected on demand. Learning the navigation graph is conducted throughout the entire

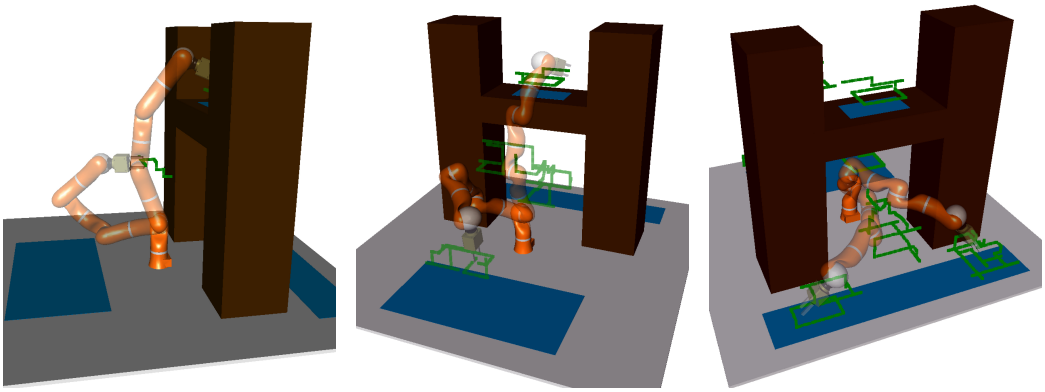


Fig. 8.3: Illustration of the redundancy resolutions for some of the $K = 8$ training areas recorded during the *CONFIGURATION* stage for the simulated 9-DoF manipulator. The recorded task space positions $\mathbf{x}_l^{(k)}$ are plotted as green dots and the selected initial joint configurations $\mathbf{q}_{\text{fixed}}^{(k)}$ are visualized with a transparent robot, respectively.

CONFIGURATION stage.

After enough training areas are demonstrated, the *CONFIGURATION* phase is quit and the resulting *hierarchical control* mode embedding the final redundancy resolution is evaluated together with the constructed navigation graph G.

8.3 Results

In the following, I present the results of this single experiment. During the *CONFIGURATION* stage training data has been recorded in $K = 8$ areas as illustrated in Fig. 8.3. For the sake of brevity not all 8 selected redundancy resolutions are displayed. They have been selected to either enable the robot to work in a specific area or to show how to move between areas. In order to make area A1 accessible two training areas are demonstrated, as well as two more for area A3 and only one needed for A2. These training data has been demonstrated to make the entire respective working area accessible to the robot. The remaining three training areas are selected to demonstrate intermediate solutions that show the robot how to move in between the working areas, e.g. how to fold through the passage under area A2 in order to get to area A3. During the *CONFIGURATION* stage no self-collisions or collisions between simulated robot and obstacles occurred. The entire *CONFIGURATION* took around 30 minutes, which sounds a lot at first glance. However, one has to account for the increased complexity of the indirect interaction.

The feasibility is indicated by the results shown in Fig. 8.4 and Fig. 8.5 which displays the resulting navigation graph, exemplary planned paths between ran-

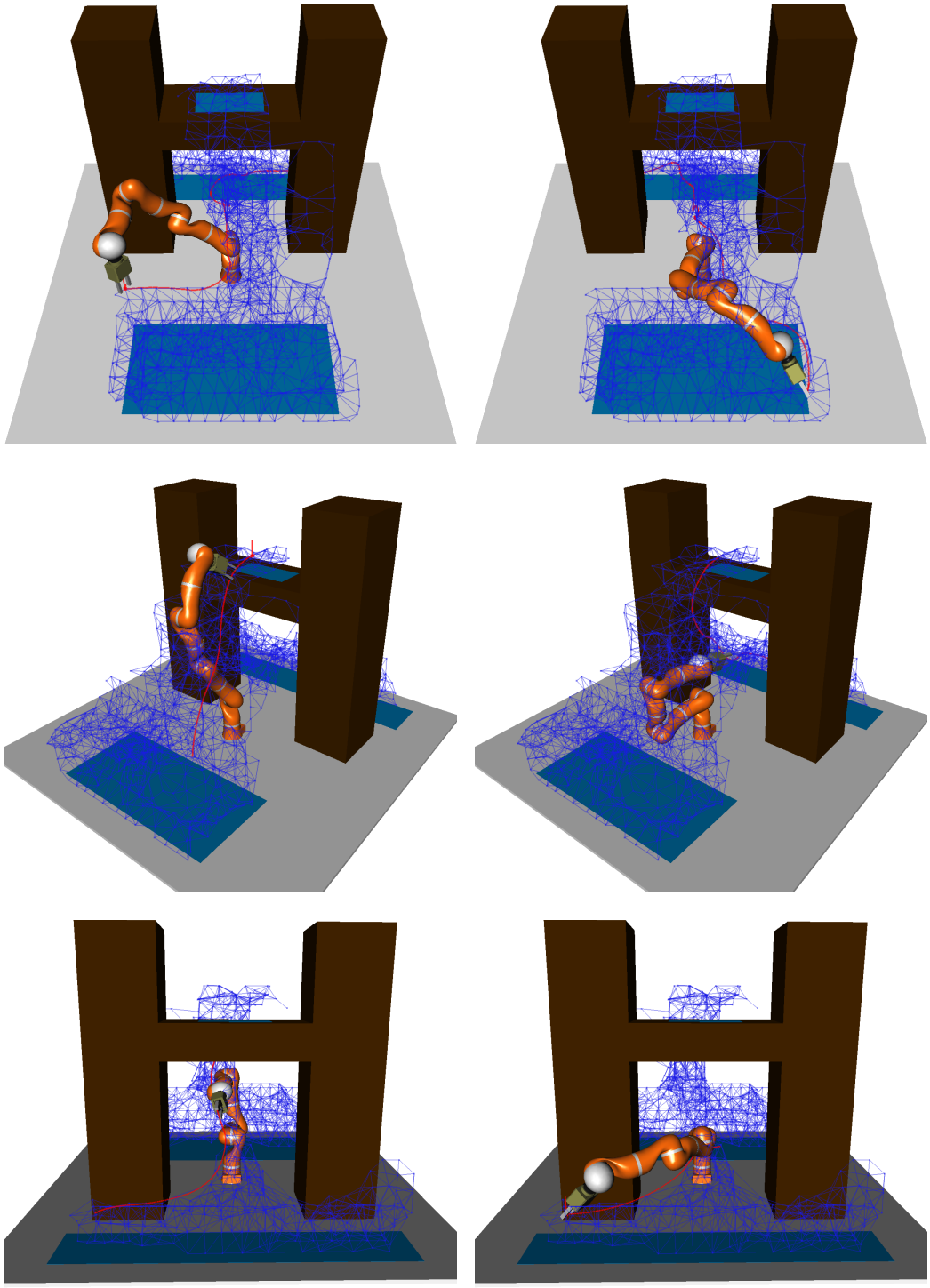


Fig. 8.4: Resulting navigation graph and exemplary planned paths after teaching the simulated 9-DoF arm.

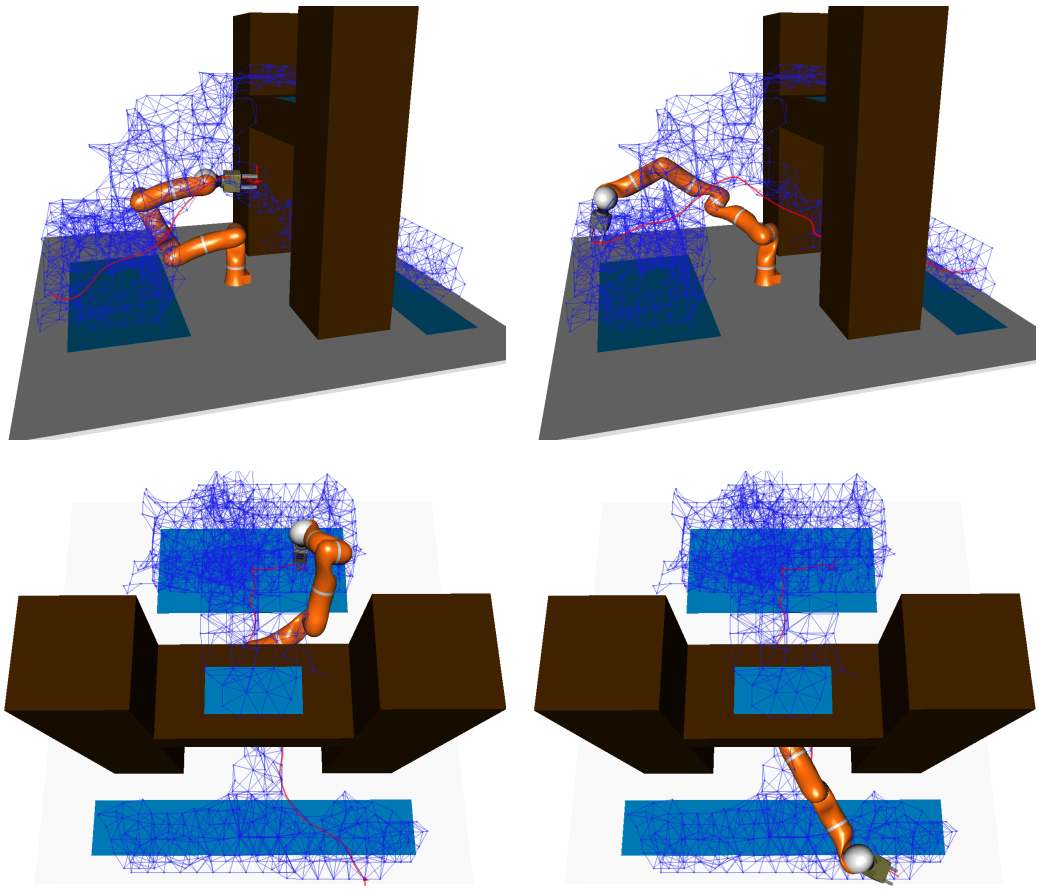


Fig. 8.5: More examples of planned paths after teaching the simulated 9-DoF arm.

dom positions and robot configurations along these paths. As indicated by the plots, all working areas are accessible for the robot as the navigation graph covers these. The provided demonstration data of the *CONFIGURATION* stage now enables the robot to navigate within but also between the different working areas completely autonomously. The demonstrated redundancy resolutions are generalized by the learner over the entire (relevant) workspace in order to comply to the strong constrains imposed by the confined scenario. As a result, the planned paths are completely collision-free as are the generalized redundancy resolutions along the paths. Noteably, the navigation graph is constructed not solely based on the training data shown in Fig. 8.3. As outlined in Sect. 8.1 the redundancy resolution is learned online during *RECORDING* phase and then, during *APPROACHING*, temporally and locally tested using the *assisted gravity compensation* controller. While during this latter stage no training data for the neural learner is recorded, the hybrid ITM still processes position data $\mathbf{x}$ and $\mathbf{q}$ of the robot as inputs to construct the navigation graph.

In order to support this visual inspection with a short numerical evaluation, 100 nodes are selected randomly from the navigation graph and set as goal positions $\mathbf{x}_{\text{goal}}$ for the path planner. The calculated task-space paths are then executed in the simulation by means of the hierarchical controller. During execution, the simulation software checks for collisions. Once a path is executed with a final position $\mathbf{x}_{\text{end}}$, it is checked whether the desired goal position $\mathbf{x}_{\text{goal}}$ actually is reached. The threshold for an acceptable accuracy is set to $E_{\text{task}} = \|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{end}}\| < 1\,\text{cm}$.

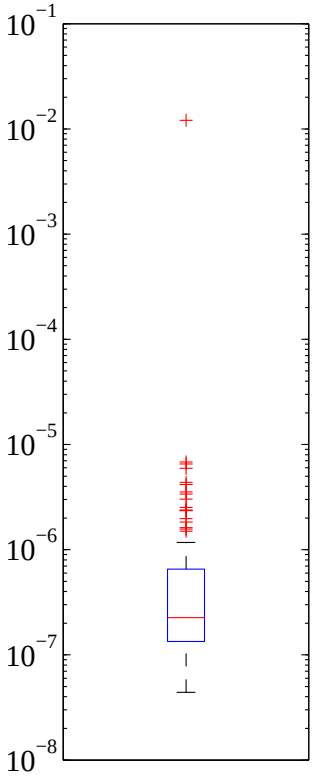


Fig. 8.6: Accuracy E_{task} at end points of planned paths.

The result of this analysis is that all 100 paths could be executed collision-free. Hence, both path planning and redundancy resolution successfully worked together to avoid getting into contact with the simulated obstacles. While the former generates a collision-free task-space trajectory through the obstacle-free workspace from start to goal, the latter transforms this to a collision-free trajectory in joint space by means of the taught and generalized redundancy mapping $\mathbf{q}_c(\cdot)$. Concerning accuracy at the end-point $\mathbf{x}_{\text{goal}}$, I report that only one of the 100 targeted goal positions could not be reached with the desired accuracy. While all other targets are reached with an accuracy far below 1 mm (cf. Fig. 8.6), the error for that single case constitutes 1.2 cm. Manual inspection of that single exception reveals that the redundancy resolution for that target point was generalized slightly beyond the joint limits of the simulated manipulator. Thus, for the same reasons as discussed in Sect. 4.3.4 task space accuracy decreases as the implemented control architecture rejects joint commands exceeding the limits.

8.4 Discussion

The presented results confirm the hypothesis of this chapter that the proposed concepts for incremental teaching of redundancy resolutions generalize in certain dimensions.

The implemented interaction model relies only on pure keyboard-based tele-operation of a simulated robot. Clearly, even for tele-operation of simulated robots there exists much more suitable input interfaces than tested in this experiment, such as 3-D input devices ("space mouse") typically used for CAD modeling. In [146] the authors used a touch screen displaying the simulated robot for adapting its posture according to taps on that screen, which can be regarded as an interme-

diate solution between physical interaction and pure tele-operation with joystick-like devices. As another example, the authors in [50] report about experiments for teleoperation of a humanoid robot's arm showing that their keyframe-based approach enhances task-accuracy. However, the goal of this chapter was rather to demonstrate that the proposed concepts of this thesis do not depend on a specific form of advanced compliance features e.g. high-frequent, precise force-/torque sensors. Exaggerating this hypothesis, the concepts could be implemented even for the present simulated robot model without any interface for physical human-robot interaction.

As a result, it is not surprising that teaching redundancy resolutions becomes much more complicated and longish in the presented analysis. The required interaction includes moving the robot's end-effector step-by-step, changing the redundancy resolution on demand, switching between *APPROACHING* and *RECORDING* as well as changing the camera position in the simulation environment. Keeping that in mind, a teaching time of 30 minutes for the entire *CONFIGURATION* stage is an acceptable value, particularly as in the subsequent stages interaction complexity reduces. Instead of controlling nine joints and three task dimensions, the *assisted gravity compensation* controller simplifies interaction drastically to three dimensions. In addition, the knowledge conveyed to the robot during the *CONFIGURATION* stage about accessible, obstacle-free regions reduces the necessity of interaction even further. The utilized graph learning algorithm enables the system to act semi-autonomously by means of simple goal-directed commands, if the application at hand permits.

Although the presented analysis lacks a decent statistical evaluation, the results also indicate the robustness of the utilized machine learning algorithms. Without changing anything but adapting the implemented learners according to the increased dimensionality, both the hybrid ITM and the online sequential ELM were able to efficiently generalize the taught constraints. No tuning of hyper parameter such as ε, R or $e_{\max}$, $d_{\theta\,\max}$ was required demonstrating the scalability of the approach concerning higher-dimensional configuration spaces.

In fact, despite also adapting the null-space controller component according to the two additional DoF, literally no other changes to the implemented control architecture were needed. This demonstrates the robustness of utilized software abstractions to implement the proposed interaction concepts in the FlexIRob prototype also from a system integrator's perspective. However, as proper software engineering in the context of this thesis is regarded rather as a required tool than subject to investigation, this direction is not further evaluated. Again, interested readers may therefore refer to Chap. A for further details about the system integration of the utilized hardware and software components.

I conclude this chapter with a short disclaiming note. As the presented experiment followed rather the paradigm of a proof-of-concept, no decent study with non-expert users was conducted. Instead, I taught the system by myself in a single-experiment design. In order to get used to the increased dimensionality of

the manipulator and complexity of the interaction during the *CONFIGURATION* stage, the *experiment was repeated three times*. While the first trial consisted in simply testing the interaction in the confined workspace, the second trial also included a shorter evaluation of the trained system with 50 randomly selected goal positions for the path planner. In that trial, only two of the 50 planned and executed motions entailed collisions. The results presented in here are obtained in the third trial. Therefore, of course the proposed implementation of the interaction model does not work "out-of-the-box" for every user. On the opposite, I emphasize that the experiment was not conducted several dozens of times, in the end reporting only a single successful run. I argue, that the first two trials can be regarded as a warm-up stage, similar to those we utilized in the FlexIRob@Harting study (cf. Chap. B) to get the participants used to the interaction and kinematics of the robot. After that warm-up, I was able to conduct the *CONFIGURATION* repeatedly successfully.

Chapter 9

Conclusion

In this thesis, I proposed an incremental kinesthetic teaching procedure that aims at enabling non-expert users intuitive teaching of redundant manipulators. As argued and demonstrated throughout this work, the typical problem, users have to face when interacting with such high-dimensional robots, is the simultaneous consideration of task constraints and (task-independent) environmental constraints. Interaction strategies solely based on the demonstrating-in-configuration-space paradigm are not feasible for an efficient and durable transfer of the tutor's implicit knowledge of these constraints. On the other hand, demonstrating-in-task-space strategies are easier to use, but mostly rely on the definition of explicit constraints e.g. for redundancy resolution. Concerning contemporary demands in industrial manufacturing, particularly with respect to small and medium-sized enterprises, I argue that the lack of intuitive and flexible human-robot interfaces could limit the applicability of modern compliant and dexterous robot platforms.

I addressed this issue in Sect. 2.3 with an approach that combines the advantages from both paradigms. A human-robot interaction model is designed to successively reduce the teaching complexity by shifting interaction effort from the user's side to the robot. However, this is not achieved by introducing explicit criteria or world knowledge in terms of geometric models, but rather by incrementally transferring the user's knowledge about environmental constraints to the robot system. As a key concept for this, I introduced the idea of efficient teaching and exploitation of redundancy resolutions. By means of a dedicated, task-independent *CONFIGURATION* stage the user is enabled to provide demonstrations for these constraints in the robot's workspace. Subsequently, the robot can exploit the conveyed information during a *PROGRAMMING* phase where the proposed *assisted gravity compensation* controller reduces the tutor's interaction effort by assisting in the selection of valid redundancy resolution.

As for learning these constraints, I proposed to rely on a pragmatic viewpoint on the problem of redundancy selection. In typical application scenarios, it is often sufficient to learn only a *feasible* solution instead of searching for the *optimal* one. Therefore, the interaction model for the *CONFIGURATION* is designed to teach and learn only a fixed, valid redundancy resolution across the workspace by providing only few demonstrations. As this might impose challenges from a machine learning viewpoint, Chap. 3 systematically evaluates neural-network-based approaches with respect to their applicability in this context. Mainly two

approaches, the standard ELM network [112] and a specifically tailored LLM variant (cf. Sect. 3.2.2), are analyzed thoroughly with respect to their generalization abilities and robustness against varying model selection parameters and varying training data distributions. Both reveal good results in generalizing the taught constraints even beyond the selected training areas. Despite the comparison, the chapter shows the practicability of the approach. In confined workspaces only few training areas are required to make large portions of the workspace accessible to the robot. Please note, that also other learning algorithms such LWPR, GMR, GPR, RBFN (cf. [147] for an interesting, very recent overview) can be employed. However, since they rely on the same underlying principles as the evaluated methods [147], I do not expect qualitatively different results.

In the next step, I analyzed the *CONFIGURATION* stage from the user's viewpoint in Chap. 4. As this interaction stage is designed to transfer implicit knowledge from the user to the robot, I investigated the implicitly, taught constraints and related them to explicit constraints in the robot's workspace (physical obstacles). However, this relation is not always given, e.g. when a user adapts the robot only according to personal needs but not related to physical constraints. Subsequently, I analyzed the interaction scheme with results obtained in the FlexIRob@Harting study. They reveal, that teaching redundancy resolutions is feasible even for non-experts, and that the general interaction experience with the FlexIRob system (including interaction controllers, feedback, interaction triggers) is pleasant and self-explanatory. However, some participants failed to successfully teach valid null-space constraints to the system. An analysis of their selected redundancy resolution indicates, that users might not be aware of the current system's capabilities and limitations, which could result in selecting inappropriate redundancy resolutions.

Motivated by these findings, I proposed an adapted variant of the interaction model in Chap. 6. It utilizes online learning of null-space constraints and building of a confidence model for the robot, in order to successively reduce the interaction effort required by the user, already during the *CONFIGURATION* stage. The robot's confidence model shapes the interaction by means of the derived *assistance blending* providing haptic feedback about already learned constraints and demonstration areas. I argue, that by this means the interaction complexity is decreasing continuously, thereby simplifying the entire teaching process which is a major goal of this work. However, a decent evaluation of this approach remains for future work.

In the further course of my work, I again presented results from the user study FlexIRob@Harting, this time to analyze the effectiveness of the separation into *CONFIGURATION* and *PROGRAMMING* stage in Chap. 5. Concerning the proposed *assisted gravity compensation* controller to reduce the teaching complexity compared to standard kinesthetic teaching methods, all expectations are confirmed. Participants utilizing the assisted mode were significantly faster, more accurate and encountered an improved interaction experience. Thus, learning and

embedding environmental constraints into the system's control architecture increases the system's autonomy and reduces the interaction effort on the user's side.

While in the tradition of the programming-by-demonstration paradigm it is logical to rely on kinesthetic teaching also for the programming of tasks, in some situations the actual executed trajectory is not important. Applications such as pick-and-place tasks in manufacturing are rather goal-directed than require a specific manner. In such situations, this simplified definition of a task should also be reflected in a more autonomous robot system, and hence in a further simplified interaction model. Consequently, Chap. 7 presents a path planning method that is purely data-driven by the interaction data gathered through the *CONFIGURATION* stage. In combination with the learned redundancy resolution, an estimated topological map of the reachable workspace enables the robot to navigate autonomously in its workspace while still avoiding the constraints encoded in the demonstration data.

Chap. 8 finally presents an artificial but yet informing example, of how the proposed methods would scale also to other robot platforms. I argue that all proposed concepts do not depend on specific properties of the underlying low-level controllers. In principle, the proposed concepts can be employed on any robot platform with proper compliance features. Future work will address this hypothesis, e.g. by implementing them on robot platforms with more DoF such as the Kuka OmniRob or passively compliant arms such as Festo's Bionic Handling Assistance [9].

Appendix A

FlexIRob: A Flexible Interactive Robot Prototype

A.1 System Overview

An overview of the FlexIRob system prototype, which was used to implement all discussed concepts in this work, is given in Fig. A.1. It consists of the KUKA Lightweight Robot IV, which is the robot platform used for all real-world experiments in this thesis, and three PCs.

A similar setup has been used to conduct the user study FlexIRob@Harting.

A.2 The KUKA Lightweight Robot IV

The first PC is used for real-time communication with and control of the LWR IV via the KUKA fast research interface (FRI, [119]). The latter gives direct low-level real-time access to the KUKA robot controller at high rates of up to 1 kHz. In our setup, a rate of 10 kHz is used. By means of this protocol, we send joint angle commands $\mathbf{q}_{\text{cmd}}$ to the robot and receive a thorough status report of the robot including joint positions $\mathbf{q}_{\text{curr}}$, Cartesian positions $\mathbf{x}_{\text{curr}}$ of the end-effector, torque measurements as well as estimated external forces f_{int} and torques τ_{int}.

A.3 Implementation of Interaction Model

Throughout this thesis, all interaction controllers are implemented using the internal joint impedance controller of the LWR IV [5], which allows to physically deflect the robot from its current commanded position $\mathbf{q}_{\text{cmd}}$. The compliance is determined by the stiffness and damping parameters utilized in the low-level control. They can be set separately for each joint but are set equally,in the current prototype setup. The parameters vary between the different interaction stages, i.e. with the different interaction controllers and are given in Tab. A.1

According to Sect. 2.3, six interaction triggers are used to switch between the respective control modes and interaction stages, of which the following two are the most used and are based on physically interacting or not interacting with the robot:

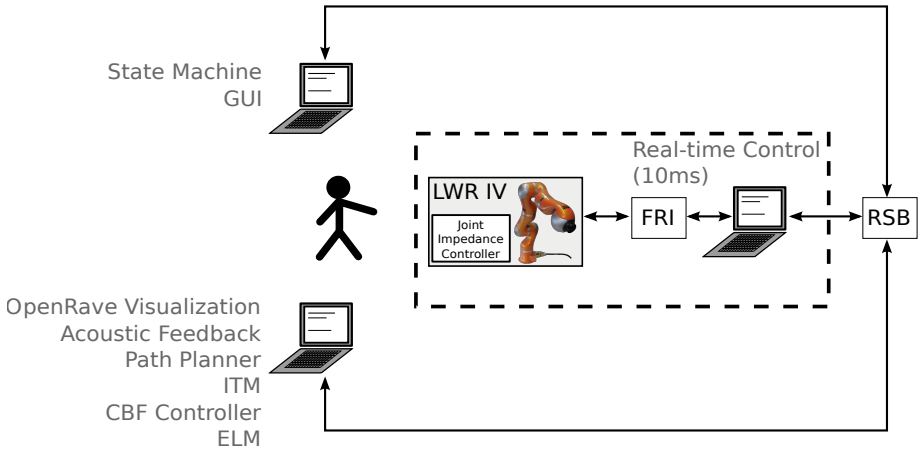


Fig. A.1: Illustration of the FlexIRob system including the KUKA Lightweight Robot IV, two PCs and a tablet PC for a graphical user interface.

Tab. A.1: Stiffness and damping values chosen for the different interaction control modes.

Mode	stiffness	damping	inertia	
gravity compensation	20.0 $\frac{Nm}{rad}$	0.7 $\frac{Nm*s}{rad}$	0.1	Sect. 4.1
compliant recording	80.0 $\frac{Nm}{rad}$	0.7 $\frac{Nm*s}{rad}$	–	Sect. 4.1
hierarchical control	300.0 $\frac{Nm}{rad}$	0.95 $\frac{Nm*s}{rad}$	–	Sect. 3.1
assisted gravity compensation	50.0 $\frac{Nm}{rad}$	0.7 $\frac{Nm*s}{rad}$	0.5	Sect. 5.1
assistance blending	50.0 $\frac{Nm}{rad}$	0.7 $\frac{Nm*s}{rad}$	0.1	Sect. 6.1

`on_affected` is based on the estimated external torques τ_{int}. Requiring that the robot is not moving, `on_affected` is triggered when $max(\tau_{\text{int}}) > 1.5$ Nm. As external torques relate to interaction forces applied by the user, this trigger is used to switch to interaction stages where the robot is moved to e.g. *APPROACHING* the next training area or to start *RECORDING* data.

`on_converged` is triggered when the robot once was in interaction with the user, i.e. `on_affected` was triggered before, and then is left without moving for more than 2 seconds. As the user typically removes his or her hands of the robot to accomplish this trigger, it acknowledges it with a short acoustic signal.

The remaining four triggers `start_configuration`, `finished_configuration`, `start_programming` and `finished_programming` are based on pressing a button on a graphical user interface, which is placed besides the robot's working place. This procedure has proven to work well during frequent demonstrations of the system prototype.

Note, that these latter four triggers have not been used in the FlexIRob@Harting study. Instead, switching between *CONFIGURATION* and *PROGRAMMING* was done manually by the experimenters.

A.4 Software Abstractions for Learning From Demonstrations

The KUKA Lightweight Robot IV and it's respective control aspects such as sensing and actuation capabilities are modeled in the Robot Control Interface (RCI) library [148]. RCI provides a set of domain-specific abstractions to represent common features of compliant robotics systems.

On top of that, all introduced components and the control flow of the FlexIRob system are implemented within the Compliant Control Architecture (CCA) [148], an event-based, middleware-agnostic component architecture for robotics research, focusing on (real-time) control of compliant platforms.

Such an example is the utilized hierarchical controller as discussed in Sect. 2.3.2. It is motivated by the ideas of [82] to incrementally build complex robot behavior from a simple control basis. The main idea of the control basis framework (CBF) is to assume, that null-space motions are generated by lower order controller, so-called subordinate controllers, which recursively project into the null-space of higher order controllers. The implementation in this thesis utilizes the CBF C++ library available in [149] and accordingly implements a CCA interface to embed the controller into the data-flow control architecture [148]. As for the FlexIRob prototype, we use a primary controller for task space motions and a subordinate controller that is informed by the learned redundancy mapping.

In order to simulate robot motions and check for collisions, the Open Robotics Automation Virtual Environment (OpenRAVE) software is utilized in FlexIRob [150]. Regarding the user study FlexIRob@Harting, all system data has been recorded by means of our middleware tools and - in a post-processing procedure - passed to the OpenRave simulator for checking collisions in the simulated environment.

A.5 Middleware

As shown in Fig. A.1, the FlexIRob prototype uses the Robotics Service Bus (RSB) [151] as its central interprocess communication framework. RSB is a message-oriented, event-driven middleware and implemented as flexible, lightweight toolkit, providing programming-language-independent concepts and communication interfaces.

Appendix B

FlexIRob@Harting: A User Study on Physical Human-Robot Interaction

The user study *FlexIRob@Harting* was designed to assess the feasibility of kinesthetic teaching methods for redundant robotic manipulators in the context of industrial scenarios such as teaching of welding or gluing trajectories. It was conducted in April 2012 with 49 industrial workers from HARTING [24], a medium-sized manufacturing company, at their production site. According to the current state of knowledge this was one of the first large field studies on kinesthetic teaching of a robotic manipulator in industrial scenarios carried out with industrial, robotics-inexperienced workers.

The design, structure and main results of the study have been published in [22]. For the sake of completeness, the study design and course of action during the study are briefly reported here as well.

B.1 Study Design and Interaction Model

The study was designed along the proposed decomposition of the overall teaching procedure into a task-independent *CONFIGURATION* stage and a task-dependent *PROGRAMMING* phase as discussed in Sect. 2.3. The high-level objective of the study was to evaluate this interaction scheme with industrial workers, but also to validate whether kinesthetic teaching and learning methods are usable for inexperienced users. Therefore, the following three questions have been the main guideline for the design of the study:

- What is the general experience with our FlexIRob system?
- How feasible is the task-independent configuration for naive users?
- How helpful is the proposed decomposition of the interaction into *CONFIGURATION* and *PROGRAMMING* stage by means of the *assisted gravity compensation* control mode?

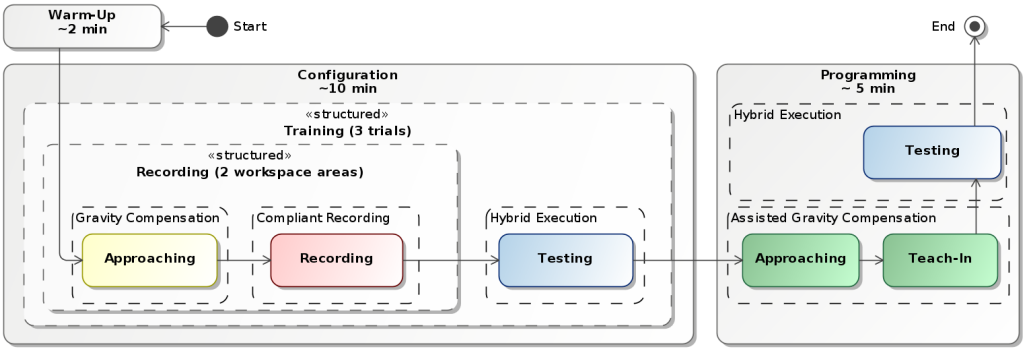


Fig. B.1: Illustration of the interaction workflow during the FlexIRob@Harting study, exemplarily for the assisted group (group A). After a short warm-up phase, the *CONFIGURATION* stage is repeated three times for each participant, each followed by a testing stage. The teach-in of the *PROGRAMMING* phase is done once and then replayed as feedback about teaching success to the user.

As the latter was a central research question in the study, it was reflected in the conditions of the experiment design. In the one condition, the participant performs a teach-in task assisted by the robot by means of the *assisted gravity compensation* controller (group A). In the other condition, the task needs to be solved without assistance, i.e. participants used the *gravity compensation* controller (group N). The course of action for each participant is illustrated in Fig. B.1 and is separated in three main stages, a warm-up phase, a repeated *CONFIGURATION* stage, and finally the *PROGRAMMING* phase where the two conditions (group A vs. group N) played the central role. Throughout all stages, a computer screen showed a simulated model of the LWR IV exactly mirroring the motions of the real robot, and acoustic feedback about the interaction trigger `on_converged` was given to the participants as described Sect. 2.3.1.

B.1.1 Warm-up phase

The warm-up was designed for the participants to familiarize themselves with the KUKA Lightweight Robot IV and the interaction with it. It started by first showing an instructional video[10] to the participants about how all seven joints could be moved and how the LWR IV could be moved into different positions. They were informed about the two main operational modes, the *gravity compensation* and the *compliant recording* (cf. Sect. 4.1) and about the interaction triggers `on_converged` and `on_affected` for switching between them. The participants were also told that they could not damage anything and could safely touch the robot's gripper. Afterwards, the participants were instructed to familiarize themselves with the robot in physical human-robot interaction. They were asked to test the interaction modes

[10] `http://www.cor-lab.de/system/files/Instruktion.mp4`

and triggers, e.g. to reproduce what they saw in the video. Depending on what each participant tried, the warm-up stage took two to five minutes.

B.1.2 Configuration Phase

After participants had finished the warm-up, the *CONFIGURATION* stage started, where they were asked to teach the robot redundancy resolutions according to the presented confined workspace (cf. Fig. 4.8). The workspace was separated into $K = 2$ training areas (left and right) indicated by blue boxes. First, they had to move the robot to the left working area (*APPROACHING*) and were instructed to perform circular movements for training this area (*RECORDING*). Then, they had to move the arm to the right working area (*APPROACHING*) and to repeat recording training there (*RECORDING*). With the recorded data, the neural network (ELM, cf. Sect. 3.2.1) was trained and embedded into the *hierarchical control* mode. Subsequently, a test trajectory in form of a straight line between left and right training area was performed by the system serving as feedback to the participants regarding their teaching success. Good demonstrations resulted in collision-free movements from left to right whereas other demonstrations resulted in collisions with the styrofoam obstacle.

In order to investigate the particpants learning effects over several trials and how much instruction participants needed, this procedure was repeated in three trials with increasingly informative instructions. These hints were the same for all participants, and were chosen such that with ongoing trials the possibilities for moving the robot from left to right working area were more and more restricted, and thus should help the participants to find valid redundancy resolutions:

1. trial:
   `The robot works in two work spaces` *`[show spaces]`* `and moves between them. Please consider this during the teaching.`
2. trial:
   `This is the working space` *`[show space]`*`; the robot should not leave this space with the gripper. Please consider this during the teaching.`
   (The difference from the first instruction is that the whole model work-space was shown, not only the two boxes on the left and on the right.)
3. trial:
   `The gripper is always supposed to point down and should make a straight line between both work spaces. Please consider this during the teaching.`

B.1.3 Wire-loop game

The last part of the study was the wire loop game which simulated a teach-in of a concrete task to the robot. During this task the experimental manipulation took place (group A vs. group N). Participants were randomly assigned to group N or to group A. In order to get reproducible results, all assisted participants utilized the same reference neural network (ELM), trained by an expert in advance and embedded into the *assisted gravity compensation* controller. After the teach-in,

participants were told to step back to watch a replay of their demonstrated teach-in.

B.1.4 Subjective Assessment of Interaction

Finally, the participants were given the questionnaire (cf. Sect. B.2). They were instructed how to fill in the questionnaire and were given a short overview of the different topics. They were instructed to ask if there were any questions they did not understand. After the completion of the questionnaire participants were debriefed. To keep the conditions and instructions stable through the whole experiment each conductor of the study adhered to a set of guidelines[11] during the experiment.

B.2 Questionnaire Design

The questionnaire was adapted to the tasks and to the specific demographic background of the participants. Characteristics of the sample included educational background and the fact that some employees were not native German speakers. The questions were derived from expert-discussions and pre-tested with four students from Bielefeld University. Due to the characteristics of the sample, all questions were also discussed with a member of the staff of HARTING. The resulting questionnaire contained 36 questions structured into seven topics:

- general experience with the robot during the interaction
- subjective experience of the wire-loop game
- whether or not participants could imagine the robot supporting them during various tasks
- demographic variables
- other control variables (e.g. stereoscopic vision)
- previous experience with robots
- suggestions/ ideas for improvement

The items concerning general experience covered important characteristics of the robot like threat, reliability, and intelligence, and characteristics of the handling like ease, pleasantness, and cognitive load during handling. All items concerning the robot and the task were rated on a five-point Likert scale ranging from 1 (yes/very much) to 5 (no/not at all). All items covering previous experience, support by the robot and control variables were rated on a three-point Likert scale

[11] Full instructions available at: `http://www.cor-lab.de/system/files/instructions.pdf`

ranging from 1 (yes/very much) to 3 (no/not at all). This reduction of dimensions was used to facilitate the completion of the questionnaire[12].

B.3 Data Assessment and Analysis

Despite the questionnaire, all relevant system data was recorded for each participant. This included the recorded training data and resulting trained networks, the demonstrated teach-in trajectories, the performed test movements, and other system data of the LWR IV such as joint torques and interaction forces. To allow exact quantification of the collisions that occurred during the study, we replayed and processed all relevant movements in a simulated environment (cf. Chap. A), where the simulated obstacles were modeled exactly to fit the physical ones in the study setup. By this means, we could, for instance, exactly count the number of collisions with the environment without manual annotation of the recorded video data.

B.4 Clustering of the Participants' Selected Redundancy Resolutions

For the analysis presented in Sect. 4.3.4, the training data of the participants recorded in the third trial of the *CONFIGURATION* stage have been clustered. The clustering is based on the initial configuration $\mathbf{q}^L_{\text{fixed}}, \mathbf{q}^R_{\text{fixed}}$ selected by the users in the left and right area and is done *separately for each of the two training areas*. As a result, the procedure described in the following will yield to a set of clusters $\{\text{L}_i\}$ for the left area and another set $\{\text{R}_j\}$ for the right area. Hence, after clustering, the participants' chosen joint configurations $\mathbf{q}^L_{\text{fixed}}$ can be assigned to one of $\{\text{L}_i\}$, and $\mathbf{q}^R_{\text{fixed}}$ to one of $\{\text{R}_j\}$.

In the following, I describe the procedure only for the left training area; clustering in the right area proceeds analogously. First of all, the analysis is based not on all joint values of the robot but only on the first six joints. The last joint q_7 only corresponds to a rotation of the mounted tool (gripper) and, hence, does not affect either collision avoidance or task space accuracy in this context. The data base for clustering consists in the 49 initially chosen joint configurations $(\mathbf{q}^L_{\text{fixed}})_p,\ p = 1, \ldots, 49$. The utilized clustering algorithm is the k-means algorithm [117], with a predefined k and initialized with cluster center candidates.[13] The number k is chosen after a visual inspection of the selected initial joint configurations for all participants by means of projecting the six-dimensional joint val-

[12] Available at: `http://www.cor-lab.de/system/files/FragebogenHartingstudy.pdf`

[13] Please note, that the variable k here indicates the number of clusters or classes, as typical for clustering algorithms, but is not to be confused with the number K of training areas in the context of this thesis. In this section the number of training areas is $K = 2$, but k will differ according to the analysis of the participants' selected configurations $\mathbf{q}^L_{\text{fixed}}$ and $\mathbf{q}^R_{\text{fixed}}$.

name	cluster initialization $(q_1, \ldots, q_6)^\mathsf{T}$ in [rad]					
L_1	0	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
L_2	0	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
L_3	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
R_1	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
R_2	0	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
R_3	0	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
R_4	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
R_5	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
R_6	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
R_7	$\frac{\pi}{4}$	$\frac{\pi}{2}$	$\frac{\pi}{4}$	0	$-\frac{\pi}{4}$	$\frac{\pi}{2}$

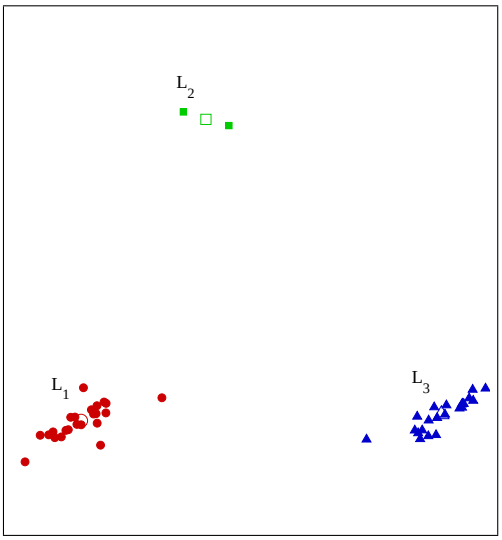
Tab. B.1: Initialization of the k-means algorithm for the cluster analysis in left and right training area.

ues to two-dimensional space using the method of multi-dimensional scaling [118]. Fig. B.2(a) shows this projection. Note, that the color-coding already indicates the clusters, but the projection was not informed about that. Hence, the distribution of the projected data already reveals three inherent clusters in the training data, i.e. $k = 3$. Before starting the k-means algorithm clustering, the cluster centers must be initialized.[14] This is done with the values shown in Tab. B.1, which relate to discrete solutions of the LWR IV's inverse kinematics to realize a task space position in the left training area, but with a bias towards right angles in the posture. Hence, they represent "prototypical postures" for the manipulator to reach around the obstacle to the left area. With this initialization, the k-means algorithm proved to reliably and deterministically identify the three clusters $\mathrm{L}_1, \mathrm{L}_2$ and L_3, to one of which each participant can be assigned to as shown in Tab. B.2.

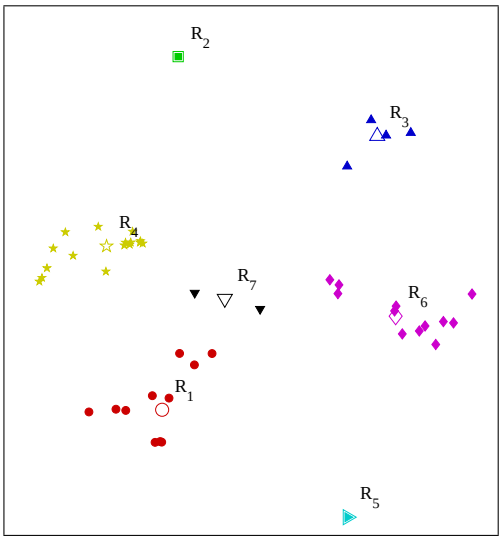
The same procedure was conducted for the right training area with $k = 7$, the two-dimensional, projected visualization shown in Fig. B.2(b) and the cluster center initializations given in Tab. B.1.

Exemplarily visualizations for some of the cluster centers' resulting postures are shown in Fig. 4.14(a) and Fig. 4.14(b).

[14] Running the k-means algorithm with randomly initialized cluster centers did not reveal any reliable results in the presented analysis.



(a) Two-dimensional visualization of the clustered initial postures $\mathbf{q}^{L}_{\text{fixed}}$ in left training area.



(b) Two-dimensional visualization of the clustered initial postures $\mathbf{q}^{R}_{\text{fixed}}$ in right training area.

Fig. B.2: Visualization of the participants' training data distribution in left and right training area using multi-dimensional scaling [118]. The color-coding indicates the clustering. The corresponding cluster centers are displayed as large, non-filled markers.

clusters							# users
name	center $(q_1, \ldots, q_6)^\mathsf{T}$ in [rad]						
L_1	−0.09	1.43	−1.45	−1.24	1.33	1.54	23
L_2	−0.10	1.45	−1.52	−1.29	−1.72	−1.48	2
L_3	−0.10	1.54	1.45	1.21	−1.54	1.68	22
R_1	0.14	1.44	1.49	−1.25	−1.45	1.58	11
R_2	0.12	1.64	−1.24	1.27	−1.76	−1.79	1
R_3	0.09	1.49	−1.53	1.35	1.61	1.63	4
R_4	1.26	1.52	1.51	1.20	−1.59	1.70	16
R_5	1.38	1.44	−1.59	−1.31	−1.78	−1.62	1
R_6	1.22	1.45	−1.31	−1.07	1.31	1.63	12
R_7	0.61	1.45	0.47	−0.16	−0.39	1.77	2

Tab. B.2: Results of the cluster analysis of the participants' selected redundancy resolutions $\mathbf{q}_{\text{fixed}}$ in left and right training area.

Appendix C

Related References by the Author

[22] **Sebastian Wrede, Christian Emmerich, Ricarda Grünberg, Arne Nordmann, Agnes Swadzba, and Jochen J. Steil. A User Study on Kinesthetic Teaching of Redundant Robots in Task and Configuration Space. *Journal of Human-Robot Interaction*, 2(1):56–81, 2013**

The contribution of this work is the FlexIRob@Harting study, a large study on physical human-robot interaction with 49 participants. The study was designed along the proposed separation of the teaching procedure into *CONFIGURATION* and *PROGRAMMING* stage. The author contributed to the research design, implemented large parts of the desired concepts on the FlexIRob system prototype, was strongly involved in the conduction of the study, and conducted the data analysis concerning the relevant system data. The ideas reported in that work serve as basis for the derivation of the separated teaching scheme proposed in Sect. 2.3. Results of this study are reported in Sect. 4.3 and Sect. 5.2.

[23] **Christian Emmerich, Arne Nordmann, Agnes Swadzba, Jochen J. Steil, and Sebastian Wrede. Assisted Gravity Compensation to cope with the complexity of kinesthetic teaching on redundant robots. In *2013 IEEE International Conference on Robotics and Automation*, pages 4322–4328. Ieee, May 2013**

This work formally derives the *assisted gravity compensation* controller as a general control scheme, applicable also to other robot platforms to reduce the complexity of kinesthetic teaching methods. The author contributed to research design and further analyzed the system data to obtain the motivational results reported in Sect. 1.2.1.

[109] **Arne Nordmann, Christian Emmerich, Stefan Rüther, Andre Lemme, Sebastian Wrede, and Jochen J. Steil. Teaching Nullspace Constraints in Physical Human-Robot Interaction using Reservoir Computing. In *International Conference on Robotics and Automation (ICRA)*, pages 1868–1875, 2012**

The paper presents the idea of teaching and learning redundancy resolutions in close physical human-robot interaction and presents a first evaluation in four different setups. Hence, the concepts presented in Sect. 2.3.1 are derived from that ideas. The results of the evaluation are reported in Sect. 3.3.1. The author contributed to the research design and also to the implementation and data analysis of the reported evaluations.

[107] **Klaus Neumann, Christian Emmerich, and Jochen J. Steil. Regularization by Intrinsic Plasticity and Its Synergies with Recurrence for Random Projection Methods. *Journal of Intelligent Learning Systems and Applications*, 04(03):230–246, 2012**

The paper investigates the role of intrinsic plasticity as a feature regularization scheme and recurrence as a technique to produce a non-linear mixture of sigmoid features for random projections. Although none of the work reported there is reused in this thesis, the obtained results served as knowledge base to derive the concepts and to conduct the evaluations of Chap. 3

[115] **Jochen J. Steil, Christian Emmerich, Agnes Swadzba, Ricarda Grünberg, Arne Nordmann, and Sebastian Wrede. Kinesthetic Teaching Using Assisted Gravity Compensation for Model-Free Trajectory Generation in Confined Spaces. In Florian Röhrbein, Germano Veiga, and Ciro Natale, editors, *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe:*, number April in Springer Tracts in Advanced Robotics, pages 107–127. Springer International Publishing, 2014**

The book chapter summarizes the obtained results obtained within the ECHORD experiment "MoFTaG - Model-free trajectory generation" concerning experimentation with kinesthetic teaching methods on the FlexIRob prototype. This includes the extensive evaluation during the FlexIRob@Harting study [22]. It also presents results concerning the idea of implicitly modeled scenes and a reachable workspace analysis of learned redundancy resolutions. This was the contribution of the author in is therefore - in a more extensive evaluation - reported in Sect. 3.3.2 and Sect. 4.2.2.

[132] **Daniel Seidel, Christian Emmerich, and Jochen J. Steil. Model-free Path Planning for Redundant Robots using Sparse Data from Kinesthetic Teaching. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4381–4388, 2014**

This work introduces model-free path planning for redundant robot in confined spaces. The main development of the method was conducted by Daniel Seidel in his master's thesis [131], supervised by the author, and then adopted by the author for publication. The published derivation and evaluation of

this approach are reported in Chap. 7. The author contributed to the research design and to the publication in [132].

[108] **Christian Emmerich, R. Felix Reinhart, and Jochen J. Steil. Recurrence enhances the spatial encoding of static inputs in reservoir networks. In *International Conference on Artificial Neural Networks (ICANN)*, number ii in Lecture Notes in Computer Science, pages 148–153. Springer Berlin Heidelberg, 2010**

The paper sheds light on certain key ingredients for generalization robustness of random-projection methods such as discussed in Sect. 3.2.1. Similar to [107], none of that work is reproduced here. However, the insights gained in that evaluations helped to design and conduct the evaluations in Chap. 3.

References

[1] T. Lozano-Perez. Robot programming. *Proceedings of the IEEE*, 71(7):821–841, 1983.

[2] AndreaKrasa Sethi and SureshPal Sethi. Flexibility in manufacturing: A survey. *International Journal of Flexible Manufacturing Systems*, 2(4):289–328, July 1990.

[3] Mario Hermann, Tobias Pentek, and Boris Otto. Design Principles for Industrie 4.0 Scenarios: A Literature Review. 2015.

[4] Wolfgang Dorst and Astrid Scheibe. Umsetzungsstrategie Industrie 4.0. 2015.

[5] Alin Albu-Schäffer, S. Haddadin, Ch. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger. The DLR lightweight robot: design and control concepts for robots in human environments. *Industrial Robot An International Journal*, 34(5):376–385, 2007.

[6] N G Tsagarakis, M Laffranchi, B Vanderborght, and D G Caldwell. A Compact Soft Actuator Unit for Small Scale Human Friendly Robots. *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, 2009.

[7] Nikos G Tsagarakis, Irene Sardellitti, and Darwin G Caldwell. A new variable stiffness actuator (CompAct-VSA): Design and modelling, 2011.

[8] Rainer Bischoff, Johannes Kurth, Günter Schreiber, Ralf Koeppe, Alin Albu-Schäffer, Der Beyer, Oliver Eiberger, Sami Haddadin, Andreas Stemmer, Gerhard Grunwald, and Kuka Roboter Gmbh. The KUKA-DLR Lightweight Robot arm: a new reference platform for robotics research and manufacturing. In *Joint 41th International Symposium on Robotics and 6th German Conference on Robotics*, pages 741–748. VDE VERLAG GmbH, 2010.

[9] Andrzej Grzesiak, Ralf Becker, and Alexander Verl. The Bionic Handling Assistant: A Success Story of Additive Manufacturing. *Assembly Automation*, 31(4):329 – 333, 2011.

[10] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, chapter 59, pages 1371–1394. Citeseer, 2007.

[11] Stefan Schaal, Auke Jan Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 358(1431):537–47, March 2003.

[12] E. Sahin Conkur and Rob Buckingham. Clarifying the definition of redundancy as used in robotics. *Robotica*, 15(5):583–586, 1997.

[13] Bruno Siciliano. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems*, 3(3):201–212, September 1990.

[14] Daimler AG. Leichtbauroboter im Piloteinsatz im Mercedes-Benz Werk Untertürkheim, 2009.

[15] B. Akgun, M. Cakmak, J W Yoo, and A.L. Thomaz. Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective. In *International Conference on Human-Robot Interaction (HRI)*, 2012.

[16] C. Breazeal, M. Siegel, M. Berlin, J. Gray, Roderic Grupen, P. Deegan, J. Weber, K. Narendran, and J. McBean. Mobile, dexterous, social robots for mobile manipulation and human-robot interaction. In *Special Interest Group on GRAPHics and Interactive Techniques*, 2008.

[17] Robotiq. *Collaborative Robot Ebook*. 5 edition, 2015.

[18] European Strategic Robotics Platform. Robotic Visions to 2020 and beyond - The Strategic Research Agenda (SRA) for robotics in Europe. Technical report, European Robotics Technology Platform, 2009.

[19] R Alami, Alin Albu-Schäffer, A Bicchi, R Bischoff, R Chatila, A De Luca, A De Santis, G Giralt, J Guiochet, G Hirzinger, and Others. Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In *IROS Workshop on pHRI - Physical Human-Robot Interaction in Anthropic Domains*, volume 6. Citeseer, 2006.

[20] DB Kaber and JM Riley. Effects of visual interface design, and control mode and latency on performance, telepresence and workload in a teleoperation task. In *Proceedings of the XIVth Triennial Congress of the International Ergonomics Association and 44th Annual Meeting of the Human Factors and Ergonomics Society*, pages 503–506, 2000.

[21] Aaron Steinfeld, Terrence Fong, Moffett Field, Michael Lewis, Jean Scholtz, and Alan Schultz. Common Metrics for Human-Robot Interaction. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 33–40, 2006.

[22] Sebastian Wrede, Christian Emmerich, Ricarda Grünberg, Arne Nordmann, Agnes Swadzba, and Jochen J. Steil. A User Study on Kinesthetic Teaching of Redundant Robots in Task and Configuration Space. *Journal of Human-Robot Interaction*, 2(1):56–81, 2013.

[23] Christian Emmerich, Arne Nordmann, Agnes Swadzba, Jochen J. Steil, and Sebastian Wrede. Assisted Gravity Compensation to cope with the complexity of kinesthetic teaching on redundant robots. In *2013 IEEE International Conference on Robotics and Automation*, pages 4322–4328. Ieee, May 2013.

[24] HARTING technology group. `http://www.harting.com`. Accessed: 2015-05-25.

[25] Alin Albu-Schäffer, Christian Ott, and Gerd Hirzinger. A Unified Passivity Based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots. In Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte, editors, *Robotics Research*, volume 28 of *Springer Tracts in Advanced Robotics*, pages 5–21. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[26] A. D'Souza, S. Vijayakumar, and Stefan Schaal. Learning inverse kinematics. *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, 1(4):298–303, 2001.

[27] Matthias Rolf, Jochen J. Steil, and Michael Gienger. Efficient exploration and learning of whole body kinematics. In *IEEE 8th International Conference on Development and Learning (ICDL 2009)*, pages 1–7, Shanghai, CH, 2009. Ieee.

[28] R. Felix Reinhart and Jochen J. Steil. Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot iCub. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 323–330. Citeseer, 2009.

[29] Sebastian Wrede, Michael Johannfunke, Andre Lemme, Arne Nordmann, Stefan Rüther, Alicia Weirich, and Jochen J. Steil. Interactive Learning of Inverse Kinematics with Nullspace Constraints using Recurrent Neural Networks. In *20th Workshop on Computational Intelligence*, Dortmund, 2010. Fachausschuss Computational Intelligence der VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Fachausschuss Computational Intelligence der VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik.

[30] FlexIRob - Flexible Interactive Robot Prototype. `http://www.cor-lab.de/flexirob`. Accessed: 2015-05-25.

[31] Stefano Chiaverini, Giuseppe Oriolo, and Ian D Walker. Kinematically Redundant Manipulators. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, chapter 11, pages 245–268. Springer Berlin Heidelberg, 2008.

[32] Charles W. Wampler. Inverse kinematic functions for redundant spherical wrists. *IEEE Transactions on Robotics and Automation*, 5(1):106–111, 1989.

[33] Y. Nakamura and H. Hanafusa. Optimal Redundancy Control of Robot Manipulators. *The International Journal of Robotics Research*, 6(1):32–42, March 1987.

[34] JE Bobrow, S Dubowsky, and JS Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985.

[35] Arati S. Deo and Ian D. Walker. Minimum effort inverse kinematics for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 13(5):767–775, 1997.

[36] A. A. Maciejewski and C. A. Klein. Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985.

[37] L. Zlajpah and B. Nemec. Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2(October):0–5, 2002.

[38] Hisashi Sugiura, Michael Gienger, Herbert Janssen, and Christian Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2053–2058. Ieee, October 2007.

[39] A S Phung, J Malzahn, F Hoffmann, and T Bertram. Get Out of the Way - Obstacle Avoidance and Learning by Demonstration for Manipulation. In *IFAC World Congress*, pages 11514–11519, 2011.

[40] Kenneth N Groom, Anthony A Maciejewski, and Venkataramanan Balakrishnan. Real-time failure-tolerant control of kinematically redundant manipulators. *IEEE Transactions on Robotics and Automation*, 15(6):1109–1116, 1999.

[41] Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Motion control of redundant robots under joint constraints: Saturation in the null space. In

Proceedings - IEEE International Conference on Robotics and Automation, pages 285–292, 2012.

[42] Sang-ik An and Dongheui Lee. Prioritized Inverse Kinematics with Multiple Task Definitions. In *IEEE International Conference on Robotics and Automation*, pages 1423–1430, 2015.

[43] Olivier Sigaud, Camille Salaun, and Vincent Padois. On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, 2011.

[44] Matthias Rolf and Jochen J. Steil. Efficient exploratory learning of inverse kinematics on a bionic elephant trunk. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1147–1160, 2014.

[45] Michael I Jordan. Constrained supervised learning. *Journal of Mathematical Psychology*, 36(3):396–425, 1992.

[46] Camille Salaun, Vincent Padois, and Olivier Sigaud. Control of redundant robots using learned models: An operational space control approach. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 878–885, 2009.

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

[48] Chrystopher L. Nehaniv and Kerstin Dautenhahn. The correspondance problem. In Chrystopher L. Nehaniv and Kerstin Dautenhahn, editors, *Imitation in Animals and Artifacts*, pages 41–61. MIT Press, 2002.

[49] Russell H. Taylor and Dan Stoianovici. Medical Robotics in Computer-Integrated Surgery. *IEEE Transactions on Robotics and Automation*, 19(5):765–781, 2003.

[50] B. Akgun, Kaushik Subramanian, and A.L. Thomaz. Novel Interaction Strategies for Learning from Teleoperation. In *2012 AAAI Fall Symposium Series*, pages 2–9, 2012.

[51] A. Muxfeldt, Jan-Henrik Kluth, and Daniel Kubus. Kinesthetic Teaching in Assembly Operations - A User Study. In Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald, editors, *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, volume 8810, pages 533–544. Springer International Publishing, 2014.

[52] Joe Saunders, Miscellaneous Imitation, Programming Demonstration, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 118–125, 2006.

[53] Sylvain Calinon and A G Billard. What is the teacher's role in robot programming by demonstration? Toward benchmarks for improved learning. *Interaction Studies*, 8(3):441–464, 2007.

[54] Agostino De Santis, Bruno Siciliano, Alessandro De Luca, and Antonio Bicchi. An atlas of physical human-robot interaction. *Mechanism and Machine Theory*, 43(3):253–270, March 2008.

[55] Neville Hogan. Impedance control: An approach to manipulation, part i- iii. *ASME Journal of Dynamic Systems, Measurements, and Control*, 107:1–24, 1985.

[56] Christian Ott, Ranjan Mukherjee, and Yoshihiko Nakamura. Unified Impedance and Admittance Control. In *2010 IEEE International Conference on Robotics and Automation*, pages 554–561. IEEE, May 2010.

[57] J. F. Queisser, K Neumann, Matthias Rolf, R. Felix Reinhart, and J J Steil. An active compliant control mode for interaction with a pneumatic soft robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–579. IEEE, September 2014.

[58] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, HRI '07, pages 255–262, New York, NY, USA, 2007. ACM.

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

[60] B. Akgun, M. Cakmak, Karl Jiang, and A.L. Thomaz. Keyframe-based Learning from Demonstration. *International Journal of Social Robotics*, 4(4):343–355, June 2012.

[61] Gu Ye and Ron Alterovitz. Demonstration-Guided Motion Planning. In *International Symposium on Robotics Research (ISRR)*, 2011.

[62] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[63] R. Felix Reinhart, Andre Lemme, and Jochen Jakob Steil. Representation and generalization of bi-manual skills from kinesthetic teaching. In *IEEE-RAS International Conference on Humanoid Robots*, pages 560–567, 2012.

[64] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input. *Advanced Robotics*, 25(5):581–603, January 2011.

[65] C. Schou, J. S. Damgaard, S. Bogh, and O. Madsen. Human-robot interface for instructing industrial tasks using kinesthetic teaching. In *2013 44th International Symposium on Robotics, ISR 2013*, volume 24, pages 1463–1467, 2013.

[66] Heni Ben Amor, Erik Berger, David Vogt, and Bernhard Jung. Kinesthetic bootstrapping: Teaching motor skills to humanoid robots through physical interaction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5803 LNAI:492–499, 2009.

[67] Alin Albu-Schäffer and G. Hirzinger. Cartesian impedance control techniques for torque controlled light-weight robots. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, (May):657–663, 2002.

[68] Zheng Wang, Angelika Peer, and Martin Buss. An HMM approach to realistic haptic Human-Robot interaction. In *Proceedings - 3rd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, World Haptics 2009*, pages 374–379, 2009.

[69] Friedrich Lange, Claudius Jehle, Michael Suppa, and Gerd Hirzinger. Revised force control using a compliant sensor with a position controlled robot. In *2012 IEEE International Conference on Robotics and Automation*, number May, pages 1532–1537. IEEE, May 2012.

[70] Luigi Villani and Joris De Schutter. Force Control. In *Springer Handbook of Robotics*, pages 161–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[71] Ville Infante, Marta Lopez; Kyrki. Usability of Force-Based Controllers in Physical Human-Robot Interaction. In *HRI '11: Proceeding of the 6th international conference on Human-robot interaction*, pages 355–362, 2011.

[72] David Nitzan and CA Rosen. Programmable Industrial Automation. *IEEE Transactions on Computers*, C-25(12):1259–1270, December 1976.

[73] Andreas Stolt, Magnus Linderoth, Anders Robertsson, and Rolf Johansson. Force controlled robotic assembly without a force sensor. In *2012 IEEE International Conference on Robotics and Automation*, pages 1538–1543. IEEE, May 2012.

[74] Andrea Cherubini, Robin Passama, Philippe Fraisse, and André Crosnier. A unified multimodal control framework for human-robot interaction. *Robotics and Autonomous Systems*, 70:106–115, 2015.

[75] Petar Kormushev, Dragomir N. Nenchev, Sylvain Calinon, and Darwin G. Caldwell. Upper-body kinesthetic teaching of a free-standing humanoid robot. In *2011 IEEE International Conference on Robotics and Automation*, pages 3970–3975. IEEE, May 2011.

[76] Ioannis Iossifidis and Axel Steinhage. Controlling a Redundant Robot Arm by Means of a Haptic Sensor. *VDI BERICHTE: ROBOTIK 2002, Leistungsstand - Anwendungen - Visionen*, pages 269–274, 2002.

[77] R. Felix Reinhart. *Reservoir Computing with Output Feedback*. PhD thesis, Bielefeld University, 2011.

[78] Matthias Rolf. *Goal Babbling for an Efficient Bootstrapping of Inverse Models in High Dimensions*. PhD thesis, 2012.

[79] Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, 1992.

[80] Jörn Diedrichsen, Reza Shadmehr, and Richard B Ivry. The coordination of movement: optimal feedback control and beyond. *Trends in cognitive sciences*, 14(1):31–9, January 2010.

[81] Luis Sentis and Oussama Khatib. Synthesis of Whole-body Behaviors through Hierarchical Control of Behavioral Primitives. *International Journal of Humanoid Robotics*, 2(4):505–518, December 2005.

[82] Roderic Grupen and Manfred Huber. A framework for the development of robot behavior. In *2005 AAAI Spring Symposium Series: Developmental Robotics*. Stanford University, 2005.

[83] Stephen Hart, Shiraj Sen, Shichao Ou, and Roderic Grupen. The Control Basis API - A Layered Software Architecture for Autonomous Robot Learning. In *IEEE Conference on Robotics and Automation (ICRA 2009)*, 2009.

[84] Sylvain Calinon and Aude Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 367–372, 2008.

[85] Auke Jan Ijspeert, J. Nakanishi, and Stefan Schaal. Trajectory formation for imitation with nonlinear dynamical systems. *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, 2:752–757, 2001.

[86] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *2010 IEEE International Conference on Robotics and Automation*, number 3, pages 2397–2403. Ieee, May 2010.

[87] J. Kober, J. a. Bagnell, and Jan Peters. Reinforcement Learning in Robotics: A Survey. *The International Journal of Robotics Research*, August 2013.

[88] Gerhard Grunwald. Touch: The direct type of human interaction with a redundant service robot. In *10th IEEE International Workshop on Robot and Human Interactive Communication, 2001. Proceedings.*, pages 347–352, 2001.

[89] G. Schreiber, C. Ott, and G. Hirzinger. Interactive redundant robotics: control of the inverted pendulum with nullspace motion. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 1, pages 158–164, 2001.

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

[91] Panagiotis K Artemiadis, Pantelis T Katsiaris, and Kostas J Kyriakopoulos. A biomimetic approach to inverse kinematics for a redundant robot arm. *Autonomous Robots*, 29(3-4):293–308, 2010.

[92] Matteo Saveriano, Sang-ik An, and Dongheui Lee. Incremental Kinesthetic Teaching of End-Effector and Null-Space Motion Primitives. In *IEEE International Conference on Robotics and Automation*, 2015.

[93] Joseph Sun De La Cruz, Dana Kulić, and William Owen. Online incremental learning of inverse dynamics incorporating prior knowledge. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6752 LNAI:167–176, 2011.

[94] Daniel H Grollman and Odest Chadwicke Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *2008 IEEE International Conference on Robotics and Automation*, pages 3315–3320, 2008.

[95] A Meyering and Helge Ritter. Learning 3D-shape perception with local linear maps. In *International Joint Conference on Neural Networks (IJCNN)*, pages 432–437, 1992.

[96] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[97] Matthias Rolf, Jochen J. Steil, and Michael Gienger. Online Goal Babbling for rapid bootstrapping of inverse models in high dimensions. In *2011 IEEE International Conference on Development and Learning (ICDL)*, pages 1–8. Ieee, August 2011.

[98] Jochen J. Steil. Backpropagation-Decorrelation: Online recurrent learning with O(N) complexity. In *International Joint Conference on Neural Networks*, volume 2, pages 843–848. IEEE, 2004.

[99] Mantas Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, August 2009.

[100] Klaus Neumann, Matthias Rolf, Jochen J. Steil, and Michael Gienger. Learning inverse kinematics for pose-constraint bi-manual movements. In *From Animals to Animats 11*, pages 487–488, 2010.

[101] Christopher Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag New York, 2006.

[102] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J. Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[103] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2(1):321–355, 1988.

[104] Yoan Miche, Benjamin Schrauwen, and Amaury Lendasse. Machine Learning Techniques based on Random Projections. In *European Symposium on Artificial Neural Networks (ESANN)*, number April, pages 28–30, 2010.

[105] G.B. Huang, Q. ZHU, and C. SIEW. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, December 2006.

[106] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 471–482, 2007.

[107] Klaus Neumann, Christian Emmerich, and Jochen J. Steil. Regularization by Intrinsic Plasticity and Its Synergies with Recurrence for Random Projection Methods. *Journal of Intelligent Learning Systems and Applications*, 04(03):230–246, 2012.

[108] Christian Emmerich, R. Felix Reinhart, and Jochen J. Steil. Recurrence enhances the spatial encoding of static inputs in reservoir networks. In *International Conference on Artificial Neural Networks (ICANN)*, number ii in Lecture Notes in Computer Science, pages 148–153. Springer Berlin Heidelberg, 2010.

[109] Arne Nordmann, Christian Emmerich, Stefan Rüther, Andre Lemme, Sebastian Wrede, and Jochen J. Steil. Teaching Nullspace Constraints in Physical Human-Robot Interaction using Reservoir Computing. In *International Conference on Robotics and Automation (ICRA)*, pages 1868–1875, 2012.

[110] Jochen Triesch. A gradient rule for the plasticity of a neuron's intrinsic excitability. *Artificial Neural Networks: Biological Inspirations - ICANN 2005*, pages 65–70, 2005.

[111] Jochen J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural networks : the official journal of the International Neural Network Society*, 20(3):353–364, April 2007.

[112] G.B. Huang, Qin-yu Zhu, and Chee-kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE International Joint Conference on Neural Networks*, volume 70, pages 985–990. Ieee, 2004.

[113] CM Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 1995.

[114] Helge Ritter, Thomas M. Martinetz, and Klaus J. Schulten. Topology-Conserving Maps for Learning Visuo-Motor-Coordination. *Neural Networks*, 2(3):159–168, 1989.

[115] Jochen J. Steil, Christian Emmerich, Agnes Swadzba, Ricarda Grünberg, Arne Nordmann, and Sebastian Wrede. Kinesthetic Teaching Using Assisted Gravity Compensation for Model-Free Trajectory Generation in Confined Spaces. In Florian Röhrbein, Germano Veiga, and Ciro Natale, editors, *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe:*, number April in Springer Tracts in Advanced Robotics, pages 107–127. Springer International Publishing, 2014.

[116] Nan-Ying Liang, G.B. Huang, P Saratchandran, and N Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks.

IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, 17(6):1411–23, November 2006.

[117] Helmut Späth. *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples (Translated by J. Goldschmidt)*. New York: Halsted Press, 1985.

[118] G. A. F. Seber. *Multivariate Observations*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 1984.

[119] G. Schreiber, A. Stemmer, and R. Bischoff. The Fast Research Interface for the KUKA Lightweight Robot. *IEEE ICRA 2010 Workshop on Innovative Robot Control Architectures*, 2010.

[120] D.G. Kendall. A Survey of the Statistical of Shape Theory. *Statistical Science*, 4(2):87–99, 1989.

[121] a. D. Dragan and S. S. Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 32(7):790–805, July 2013.

[122] A.L. Thomaz and C. Breazeal. Transparency and socially guided machine learning. In *5th Intl. Conf. on Development and Learning (ICDL)*, volume 1, 2006.

[123] Anna Lisa Vollmer, Manuel Mühlig, Jochen J. Steil, Karola Pitsch, Jannik Fritsch, Katharina J. Rohlfing, and Britta Wrede. Robots show us how to teach them: Feedback from robots shapes tutoring behavior during action learning. *PLoS ONE*, 9(3):1–12, 2014.

[124] Peter J Rousseeuw and Katrien Van Driessen. A Fast Algorithm for the Minimum Covariance Determinant Estimator. *Technometrics*, 41(3):212–223, August 1999.

[125] Terrence Fong, Charles Thorpe, and Charles Baur. Collaboration, Dialogue, Human-Robot Interaction. In Raymond Austin Jarvis and Alexander Zelinsky, editors, *Robotics Research - The Tenth International Symposium*, volume 6, chapter 5, pages 255–266. Springer Berlin Heidelberg, 2003.

[126] A.M. Okamura. Methods for haptic feedback in teleoperated robot-assisted surgery. *Industrial Robot: An International Journal*, 31(6):499–508, 2004.

[127] M. a. Goodrich, J. W. Crandall, and E. Barakova. Teleoperation and Beyond for Assistive Humanoid Robots. *Reviews of Human Factors and Ergonomics*, 9(1):175–226, 2013.

[128] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, and M. Sassin. Robot programming by Demonstration (RPD): Supporting the induction by human interaction. *Machine Learning*, 23(2-3):163–189, 1996.

[129] T. Kröger. *On-Line Trajectory Generation in Robotic Systems*, volume 58 of *Springer Tracts in Advanced Robotics*. Springer, 2010.

[130] J. Jockusch and Helge Ritter. An instantaneous topological mapping model for correlated stimuli. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings*, volume 1, pages 529–534. Ieee, 1999.

[131] Daniel Seidel. Path planning for a redundant robot manipulator using sparse demonstration data. Master thesis, Institute for Cognition and Robotics, Bielefeld University, 2014.

[132] Daniel Seidel, Christian Emmerich, and Jochen J. Steil. Model-free Path Planning for Redundant Robots using Sparse Data from Kinesthetic Teaching. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4381–4388, 2014.

[133] Lydia E. Kavraki and Steven M. Lavalle. Motion planning. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, number 1, pages 109–131. Springer-Verlag Berlin Heidelberg, 2008.

[134] S. M. LaValle and Kuffner. Rapidly-Exploring Random trees: Progress and Prospects. In B.R. Donald, K.M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Direction*, pages 293–308. A. K. Peters, Wellesley, 2001.

[135] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566 – 580, 1996.

[136] Thomas Villmann, Ralf Der, Michael Herrmann, and T.M. Martinetz. Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, March 1997.

[137] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[138] Stavros D Nikolopoulos and Leonidas Palios. Hole and Antihole Detection in Graphs. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 850–859, 2004.

[139] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[140] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[141] Jean Scholtz. Theory and evaluation of human robot interactions. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, volume 3, page 10 pp. IEEE, 2003.

[142] Jonathan Claassens. An RRT-based path planner for use in trajectory imitation. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3090–3095, 2010.

[143] Javier V. Gomez, David Alvarez, Santiago Garrido, and Luis Moreno. Kinesthetic teaching via Fast Marching Square. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1305–1310, October 2012.

[144] Rosen Diankov, Nathan Ratliff, D Ferguson, S Srinivasa, and James Kuffner. Bispace planning: Concurrent multi-space exploration. In *Proceedings of Robotics: Science and Systems IV*, pages 159–166, 2008.

[145] Matthias Behnisch, Robert Haschke, and Michael Gienger. Task space motion planning using reactive control. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 5934–5940, 2010.

[146] Fabio Dalla Libera, Takashi Minato, Ian Fasel, Hiroshi Ishiguro, Emanuele Menegatti, and Enrico Pagello. Teaching by touching: An intuitive method for development of humanoid robot motions. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 352–359. IEEE, November 2007.

[147] Freek Stulp and Olivier Sigaud. Many regression algorithms, one unified model: A review. *Neural Networks*, 69:60–79, April 2015.

[148] Arne Nordmann, Matthias Rolf, and Sebastian Wrede. Software Abstractions for Simulation and Control of a Continuum Robot. In *SIMULATION, MODELING, and PROGRAMMING for AUTONOMOUS ROBOTS*, pages 113–124, 2012.

[149] Control basis framework. `https://github.com/fps/CBF`. Accessed: 2015-05-25.

[150] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics, 2008.

[151] Johannes Wienke and Sebastian Wrede. A middleware for collaborative research in experimental robotics. In *2011 IEEE/SICE International Symposium on System Integration (SII)*, pages 1183–1190. IEEE, December 2011.